TECHNISCHE UNIVERSITÄT BERLIN
FAKULTÄT FÜR ELEKTROTECHNIK UND INFORMATIK
LEHRSTUHL FÜR INTELLIGENTE NETZE
UND MANAGEMENT VERTEILTER SYSTEME
UND
LEHRSTUHL FÜR SECURITY IN TELECOMMUNICATIONS

# An Empirical Evaluation of Misconfiguration in Internet Services

vorgelegt von
Tobias Fiebig geb. Wrona, MSc
Geboren in Dissen am Teutoburger Wald
Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades
DOKTOR DER INGENIEURWISSENSCHAFTEN (DR.-ING.)

**Promotionsausschuss:**

| | |
|---|---|
| Vorsitzender: | Prof. Dr.-Ing. Sebastian Möller, TU Berlin |
| Gutachterin: | Prof. Anja Feldmann, Ph. D., TU Berlin |
| Gutachter: | Prof. Dr. Jean-Pierre Seifert, TU Berlin |
| Gutachter: | Prof. Dr. Steve Uhlig, Queen Mary University of London |

Tag der wissenschaftlichen Aussprache: 16. Juni 2017

Berlin 2017

*Für meinen Opa.*

# Abstract

Within the past thirty years we have seen computers rise from room sized niche equipment to handy pocket sized devices found in every household. At the same time there has been a significant increase in the research effort on computer security. The literature is full of sophisticated attacks to obtain confidential information from computer systems, compromise them, or prevent them from being used at all. Simultaneously, mitigations to these attacks are as well studied. Technically, current attacks could be mitigated by deploying these techniques. In fact, there is a constant stream of new, complex techniques to ensure the confidentiality, integrity, and availability of data and systems.

However, even the recent past has not been short of security incidents affecting billions of people. Yet, these incidents are usually neither enabled nor mitigated by these complex techniques. On the contrary, we find that these breaches are usually caused by something far more simple: Human error in deploying and running network services, e.g., delayed software updates, or, no authentication and authorization being configured even though it would have been available. We refer to these as security misconfigurations.

In this thesis we empirically investigate the nature of security misconfigurations. Specifically, to approach the unscoped question, if and how security misconfigurations introduce challenges to the security of Internet services, we investigate: (i) What are security misconfigurations (in Internet Services), (ii) How we can measure security misconfigurations, and, (iii) How security misconfigurations can be measured and mitigated in today's as well as tomorrow's IPv6 Internet.

We find that complex attacks are commonly prevented by easy *to implement* technical mitigations. In contrast, misconfiguration based issues require a more demanding approach that is focused on the personnel operating Internet services. Patching humans is incredibly hard. Furthermore, our literature study indicates that there are already problems in the design of protocols. A good design can prevent misconfigurations, while a bad design can lead to multiple, easily misconfigured implementations.

Current mitigation techniques are focused on identifying and contacting affected operators, so they can take appropriate action and remove the misconfiguration. However, this process heavily relies on security scans of the whole Internet. While this is feasible with IPv4, the current Internet Protocol Version, a similar bruteforce approach is unfeasible for the larger address space of the upcoming IPv6. Hence, we develop and evaluate a new methodology that enables researchers to perform security scans of IPv6 connected hosts.

Hence, in summary, this thesis outlines the first steps towards addressing security misconfigurations as an Internet wide issue with an exploratory approach rooted in empirical measurements. We conclude this work by discussing various paths of future research, which should be pursued to reduce the impact of security misconfigurations on the Internet.

# Zusammenfassung

In den letzten zwanzig Jahren haben sich Computer von einer raumfüllenden Nischentechnik zu omnipräsenten Mobilgeräten entwickelt. Gleichzeitig haben sich Forschungsanstrengungen zur Computersicherheit intensiviert. In der Fachliteratur finden sich unzählige ausgeklügelten Angriffen auf Computersysteme, um aus diesen Informationen zu stehlen, sie zu kompromittieren oder ihre Nutzung gleich ganz zu verhindern. Zeitgleich sind Schutztechniken gegen diese Angriffe gut untersucht und würden, wenn sie Anwendung fänden, diese Angriffe unterbinden. In der Tat werden regelmäßig neue, komplexe, Techniken vorgestellt um die Vertraulichkeit, Integrität, und Verfügbarkeit von Daten und System zu garantieren.

Trotzdem finden sich in der jüngeren Vergangenheit unzählige Sicherheitsvorfälle, bei denen Milliarden von Menschen betroffen waren. Jedoch wurden diese Zwischenfälle in aller Regel nicht von komplexen Angriffen verursacht oder durch ebenso komplexe Verteidigungstechniken verhindert; im Gegenteil: Wir müssen feststellen, dass sie einfach durch menschliches Versagen beim Betrieb von Internetdiensten ausgelöst werden, z.B. durch verzögerte Softwareupdates, oder fehlende Authentifikations- und Authorisierungsregeln, obwohl solche vorhanden wären. Wir nennen diese Fälle sicherheitsrelevante Fehlkonfigurationen.

Um die offen gehaltene Frage, ob und wie sicherheitsrelevante Fehlkonfigurationen die Sicherheit im Internet beeinflussen, anzugehen, wenden wir uns in dieser Arbeit drei Kernfragen zu: (i) Was sind sicherheitskritische Fehlkonfigurationen (in Internetdiensten), (ii) Wie können wir sicherheitskritische Fehlkonfigurationen messen, und (iii) Wie können wir sicherheitsrelevante Fehlkonfigurationen im heutigen wie zukünftigen IPv6 Internet messen und beheben.

Unsere Untersuchungen zeigen, dass komplexe Angriffe meist durch vergleichbar einfach *umsetzbare* Techniken verhindert werden können. Im Gegensatz dazu sind Fehlkonfigurationen eine grössere Herausforderung und verlangen eine genauere Auseinandersetzung mit dem Personal, welches Internetdienste betreibt. Unsere Literaturarbeit zu Fehlkonfigurationen zeigt, dass bereits beim Design von Internetdienstprotokollen wichtige Weichen gestellt werden. Ein gutes Design kann das Auftreten von Fehlkonfigurationen verhindern, während ein schlechtes Design leicht zu einfach fehlkonfigurierbaren Implementationen führen kann.

Derzeit liegt der Fokus von Versuchen Fehlkonfigurationen zu beheben darauf, die betroffenen Anbieter zu identifizieren und zu informieren, damit diese dann angemessen reagieren können. Obwohl dieser Prozess direkt an den Personen ansetzt, die Dienste betreiben, ist er unzureichend robust gegen Änderungen an der Architektur des Internets, d.h. gegen die Einführung von IPv6. Um diese Herausforderungen auch in Zukunft angehen zu können haben wir eine neue Methodik entwickelt, welche den ersten Schritt darstellt, um bestehende Techniken und Abläufe an zukünftige Änderungen in der Architektur des Internets anzupassen. Diese Prozesse erfordern jedoch Sicherheitsabtastungen des gesamten Internets. Obwohl

dies mit der aktuellen Internet Protokol Version, IPv4, möglich ist, verhindert der große Adressbereich des kommenden IPv6 solche Sicherheitsabtastungen. Daher entwickeln wir eine neue Methode, welches es Forschern ermöglicht IPv6 Systeme auf Sicherheitsprobleme abzutasten.

Zusammenfassend stellen wir in dieser Arbeit die ersten Schritte in zur Bearbeitung von sicherheitskritischen Fehlkonfigurationen als internetweites Problem vor, wobei wir einen explorativen Ansatz verwenden, welcher auf empirischen Methoden aufbaut. Zum Abschluss dieser Arbeit stellen wir im Weiteren mögliche weitere Forschungsansätze vor, welche zur Reduktion von sicherheitskritischen Fehlkonfigurationen im Internet verfolgt werden sollten.

# Publications

Scientific contribution is an emergent property of a published article that can not be reduced to specific sentences. This thesis is based on the following papers, which have been integrated with the entirety of their main matter with minor adjustments to preserve the scientific value of the individual works. I do acknowledge that this may lead to textual overlap between this thesis and those of my co-authors. To enable the reviewers to assess my personal contribution, this sections also provides a summary of my personal contribution to each paper. I informed my co-authors about the use of these works to the feasible extent. All my collaborators in these works are listed as authors on them.

I would like to express my gratitude to the co-authors of the following published papers. The research endeavor presented in this thesis would have been impossible without their interdisciplinary collaboration.

## Pre-published Papers

### Peer-Reviewed International Conferences and Workshops

"**Something From Nothing (There): Collecting Global IPv6 Datasets From DNS**", Tobias Fiebig, Kevin Borgolte, Shuang Hao, Christopher Kruegel, Giovanni Vigna, Passive and Active Measurement: 18th International Conference, 2017.

Being the lead author, for this work I adapted a methodology for enumerating IPv6 PTR records for global use. I designed the experiments, implemented the necessary toolchain and gathered the datasets used in the evaluation. Subsequently I evaluated and discussed the gathered data. Furthermore, I identified, conducted, and evaluated relevant case-studies.

"**A One-Year Perspective on Exposed In-memory Key-Value Stores**", Tobias Fiebig, Anja Feldmann, Matthias Petschick, Proceedings of the 2016 ACM Workshop on Automated Decision Making for Active Cyber Defense, 2016.

As the lead author of this paper, I gathered the historic dataset utilized in the case study and performed the subsequent analysis on that data. Furthermore I structured the set of possible attacks on exposed key-value stores and dependent systems. I also performed the analysis of specific operators that expose key-value stores.

"**Security Impact of High Resolution Smartphone Cameras**", Tobias Fiebig, Jan Krissler and Ronny Hänsch, 8th USENIX Workshop on Offensive Technologies (WOOT 14), 2014.

I initiated this work as lead author based on my idea of creating a keylogger that works by utilizing a smartphone's camera to record facial reflections. The work of Jan Krissler on biometry complemented my ideas and I realized that our works can be combined to shine a new light on the security implications of high resolution smartphone cameras. During the research I implemented our attacks on actual smartphones, and performed the experiments to evaluate our attacks. Jan Krissler contributed by extracting fingerprints from the data I gathered and subsequently creating fingerprint forgeries from these images. We were supported by Ronny Hänsch, who joined this project as an expert in computer vision and as senior researcher.

### Technical Reports

"**SoK: An Analysis of Protocol Design: Avoiding Traps for Implementation and Deployment**", Tobias Fiebig, Franziska Lichtblau, Florian Streibelt, Thorben Krüger, Pieter Lexis, Randy Bush, Anja Feldmann, arXiv preprint arXiv:1610.05531

This work was initiated by me as lead author. I collected the set of misconfiguration relevant protocols and implementations for this work. When I created the systematization of protocols over time, Franziska Lichtblau assisted me by providing constant feedback on the feasibility of my approach. Besides this, my work was supported by various co-authors to whom I could delegate tasks, e.g., weaponizing the proof-of-concept protocol-scanner I wrote to study an affected protocol.

## Under submission

### International Conferences

"**Reverse DNS Revisited: A Close Look at Reliability and Completeness**", Tobias Fiebig, Kevin Borgolte, Shuang Hao, Christopher Kruegel, Giovanni Vigna, Anja Feldmann, Under Submission

This work is the logical continuation of my prior work on IPv6 dataset collection. Being the lead author, for this work I continued developing the methodology for collecting IPv6 PTR records presented in an earlier paper. I designed the experiments, implemented the necessary toolchain and gathered the active datasets used in the evaluation. Subsequently I evaluated and discussed the gathered data. Furthermore, I identified, conducted, and evaluated relevant case-studies. I cordially thank my co-authors for supporting me in structuring the presentation of my work and granting me access to the passively collected trace sets.

# Contents

# 1

# Introduction

Starting with the ARPA-NET—the foundation of what we now know as the Internet—computer systems have become an integral part of modern human life. The Internet has lead to an unprecedented influx of novel technologies and trends. While it slowly developed between 1960 and 1990, especially in academic and enterprise environments, the 1990's certainly were the decade in which the general population started to use the Internet. They are followed by the 2000's with the rise of what is now called the "Web2.0", revolutionizing the way content is made available on the Internet. While in earlier times, publishing information required at least some material and financial means, suddenly everyone could publish their experience, opinions and world-views, readily accessible to anyone else. Along with this, the size of systems connected to the Internet changed. The Internet started with the first room sized systems in the 1960's. These systems shrank to desk size in the late 1980's, to portable size by the late 1990's and early 2000's, and by now—with smartphones—even fit into a pocket.

With the current rise of IoT, the Internet of Things, these developments will go even further. IoT drastically changes the very nature of the Internet, by networking everyday items, from door controls, fire detectors, toasters and fridges, up to whole factories and buildings. How this development has reached every part of human life is best illustrated with the first toys for exclusive use by adults being connected to the Internet becoming available (We-Vibe, 2016). With these developments, modern technology is clearly on the verge of reaching the most intimate aspects of our modern lives.

These new features and networked systems were accompanied by an equally vast set of new security mechanisms. Complex encryption schemes have been devised to protect the confidentiality of data and various authentication mechanisms have been developed to prevent unauthorized access. Yet, when we look at recent security incidents, it seems like security *did not* grow as fast as networked systems infused our world. This is ideally illustrated by the following three examples: (i) Petabytes of personal data stored in unprotected key-value stores and other NoSQL databases (Binary Edge, 2015a; MongoDB website, 2015) accessible via the Internet without authentication. (ii) A DDoS (Distributed Denial of Service) attack that disabled the heating of apartment complexes in Finnland in the middle of the winter (Mathews, 2016). (iii) Researchers demonstrating how an attacker can compromise the traffic light control of a whole city (Ghena et al., 2014).

Initially, (criminal) adversaries on the Internet were individuals and groups motivated by gaining reputation (Voiskounsky and Smyslova, 2003). This shifted towards organized crime groups that perform security breaches to obtain funds (Wall, 2007). Now, in addition to the criminal groups previously dominating the attacker landscape, nation state sponsored and operated groups can be found on the Internet.

The first noteable event in this regard has been documented by Mandiant, an US-based IT company. In their 2013 report the company provides an in-depth description of a group that continously compromised high-profile industry clients of Mandiant, to obtain trade secrets (Mandiant Intelligence Center, 2013). Mandiant suggests that this group is in fact a nation state operated group located in China. Since then, attacks on public, industrial and government systems in western states have been commonly attributed to China. However, the documents released by Edward Snowden (G. Greenwald et al., 2013) indicate that the National Security Agency (NSA) of the United States also operates a program to strategically attack computer systems in other states. Regardless of who created them, tools like Stuxnet (Kushner, 2013), and its cousins Duqu, glame, and gauss (Bencsáth et al., 2012), demonstrate that targeted IT attacks are used to strategically destroy infrastructure in nation state confrontations.

With the developments outlined in the last paragraphs not only IT professionals, but also the average citizen has become aware of the need to have secure systems (Kraus et al., 2015). Yet, IT security is a vague term. In this thesis we define IT security following, among others, Bishop (Bishop, 2003), by: (a) Confidentiality, (b) Integrity, and, (c) Availability. In our context confidentiality means, that any data in a system or in transit between systems is not read or obtained by an unauthorized entity. Integrity is defined as no unauthorized entity being able to change data in a system or in transit between systems. Availability is defined as no unauthorized party being able to disrupt or prevent the use of a system by an authorized entity.

## 1.1 Motivation

In the last section we reiterated the three main properties of security. These are the cornerstones of modern IT security research. Currently, the focus of security research has been on sophisticated attacks against these three properties. At the same time we can find even more sophisticated mitigations to restore confidentiality, integrity and availability in the light of sophisticated attacks. The most paradigm changing attacks in the recent past were certainly those attacks exploiting memory bit-flips (Y. Kim et al., 2014; Pessl et al., 2016), a technique commonly referred to as *rowhammer*, as well as the associated defenses (Aweke et al., 2016). Other prominent new developments defending confidentiality, integrity, and availability are SGX (Schuster et al., 2015) and seperate compartment phones for high security environments (Ning, 2014). Given this focus on complexity, one might expect that successful attacks are relatively rare, and if they occur have been conducted using advanced exploitation techniques.

Yet, when we revisit the examples mentioned earlier, including exposed databases (Binary Edge, 2015a; MongoDB website, 2015), and Internet accessible heating (Mathews, 2016), we find that most noticed attacks are relatively simple. Even though they impact large groups of people they are neither executed using complex attacks nor prevented by sophisticated protection technology. This expands beyond exposed database systems (Binary Edge, 2015a; MongoDB website, 2015), open management ports on IoT systems (Pa et al., 2015), to, e.g., an UEFI signing key on an internal FTP server which was available via the Internet (Caudill, 2013). Conducting an investigation of what taints confidentiality, integrity, or, availability in these and similar cases, we find simple human error to be prominent among the root-causes, e.g., delayed software updates or no authentication and authorization being configured (Meixell and Forner, 2013). For the purpose of this thesis—we refer to a more complete definition in Chapter 2—we refer to the latter as a security misconfiguration. In particular, security misconfigurations occur if the service is deployed on the Internet in such a way that any of the three main security properties - confidentiality, integrity, or availability (CIA) - can be tainted.

## 1.2 Research Questions

Considering the frequency of incidents that fit into this broad definition of security misconfigurations, the objective in this thesis is answering the question if and how security misconfigurations introduce challenges to the security of Internet services.

This question dictates an explorative approach for our research. However, even with an explorative approach aimed at the intersection between systems operation and human factors, this question is to broad. Hence, to provide an initial starting point for research in the direction of security misconfigurations, we identified four distinct sub-questions. Answering these sub-questions will allow us to scope and continue future research on security misconfigurations. Furthermore, these questions partly rely upon each other, i.e., results from one question are necessary to investigate subsequent sub-questions.

### Sub-Question 1

*What are security misconfigurations (in Internet Services)?*

Human error induced security challenges—or security misconfigurations—in networked computer systems have long-since been known as a significant problem. However, so far, there has not been a consistent definition of security misconfigurations. This question aims at finding a definition for security misconfigurations, which distinguishes them from other security challenges found in IT systems. By providing such a definition, we can subsequently characterize and cluster incidents that fit this definition to get a better understanding of *what* security misconfigurations are.

### Sub-Question 2

*How can we quantitatively evaluate the prevalence of security misconfigurations, and which example factors influence the observed number of misconfigurations?*

The next step towards a more complete picture of security misconfigurations is investigating them on the Internet. Specifically, this pertains to how security misconfigurations are currently measured on the Internet in general. This question also entails applying these techniques, i.e., using this technique to investigate specific security misconfiguration instances.

### Sub-Question 3

*How can we measure misconfigurations in the future, given changes to the Internet, i.e., the introduction of IPv6?*

Given the currently ongoing exhaustion of IPv4 (Richter, Smaragdakis, et al., 2016), migrating to IPv6 has become imperative. However, while the whole IPv4 space can be scanned in minutes (Durumeric et al., 2013), similar scans for IPv6 would take millions of years. Hence, the question arises how we can scan for security misconfigurations in an IPv6 Internet.

## 1.3 Methodology

The diverse research questions that are investigated in this thesis require a diverse set of methodologies. As the research objective is explorative, we lack specific and testable hypotheses. Instead, we utilize observational methodology suitable and valid for the applied nature of our work.

We motivate our work with reflections on security misconfigurations. We then perform comparative prototyping to relate our observations on misconfigurations to more complex attacks. In the associated work, we demonstrate the feasibility of our attacks by executing them under laboratory conditions as described in Chapter 3.

Our work on protocol design is the foundation of our methodological analysis of misconfiguration and protocol design in Chapter 4. This work systematically evaluates current and historical protocol definitions. The goal of this process is to identify, if these specifications have led to Internet service implementations that are prone to misconfiguration. Appropriately, our methodology includes a careful analysis of the pre-existing literature in the field. Due to its focus, this work is again supported by scans and reports of misconfigurations' occurrence on the Internet. Subsequently, the perspectives gained in the literature

research are used to further explore the overall impact of misconfiguration based attacks in Chapter 11.

To effectively assess the impact of security misconfiguration, we build on the coarse grained observations from Chapter 4 with a more detailed approach in Chapter 6. Specifically, we observe the impact of misconfiguration on two popular in-memory key-value stores, Redis and memcached. These two were chosen as they are instances of the most prominently misconfigured systems following Chapter 2 and Chapter 4. Furthermore, the misconfiguration issues around them have been made public to the operations community quiet recently and nearly at the same time (John Matherly, 2015a). Public historic datasets on the mitigation efforts were already available at the time of our investigation (ShadowServer Foundation, 2014). Furthermore, Redis exhibited unusual events during this timeframe, which may provide promising anecdotal observations to be used for building testable hypotheses on misconfiguration mitigation (Sanfilippo, 2015). In summary, this allows us to conclusively document the impact even small misconfigurations can have on the security of systems. Furthermore, we are able to observe how the networking community handles mitigation for misconfigured services. This allows us to identify security scanning as an important tool in the process of detecting, measuring, and mitigating misconfigured services.

Security scanning is an important tool in mitigating and identifying misconfiguration on the Internet. Since the publication of zMap (Durumeric et al., 2013), large scale brute-force security scanning has become feasible for the IPv4 Internet. However, for IPv6, the new Internet Protocol version, a brute-force approach is no longer feasible. Hence, in Chapter 7 we review the literature on how IPv6 security scans might be conducted. We find points for improvement in these approaches, and establish requirements for an admissible dataset to support IPv6 security scans: Public availability, addresses of network devices and servers, and, a sufficient size. Note that we do not strictly define "sufficient", but instead aim at "as large as possible".

Based on these requirements, we introduce a new methodology to enumerate IPv6 enabled systems for subsequent security scans. The main objective here is developing and evaluating methodology that may enable *average* researchers with *no access to privileged vantage points* to conduct security scanning of IPv6 networks. We are able to improve existing approaches for IPv6 address reconnaissance and are the first to present a globally applicable framework using this methodology. The toolchain allows us to experimentally evaluate our method to enumerate assigned IPv6 addresses on the Internet. Based on our results we provide additional case-studies that illustrate the security impact of our technique. Specifically, we first investigate the security impact of the collected data without additional security scans. This already allows us to demonstrate a significant security impact. In addition, we also perform security scans on the enumerated IPv6 address dataset. Using the results of these scans, we demonstrate a critical misconfiguration in a vendors Internet backbone routers, which is *exclusively* exposed via IPv6, i.e., is not visible in classical IPv4 only security scans.

As our technique is highly dependent on reverse DNS, we subsequently evaluate the admissibility of rDNS as a datasource, as several authors suggest that it is insufficiently main-

tained, e.g., (Gao et al., 2013). Contrary to earlier findings, we find rDNS to be well maintained, with IPv6 rDNS being even better maintained than IPv4 rDNS.

## 1.4 Structure of the Thesis

We investigate the nature of security misconfigurations in Part I. We introduce the necessary terminology and provide definitions used in the remainder of this thesis in Chapter 2. We then compare and distinguish security misconfigurations from more complex attacks in Chapter 3. Subsequently, in Chapter 4, we discuss one possible misconfiguration facilitating factor, i.e., the paradigms under which the application layer protocols implemented by Internet services are designed.

In Part II of this thesis, we take a closer look at how security misconfigurations can be identified on the Internet using active measurements, and how these are used for active measurements. We first re-iterate how scans on the Internet are conducted in Chapter 5. Subsequently, we discuss a case study of exposed in-memory key-value stores in Chapter 6. We specifically selected key-value stores for our case study based on the results from Part I.

As discussed in Chapter 5, the introduction of IPv6 leads to new challenges for scanning the Internet. Hence, Part III of this thesis is concerned with adapting techniques to collect global IPv6 datasets. We discuss requirements for IPv6 security scanning in Chapter 7. Based on the observations and requirements established there, we adapt a technique for global applicability in Chapter 8. Subsequently, We evaluate the admissibility of our datasource in Chapter 9. We conclude Part III of our thesis with several case-studies that underline the promise held by our technique for network measurement as well as network security scanning.

Finally, we conclude in Part IV, with a brief discussion of our observations in Chapter 11 followed by a conclusion and outline of further work in Chapter 12.

# Part I

# Security Misconfigurations

# 2

# Security Misconfigurations: Introduction and Definitions

In the introduction of this thesis we set its core topic: Security misconfigurations. However, so far we only discussed anecdotal descriptions of individual misconfiguration incidents. To conduct a structured investigation of the matter, we first have to find definitions for the key-subjects in this work, specifically *Internet Services* and *Security Misconfigurations*. Similarly, we also have to define what a *Security Misconfiguration Incident* is. In this chapter we introduce and define the terms we use in the remainder of this work.

## 2.1 Internet Services

Internet Services are, just by their name, services that are connected to the Internet, i.e., a sub-set of network services. However, finding a clear definition of a network services, and even for networks, from the early times of the Internet is surprisingly difficult.

Definitions, at that time, had a different context, which makes them hardly usable today. For example, Roberts, one of the pioneers—if not *the* pioneer—in the field, defines a network as follows: *"[...] a network between computers, not including the network of typewriter consoles surrounding each computer."* (Roberts, 1967). This distinction between periphery and the computers, and the associated (electrical) networks made sense in times where single, large, computers where interfaced by a set of various physical terminal types (Schicker and Duenki, 1976).

Robert's later work, together with Wessler, provides a more tractable definition of a network for our purposes: *"[...] a computer network is defined to be a set of autonomous, independent computer systems, interconnected so as to permit interactive resource sharing between any pair of systems."* (Roberts and Wessler, 1970).

This does not directly entail a definition for a network service. However, from his definition we can gather that there are programs which run *on* a network, i.e., use the opportunity to perform that *interactive resource sharing between any pair of systems* (Roberts and Wessler, 1970). Following Roberts and Wessler, this exchange of information takes place using

certain protocols. In the context of computer networks an *"[...] intercomputer protocol, [is] the language the systems use to talk to one another"* (Roberts and Wessler, 1970).

Next, we have to extend networks to the Internet. The Internet has become a self-defining term over the last decades, as a description for the global "network of networks". Here a single network refers to a set of routers under one administrative control, also called Autonomous Systems (AS), can exchange information with each other. The most defining property of networks that take part in the Internet is their use of the Internet Protocol, either in version 4 (Postel, 1981b) or version 6 (Deering and Hinden, 1995, 1998) to exchange information between hosts.

For computers to exchange information via networks using protocols, the terms "server-host" for the host providing a resource and "user-host" for the host used by a user to access the resource has emerged (Rulifson, 1969; Shapiro, 1969). Since then, these terms have evolved to the notions of "client" and "server", not referring to hosts, but—in the context of client-server-architectures—the programs that have the function of accessing or providing a remote resource (Tanenbaum and Van Steen, 2007). Clients and servers again use dedicated protocols that reside atop of the Internet Protocol in version 4 and 6 to exchange information with each other. For the purpose of this thesis, an Internet service is thus a server program which communicates via the Internet.

### 2.1.1 Definition 1: Internet Service

Hence, in short, an *Internet Service* is defined as follows:

*An Internet service is a program running on a host which makes a resource available to (client) programs running on other computers via the Internet, i.e., using protocols that are run atop the Internet Protocol in either version 4 or 6.*

## 2.2 Security Misconfigurations

The program that provides an Internet Service usually has to be customized for the specific use-case and network environment. This includes whether the service uses authentication, to which local Internet Protocol it responds to, and further service-specific aspects. For example, for a mail server one of the configuration option is the domain it should receive mail messages for. The process of configuring an Internet service and making it available to its clients is typically called "deploying" an Internet service. Configuration information is often stored in a plain text file (J. Zhang, Renganarayana, et al., 2014), but can also reside in a data-base or any other information bearing system. Client programs, similar to server side programs, have to be configured as well. However, for the purpose of this thesis, they are out of scope.

A misconfiguration occurs, if the configuration of an Internet Service contains an error that leads to a behavior that has not been intended by the entity responsible for configuring/deploying it. We call this entity the operator.

Furthermore, an error in the configuration of one Internet Service may impact the operation of another Internet service, for example when insufficient authorization of a database prevents access for the corresponding application, which thereby becomes unavailable. Similarly, the configuration of the network and network devices may impact an Internet service. For example, as in the prior example, a misconfigured firewall may simply prevent access to the database server. We capture this complexity around the operational environment of an Internet service by referring to it as "the deployment", i.e., the way an Internet service is deployed. A deployment contains not only a specific Internet service (program) on a single host, but the whole operational environment controlled and/or utilized by the operator of a service.

With respect to the security aspect of misconfigurations, we first have to revisit our notion of security. Security, especially in the context of computer networks, is a vague term. In this thesis we define IT security following, among others, Bishop (Bishop, 2003), by: (a) Confidentiality, (b) Integrity, and, (c) Availability. In our context confidentiality means, that any data in a system or in transit between systems is not read or obtained by an unauthorized entity. Integrity is defined as no unauthorized entity being able to change data in a system or in transit between systems. Availability is defined as no unauthorized party being able to disrupt or prevent the use of a system by an authorized entity. Hence, a security misconfiguration occurs if the way and Internet service is deployed leads to the tainting of one of these three security principles.

For an misconfiguration to occur it has to be the case that the service could have been deployed in a different way, which would have prevented the tainting of any security principle, without preventing the service from being used, i.e., restricting availability for legitimate clients.

### 2.2.1 Definition 2: Security Misconfigurations

In summary, our definition for Security Misconfigurations is:

*A security misconfiguration has occurred, if the way an Internet service is deployed, i.e., configured, enables an attacker to taint either its confidentiality, integrity or availability, and the property could not have been tainted if the service would have been deployed and configured correctly, without restricting availability for legitimate clients.*

## 2.3 Security Misconfiguration Incident

A deployment being misconfigured does not necessarily entail that this misconfiguration is detectable. Mahajan et al. (Mahajan et al., 2002), for example, documented this for the configuration of routers. In fact, detecting misconfigurations in deployments—regardless of them being security misconfigurations or not—is a significant challenge (J. Zhang, Renganarayana, et al., 2014).

The question now is, what constitutes a security misconfiguration *incident*. NIST, the United States National Institute of Standards and Technology, in its *Computer Security Incident Handling Guide* distinguishes between *events* and *incidents*. In there terms, *"[an] event is any observable occurrence in a system or network* (Cichonski et al., 2012). If an event had negative consequences, NIST refers to it as an *adverse event*. Subsequently, they define *computer security incidents* as follows: *"A computer security incident is a violation or imminent threat of violation of computer security policies, acceptable use policies, or standard security practices"* (Cichonski et al., 2012).

Following the definitions of NIST, we consider an event a security misconfiguration incident, if the event fulfills NISTs requirements for a security incident and the root-cause has been a security misconfiguration. Following NISTs notion on an *"imminent threat of violation"*, this does not require an event to be an adverse event or to have been executed with malicious intent. Instead it suffices if the operator *"has a factual basis for believing that a specific incident is about to occur"* (Cichonski et al., 2012). To illustrate: If a database is left configured without authentication, it is not necessary that an unauthorized attacker extracts information from it. Instead it already constitutes an incident, if a benevolent user reports the unsecured system, demonstrating that an attacker ***could*** have unlawfully obtained data.

### 2.3.1 Definition 3: Security Misconfiguration Incident

In correspondence to the NIST definitions, we, for the purpose of this work, define security misconfiguration incidents as:

*A security misconfiguration incident has occurred if the root-cause of a security incident is a security misconfiguration, if an operator obtains knowledge that a security misconfiguration exists on a deployed system the operator is responsible for, or, if a third party obtains knowledge on a security misconfiguration in a deployed system and the operator is not notified.*

## 2.4 Misconfigurations in Practice

Equipped with the three definitions of the previous section, we now provide a brief survey of security misconfiguration based incidents in the recent past reported on in the literature. This gives us the opportunity to gather an explorative overview on six key-aspects of security misconfigurations:

1. **What** commonly happens in security misconfiguration incidents?
2. **Where**, i.e., in which kind of organizations do such incidents happen?
3. **Which** services are commonly affected?
4. **Who** is affected, i.e., how many users and which user-groups?
5. **Why** does the incident happen, i.e., which component of a system is misconfigured?
6. **How** was the incident inflicted, i.e., was it targeted exploitation, or, accidental exploitation or non-malicious notification?

We summarize our findings in Table 2.1. Each incident is listed with a brief description. We furthermore report on the affected organizations' type, e.g., "ISP", the organizations size (large, medium or small), the affected services' type, e.g., internal system orchestration, and, the number of affected users. Furthermore, we report on the nature of the misconfiguration. Specifically, we document if missing authentication or incorrectly configured authorization caused the incident, even though these would have been supported by the service (Auth). We also indicate if the mere exposure of a service was already sufficient to lead to the incident (Exposure), or if the operators incorrectly handled cryptography related tasks (Crypto). The latter concerns cases where, e.g., keys are accidentally published or where an operator configures cipher settings that are too weak. Finally, we also indicate if the incident was intentional and malicious, the result of a non-malicious (responsible) disclosure, or caused accidentally (an attacker broke a service by mistake).

Investigating the types of incidents, i.e., what has happened, we find that the majority of events relates to exposed databases and unintentional publication of data. The remainder of incidents in our list concerns the exploitation of weak authentication to assimilate systems into botnets for subsequent use and (accidental) denial-of-service (DoS) attacks on exposed embedded systems and customer premise equipment (CPE).

A first glance at the affected companies yields small not-for-profit organizations and large multi-national companies alike. Furthermore, we find that misconfiguration related incidents stretch over all aspects of network driven businessmodels: From service providers to (e-)government operations. However, we note that "ultra large" trans national companies, e.g., Google and Facebook, rarely suffer from large misconfiguration based incidents. Similarly, we hardly find "traditional" companies, e.g., from the manufacturing industry among companies affected by a single large incident. Yet, we also recognize that "traditional" companies are regularly found among the affected parties in IoT related incidents, that affect a variety of various organizations. We conjecture that this is due to these companies more rarely running their own large-scale self-developed infrastructures, while still being susceptible to the deployment of unsecured IoT devices by untrained staff.

| Description | Org. Size | Org. Type | Affected Service | Affected Users | Misconfiguration | | | | Ref. |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | Auth | Exposure | Crypto | Targeted | |
| Around 2012, the carna botnet infected hundreds of thousands of Internet accessible embedded devices exploiting weak (default) credentials. Subsequently, this botnet was used for Internet measurements. | Various | Various | IoT | ? | ● | | | ● | (Krenc, Hohlfeld, et al., 2014) |
| UK health data published in a web folder. | Small | Datascience | Webserver | $\geq 5m$ | ● | | | ◐ | (Solon, 2014) |
| MacKeeper lost user data from its customers due to an insufficiently protected MongoDB. | Medium | Software Provider | Database | $\geq 13m$ | ● | | | ◐ | (Krebs, 2015) |
| Attacks on a defective TR/069 implementation took Deutsche Telekom customers offline. | Large | ISP | Remote Management | $\geq 900k$ | | ● | | ○ | (Weinmann, 2016) |
| A DoS attack on IoT devices disabled the heating of a housing complex. | Small | Housing | IoT | $60k$ | | ● | | ? | (Mathews, 2016) |
| The AES key for encrypted backups of mitfahrzentrale.de is published along with the backups | Medium | Shareconomy | Backups | $\geq 600k$ | | | ● | ● | (Schirrmacher, 2016) |
| An unprotected MongoDB exposed the data of ModBSolutions.com's users. | Medium | Datascience | Database | $\geq 58m$ | ● | | | ● | (Dissent, 2016) |
| An unauthenticated database exposed the voters registry of Mexico. | Large | Government | Database | $\geq 87m$ | ● | | | ◐ | (Baraniuk, 2016) |
| The Mirai botnet takes down dyn.com, a DNS hosts, using a more than 1.2 Tbps dDoS attack. Subsequently, hundreds of millions of users are unable to reach some of the most popular Internet services, including github.com, cnn.com, bbc.com, and twitter.com. | Various | Various | IoT | $\geq 100m$ | ● | | | ● | (Hilton, 2016) |
| An unauthenticated CouchDB instance exposed records on American voters. | Medium | Datascience | Database | $\geq 154m$ | ● | | | ◐ | (Vaas, 2016) |
| Countless exposed databases (MongoDB, Hadoop, CouchDB) on the Internet have been encrypted and held for ransom | Various | Various | Databases | ? | ● | | | ● | (Spring, 2016) |
| Users of .onion (TOR hidden service websites) were exposed due to installed Apache modules. | Unknown | TOR | Webserver | ? | | ● | | ◐ | (Sultan, 2016) |
| skirtclub.co.uk, a lesbian dating site focused on bi/inclosed users, exposed pictures of users due to incorrect webserver permissions. | Small | Onlinedating | Webserver | $\geq 5k$ | ● | | | ◐ | (Kolosowa and Hoppenstedt, 2017) |

Targeted: ●: yes; ◐: non-malicious exposure; ○: accidental; ?: unknown;

Table 2.1: A summary of misconfiguration related incidents.

When investigating the types of affected Internet service types, we find data providing systems to be dominating. Specifically web and database system misconfigurations account for the majority of incidents. This correlates well with the earlier observation that observable misconfiguration incidents mostly coincide with a violation of confidentiality. This, however, may be a bias induced by either, (i) integrity and availability related incidents being less likely to be clearly attributed to misconfiguration, or, (ii) confidentiality related events being more likely to receive publicity.

However, a striking observation on incidents which were reported to the public is the absence of severe incidents involving industry control systems/SCADA/Industry 4.0 installations, i.e., cyber physical systems. It has been a common topic in the past that these systems are exposed on the Internet, commonly without authentication. Just recently, Lundgren presented another IoT/Industry related protocol exhibiting thousands of instances on the Internet that can be controlled without authentication (Lundgren, 2017). According to him, these devices range from household IoT installations to nuclear reactors. The only incident connected to cyber physical systems we found was a housing complex in Finnland that suffered from a DoS attack on the accidentally exposed heating system. However, in that case, similar to a Deutsche Telekom scan, the devices were not targeted specifically. Instead, they were exposed on the Internet and succumbed to an overburdening amount of Internet background radiation, i.e., scanning traffic (Janita, 2016).

Looking at the number of affected users, we find that just our example list of severe incidents has affected millions of users. We find single incidents that affect not a large user base, but severely impact extremely private areas of the affected users' life. Hence, in summary, we can conclude that misconfiguration related incidents do have a dramatic impact on end-users.

When we turn towards misconfiguration related incidents that directly expose user data, we find that many of these are not malicious. In fact, most relate to press coverage of a vulnerable database system, after responsible disclosure has been conducted. In those cases where malicious action was tied to the misconfiguration incidents, we find that the obtained data is simply sold on the Internet, e.g., (Dissent, 2016).

### 2.4.1 Monetization as Exploitation Catalyst

These observations are surprising, especially in combination with the observed lack of incidents involving cyber physical systems. Given that these are commonly suffering from security misconfigurations (Lundgren, 2017) and there are reports that cyber physical systems are attacked using complex attacks with the sole purpose of causing harm (Bundesamt für Sicherheit in der Informationstechnik, 2014), we would have expected various sever incidents involving cyber physical systems.

We conjecture that attackers are most interested in monetizing vulnerabilities. While it is hard to sell access to, say, a security camera in a remote part of the world, it is surprisingly

easy to monetize a botnet (Santanna et al., 2015). Similarly, we see that databases containing financial information (bank accounts and credit card information) tend to be connected to incidents with malicious intent (Schirrmacher, 2016). With personal data apparently loosing value, new trends emerge where the data in unprotected databases is encrypted and held for a ransom (Spring, 2016).

This assumed connection between attackers and their desire to monetize vulnerabilities may also explain why we have not yet seen the *direct* exploitation of various misconfigurations, e.g., openly accessible VNC services. The attackers on the Internet currently do not have a way of monetizing these vulnerabilities. In turn, this means that the only thing that prevents attackers from causing severe incidents, including physical harm, by exploiting, e.g., Industry 4.0 devices, is their lack of ways to monetize them. If this is true, many future incidents are yet to come. Either, when criminals found a way to monetize such attacks, or when physical harm is the actual target of an attacker.

# 3

# Complex Attacks vs. Misconfigurations

In the last chapter, we provided basic definitions of terminology we need for working with misconfigurations and described various major incidents which took place in the last years which have misconfigurations as root-cause. However, in the public perception highly sophisticated attack *possibilities* regularly cause significant hype, while actual *incidents*, even those with misconfigurations as root-cause, are commonly interpreted as "inevitable" events caused by omnipotent "hackers." A prime example for the former is "Cloak and Dagger" (Fratantonio et al., 2017). Similarly, more easily exploitable, and hence practical, attacks like "Drammer" (Veen et al., 2016) and "Stagefright" (Drake, 2015) gained significant public attention. The latter point of incidents attributed to inevitable hackers is well illustrated by the heating-incident in a Finish apartment complex (Janita, 2016) first attributed to a targeted attack, or the very similar event at Deutsche Telekom (Weinmann, 2016).

Despite this public recognition, practical exploitation of attacks like "Cloak and Dagger" or "Stagefright" is rare. In fact, we note that at the time of writing Google's chief of Android security just commented on this issue at an industry conference (Ludwig, 2017). He reports that Google's internal data demonstrates that complex attacks are a non-issue for smartphone security. He illustrates this point with the stagefright vulnerability (Drake, 2015), an issue in modern Android devices that allows for remote compromises. Since the vulnerability was published, not a single attack exploiting this vulnerability was observed. Indeed, a similarly critical vulnerability, MasterKey, e.g., see (Lindorfer et al., 2014), peaked at merely 8 infections per 1 million devices when the vulnerability was published. Instead, Ludwig finds that malicious applications exploit devices' users: By abusing their trust, or, ingenuous and naive use of apps, these malware groups generate revenue. These assertions correspond to our earlier observation, that *large scale* exploitation needs a clear monetization perspective for the attacker.

To get a better understanding of the hardships one encounters in exploiting such a high-profile attack vector, we document how we discovered and exploited such an attack on modern high-security smartphones. We selected these as an example target, since the surveillance of government officials' phones was a matter of public interest, we opted to investigate attacks on high-profile government officials' smart-phone devices when we started this research. This includes multi-compartment security technologies, e.g., Samsung KNOX (Ning, 2014) or the SIMKO phones used by German government officials.

When drafting our attacks, we focused on the smartphone's camera's, as similar to classical smartphones, these devices include two cameras that can be used from their "insecure" compartments. With advances in technology the resolution of these sensors has constantly increased. While this development provides great convenience for users, for example with video-telephony or because they replace a dedicated digital camera, the security implications of including high resolution cameras on mobile devices has yet to be considered in greater detail.

We demonstrate how an attacker can abuse the cameras—even in modern high-security smartphones—to extract valuable information from a victim. We exploit the front-facing camera to capture users' keystrokes. By observing facial reflections, it is possible to capture user input with the camera. Subsequently, individual keystrokes can be extracted from the images acquired with the camera. Furthermore, we demonstrate that back facing smartphone cameras can be used by an attacker to extract and later forge the fingerprints of a victim. This enables an attacker to perform a wide range of malicious actions, including authentication bypass on modern biometric systems and falsely implicating a person in a crime by planting fingerprints. We conclude this chapter with a discussion on how and why our attacks are extremely unlikely to be exploited in practice.

## 3.1 Motivation

In recent years, smartphones have become ubiquitous and their popularity continues to grow. While sales of PC hardware continue to decline, sales of mobile hardware continue to increase. More and more applications, like banking and mobile payment services, now target mobile platforms as well. Social networks and messaging services are also extremely popular on mobile devices. As a result, modern smartphones contain significant amounts of sensitive data.

With ever-increasing features, new sensors and peripherals are continuously integrated into these systems, the most notable of which are the multi megapixel front- and rear-facing cameras. However, attackers lack the capability of accessing these peripherals directly. Modern mobile operating systems implement fine-grained permission systems to prevent unauthorized access. Users can choose to permit or deny access to certain peripherals at the time of the installation. However, such access restrictions are ineffective as many users will agree to grant access to malicious applications that they willingly install on their devices.

Work by Felt et al. published in 2011 has placed the camera permission in the top 10 of unnecessary, yet commonly requested permissions across the Android app market (Felt, Chin, et al., 2011). Other publications have concluded "that the majority of Android users do not pay attention to or understand permission warnings" (Felt, Ha, et al., 2012). Based on this information a malicious attacker can gain remote access to the device's cameras when a user installs a malicious application. Moreover, numerous local root exploits exist for popular Android devices, allowing an attacker to gain system privileges. This allows an attacker to completely bypass user permission requests (Zhou and Jiang, 2012).

Even though an attacker can gain access to a camera, they are seldom considered security-relevant sensors. The primary focus of previous research has been on the associated privacy issues (Brocker and Checkoway, 2014; Froomkin, 2000). In particular, Simon and Anderson investigated privacy issues of the front facing camera in smartphones by developing an orientation based keylogger (Simon and Anderson, 2013). However, the ever increasing resolution of modern smartphone cameras makes attacks more and more feasible with each generation of new smartphones. Hence, we chose to use the highest resolution cameras currently available on the market. This allows us to evaluate the current as well as future effectiveness of the presented attacks. The key-contributions of this thesis presented in this chapter are:

**Facial Reflection Keylogger:** We present a new method utilizing facial reflections for recording a user's keystrokes. By continuing from work of Xu et al. (Y. Xu et al., 2013), this technique surpasses the performance of previous front-camera based keyloggers (Simon and Anderson, 2013) and removes the need of physical proximity (Y. Xu et al., 2013). Our evaluation on a real device furthermore proves that a complex framework is not necessary for a real-world attack as the extraction process can be performed manually. This allows an attacker to circumvent even most advanced anti malware and keylogging mechanisms like separate operating system compartments on the most recent high-security smartphones.

**Fingerprint Extraction:** We present a new method to extract a user's fingerprints with a mobile phone's camera for creating forgeries. With these forgeries used in our experiments, we were able to bypass the most recent fingerprint readers found in smartphones, as well as traditional fingerprint sensors. An attacker could use these prints to gain access to biometrically secured areas or implicate a victim in a crime by placing false prints on a crime scene. Furthermore, these techniques enable an attacker to circumvent modern fingerprint-based authentication methods, such as those which now become common on modern smartphones (Simonite, 2014). While traditional methods rely on a fingerprint being found on the device itself (CCC, 2013), an attacker can prepare a forgery before obtaining the phone. This solves the issues that a well preserved fingerprint may be unavailable on the device surface and relevant data may be remotely wiped if the user considers the phone misplaced or stolen.

## 3.2 Threat Analysis and Technical Background

Due to their unique characteristics, different threats apply to the front- and rear-facing cameras on smartphones. To fully cover both types of cameras, we introduce a reflection based keylogger that abuses the front-camera and a technique to extract fingerprints with the rear-facing camera. Hence, we present the required background and related work for our reflection based keylogger first. We then continue by introducing the background necessary to understand the fingerprint extraction technique using the back-facing camera of a smartphone. To provide some practical context for our work we then describe how the issues we found may be utilized in a hypothetical attack.

### 3.2.1 Visual Keylogging

Keyloggers are small programs installed on a computer system that extract input information. Due to the unique nature of mobile devices, the acquisition of input information from mobile devices poses a greater challenge to attackers than on the PC platform. On one hand, most of these devices lack a physical input method, i.e. keyboard. Instead, a soft-keyboard is used in these systems, which is displayed on a touch screen and records input based on the section of the screen being touched. On the other hand, advanced privilege separation models have been implemented on mobile platforms, which restrict an attacker from accessing security sensitive functions.

Nevertheless, various publications in recent years have shown that such attacks are possible. While the initial publications are mostly concerned with obtaining touch-input information from the touch-screen of a mobile device (Damopoulos et al., 2012; Sagiroglu and Canbek, 2009) these attack vectors have been mitigated on modern devices. New attack vectors applying heuristic approaches on sensory information available on mobile devices were subsequently discovered. Most prominently, the accelerometer of phones was exploited to determine the orientation of a mobile device (Aviv et al., 2012; Z. Xu et al., 2012).

Visual keylogging utilizing cameras is based on two important strategies. The first strategy has been introduced by Simon and Anderson (Simon and Anderson, 2013) in 2013. With their work, they have demonstrated the privacy issues that arise from the presence of front facing user cameras as orientation information can be reconstructed from pictures taken with it. Following the concept of various accelerometer based keyloggers presented earlier (Aviv et al., 2012; Z. Xu et al., 2012), the user's keystrokes can be extracted from the images. Although their work is promising, the limited orientation resolution extracted from images leads to serious limitations. This means that their approach is limited to $3 \times 3 + 1$ numerical keypads and they could not identify a key press with significantly more than 50% certainty in their empirical study (Simon and Anderson, 2013).

The second strategy is the visual eavesdropping on computer screens and smartphones, also known as shoulder surfing. This technique is concerned with capturing the screen from a relative distance while the screen content and possible inputs can still be reconstructed. Kuhn et al. published one of the earliest works on this matter in 2003 (M. Kuhn and C. Kuhn, 2003), where the authors reconstruct a $32 \times 24cm$ display from a distance of $60m$ by using a professional telescope. Later works (e.g. Backes et al. (Backes et al., 2008)) extended existing approaches by utilizing reflections and consequently overcame the requirement of a direct line-of-sight between observer and target. While the proposed methods were able to successfully recover the typed input in the case of direct line-of-sight attacks, the reconstruction accuracy decreased significantly in cases of even a single reflection. In 2013, Xu et al. (Y. Xu et al., 2013) proposed a new approach that neither depends on the detection of small visual details, nor on a direct line-of-sight. Instead of trying to reconstruct the whole screen content, the method tracks the user's fingers as they move over the screen. The relation of the movement, i.e. pauses, and the position of the fingertips is used afterwards to reconstruct the typed input.

For this purpose, Xu et al. created a semi-automatic framework which analyzes a video file in a multi-step process, automatically performing the steps mentioned in the previous paragraph (Y. Xu et al., 2013). They carried out a range of experiments with varying settings, evaluating distance between $30-50m$ in direct line-of-sight, to $4-10m$ distances for single and $3m$ distance double reflection scenarios. The general results are very promising with $23\%$ $(17/73)$ of the test sentences being perfectly reconstructed, while $92\%$ of them have an METEOR score ((Lavie and Denkowski, 2009)) above $0.5$, which means that they are still understandable by a human. Of the $15$ selected test passwords, $12$ have been reconstructed in $12$ or fewer guesses and no password needed more than $6,000$ guesses. For our comparisons we choose their experiments with a Canon VIXIA camcorder as documented in Table 3.1. The results of their experiments with this camera in a single reflection on sunglasses setting state a perfect reconstruction for a distance of $4m$ and a well understandable reconstruction (METEOR-score of $0.71$) for one of $10m$. The distance between object and reflecting surface is assumed to be $30cm$ in both cases.

As can be seen, the two most important factors of the approach are the quality of the camera and the distance between observer and target. Taking these two factors into account, the issue boils down to the resulting size of the target device in the recording. Xu et al. (Y. Xu et al., 2013) created two formulas to calculate the target's size in the resulting image based on these factors. The target size in pixels per axis in the captured image for a direct observation can be calculated by Equation 3.1 (Y. Xu et al., 2013).

$$Size_{Direct} = \frac{SensorResolution}{SensorSize} \cdot \frac{ObjectSize}{\frac{TargetDistance}{FocalLength} - 1} \tag{3.1}$$

If, however, reflections are involved, the curvature of the reflective surface and the distance of the reflective surface from the target have to be taken into account as well. This leads to Equation 3.2 (Y. Xu et al., 2013).

$$Size_{Reflection} = Size_{Direct} \cdot \frac{1}{\frac{2 \cdot DistanceFromSurface}{CurvatureRadius} + 1} \tag{3.2}$$

### 3.2.2 Fingerprints and Biometrics

Fingerprints are the oldest biometric feature that has been actively used (Kücken and Newell, 2005). Fingerprints on ancient seals and clay tablets are even actively used in the field of archaeology (Kamp et al., 1999). In modern times, they are used in forensics to identify a perpetrator among the of suspects of a crime (J. Li et al., 2006).

The question how fingerprints do form during the pregnancy has not yet been conclusively solved (Kücken, 2007; Kücken and Newell, 2005). What however is certain is that folds of the epidermal layer leads to so called ridges and valleys on the outside of fingers and feet. Until a couple of years ago, fingerprints were taken using ink and paper and were compared

manually based on global structures like whirls, arches or loops (J. Li et al., 2006). These patterns are formed by the ridges and valleys of a fingerprint.

Their widespread use started with the development of digital sensors and associated image recognition libraries. These sensors are based on a variety of techniques for capturing the fingerprints, but only two have prevailed (Ratha, Bolle, et al., 2004). Optical sensors use the physical principle of scattered total reflection (Fujieda and Haga, 1997). They have a high resolution, but are relatively large and are therefor mostly found at static stations like for example border controls or access to buildings.

Capacitive sensors on the other hand are considerably smaller. They rely on measuring the difference in the capacity between the skin where the the ridges directly touch the sensor and the air between sensor and skin in the valleys (Tsikos, 1982). Due to their smaller size, they are mainly found in mobile devices like notebooks, and nowadays smartphones.

New sensors also use an additional RF (Radio Frequency) field to measure deeper skin layers. When the finger touches the sensor, an electrical field penetrates the finger and is reflected on lower layers of the skin. Hence the sensor images not the surface, but lower skin layers and is therefore more resilient against dirt and injuries of the upper skin layer (Ratha, Bolle, et al., 2004).

### 3.2.3 Attack Model

As for this attack model, let us consider one of the most high profile targets using the most advanced security mechanisms, being challenged by an equally advanced attacker. This means that we will investigate how the secretary of defense might be targeted by a foreign agency with the goal of stealing state secrets of utmost importance. Being aware of the constant threat of espionage, the ministry decided to issue high security phones with separate compartments for personal and confidential use, e.g., Samsung KNOX (Ning, 2014) or the SIMKO phones used by German government officials, secured by a pin-code and a fingerprint of the user. Within this high profile situation, the secretary of defense also decides to keep confidential documents in a fingerprint and combination secured safe in the office, instead of relying on a physical key, which might get stolen. As the combination is rather complex, he however notes it down in the confidential compartment of her smartphone.

By publishing a rather sophisticated malware posing as a harmless game, but sneaking in the permission to use the systems camera, they can infect a vast amount of phones, including the compartment for personal use on the victims device. While they can use their foothold application to obtain root access on the personal compartment, the confidential compartment and the pin-pad for unlocking it remain out of reach. While the rear-facing-camera is used to extract the targets fingerprints, further information from the personal sector indicates that the target will go on an rather sunny holiday trip. This gives the attacker the opportunity to use a facial reflection based keylogger to extract the pin-code entered on the secure compartment while the target wears sunglasses.

In a final sweep, the foreign agency obtains all the confidential data they desire. While the target is on vacation, a targeted thief retrieves the victims phone. As recommended, the phone is encrypted, but with a replica of the extracted fingerprint and the previously extracted pin-code, the attackers can quickly recover all confidential data. When the target notices the theft 15 minutes later, and immediately issues a remote wipe, the phone is successfully wiped. However, the data, including the combination for the safe in the office, has already been stolen. Shortly after this incident, bribed cleaning personnel uses the recovered combination and a fingerprint replica created from the prints extracted with the mobile device to empty the safe in the office. As it turns out, the attackers were able to extract all confidential data from the phone and conveniently got access to the documents stored in the safe in the target's office. To hide their actions behind confusion, the attackers then use the forged fingerprints on a knife that is used in a murder. With the secretary of defense implicated in a crime, the whole incident goes unnoticed within the ensuing scandal.

We do realize that this scenario may, at first, seem highly unlikely, greatly exaggerated and the stuff of Hollywood fiction—at best. However, we will demonstrate in this chapter, that the outlined attacks are indeed feasible. Furthermore, we would like to point out, that techniques developed from the findings in this papers did in fact allow us to obtain the German Ministry of Defense's fingerprint (Biermann, 2015).

## 3.3 Front Camera: Visual Keylogger

In this section, we demonstrate how techniques on reflection based shoulder surfing can be combined with a smartphone's front camera to construct a reflection based keylogger. To that end, we first consider algorithms proposed in the related work. Based on these metrics, we are able to demonstrate that our approach outperforms other solution. Furthermore we describe a set of experiments demonstrating that such a framework is not even necessary as the input can be obtained manually.

### 3.3.1 Theoretical Applicability

Utilizing the equations presented in Section 3.2, we can calculate if and with which expected accuracy, the framework as proposed by Xu et al. (Y. Xu et al., 2013) is applicable to the new attack vector which we have identified. We decided to utilize the case of the Canon VIXIA in a 4m distance one-time reflection scenario for this comparison. For both cases, we assume an OPPO N1 with a $13cm \times 7.5cm$ screen to be the target device. Equations 3.3 and 3.4 present the estimated $(x, y)$-target size in the source-image for that case using Equation 3.2.

$$X_{VIXIA} = \frac{1920px}{4.84mm} \cdot \frac{130mm}{\frac{4000mm}{57mm} - 1} \cdot \frac{1}{\frac{2 \cdot 300mm}{8mm} + 1} \cong 9.80 \tag{3.3}$$

$$Y_{VIXIA} = \frac{1080px}{3.42mm} \cdot \frac{75mm}{\frac{4000mm}{57mm} - 1} \cdot \frac{1}{\frac{2 \cdot 300mm}{8mm} + 1} \cong 4.50 \tag{3.4}$$

In this case the resulting image would have a $Size_{Reflection}$ of approximately $9.80px \times 4.50px$. If we now consider the case of the OPPO N1 in a scenario where the phone is used on an reflection of itself in the user's eye, we get Equations 3.5 and 3.6.

$$X_{OPPO} = \frac{4160px}{4.4mm} \cdot \frac{130mm}{\frac{300mm}{5mm} - 1} \cdot \frac{1}{\frac{2 \cdot 300mm}{8mm} + 1} \cong 27.41 \tag{3.5}$$

$$Y_{OPPO} = \frac{3120px}{3.6mm} \cdot \frac{75mm}{\frac{300mm}{5mm} - 1} \cdot \frac{1}{\frac{2 \cdot 300mm}{8mm} + 1} \cong 14.50 \tag{3.6}$$

As can be seen, $Size_{Reflection}$ is with $27.41px \times 14.50px$ nearly nine times larger in our case. Using less curvy reflection surfaces like sunglasses would even lead to a larger value for $CurvatureRadius$, hence even larger values for $Size_{Reflection}$.

| Feature | Canon VIXIA | OPPO N1 |
|---|---|---|
| Resolution | $1920 \times 1080px$ | $4160 \times 3120px$ |
| SensorSize | $1/3.2$“ $= 4.84 \times 3.42mm$ | $1/3.06$“ $= 4.4 \times 3.6mm$ |
| Focal Length | $57mm$ | $5mm$ |
| Target Dist. | $4 \cdot 1000mm$ | $300mm$ |
| Surface Dist. | $300mm$ | $300mm$ |
| Object Size | $73 \times 130mm$ | $73 \times 130mm$ |

Table 3.1: Base characteristics of the Canon VIXIA used by Xu et al. (Y. Xu et al., 2013) and the OPPO N1 camera.

A comparison of the observable quality for the presented sensors and reflective surfaces is depicted in Figure 3.1. The quality of the reflection for the case of the Canon VIXIA recording a reflection on sunglasses in 10m distance as produced and published by Xu et al. (Y. Xu et al., 2013) is shown in Figure 3.1(a). Their work produces reflections comparable to the case of the OPPO N1 retrieving reflections from a user's eyes in our experiments, as depicted in Figure 3.1(b). Without applying a framework, it can be determined that the user in Figure 3.1(b) presses a 3 on a numerical keypad. Figure 3.1(c) demonstrates the quality of reflections on sunglasses obtained with the OPPO N1. Please note that the used QWERTY keyboard can be clearly identified as such. Hence, it is save to assume that results obtained with this technique will reach at least the accuracy observed by Xu et al. (Y. Xu et al., 2013), which was already close to the ideal case of obtaining all inputs without any error.

Using Equation 3.2, we can investigated with which resolutions an corneal reflection can be used for effective keylogging, depending on the reflective surface, distance to the face and utilized front camera. In Figure 3.2(a) the size of an reflection of a 5.1" display in the user's eyes is depicted. The red line indicates the reflection size used by Xu et al. (Y. Xu et al., 2013) for nearly perfect reconstruction with their framework using the case of the

(a) Canon VIXIA, Sunglasses with overlay from Xu et al. (Y. Xu et al., 2013)

(b) OPPO N1, Eyeball with overlay

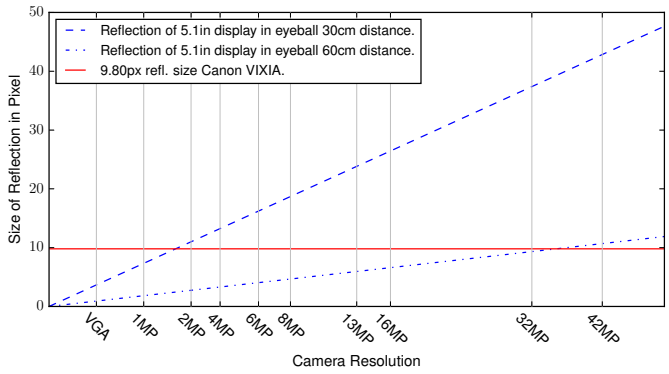
(c) OPPO N1, Sunglasses no overlay

Figure 3.1: Comparison of different source images for keystroke recovery.

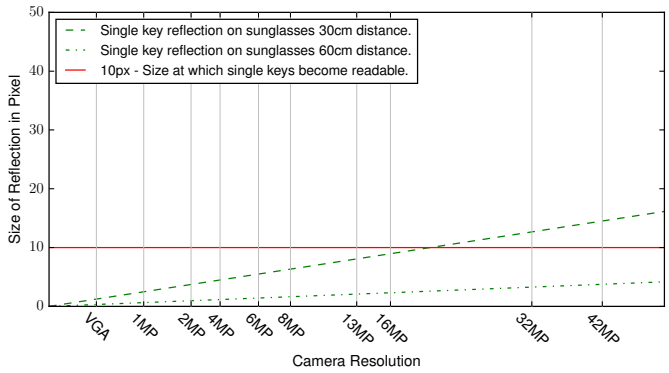
Canon VIXIA. A comparison between the Canon VIXIA and the OPPO N1 can be found in Table 3.1. Note, that the lower focal length of the OPPO N1 is more than compensated by the significantly larger resolution of the sensor. This demonstrates that cameras with only 2MP are already sufficient for corneal keylogging if the phone is held in not more than $30cm$ distance. Cameras of 32MP even allow for keylogging operations if the phone is held at $60cm$ distance.

Figure 3.2(b) depicts the size of a single key in the recorded image when reflections on sunglasses are used. With increasing resolutions, the distance between the device and the face can be increased as visualized in Figure 3.2(b). Assuming that a size of 10px per key in the recorded image is sufficient to distinguish different keys, we determined that sunglasses in conjunction with future higher resolutions enable an attacker to actually read the used keyboard. Cameras with a resolution between 16MP and 32MP suffice to actually read the screen content from a reflection.

Although the OPPO N1 has a 13MP camera, which can be front-facing, most devices on the market do not yet have front-facing cameras with a resolution that high. Following the current market developments for smartphone cameras, we can make a prediction on when devices with sufficiently large cameras will become available. As depicted in Figure 3.3(a), the most common resolutions are around 2MP. With the development for front-facing cameras being roughly six years behind, front-facing cameras with 16MP and more can be expected for 2018-2020.




(a) Reflection of a 5.1" device for $30 - 60cm$ distance in an eye



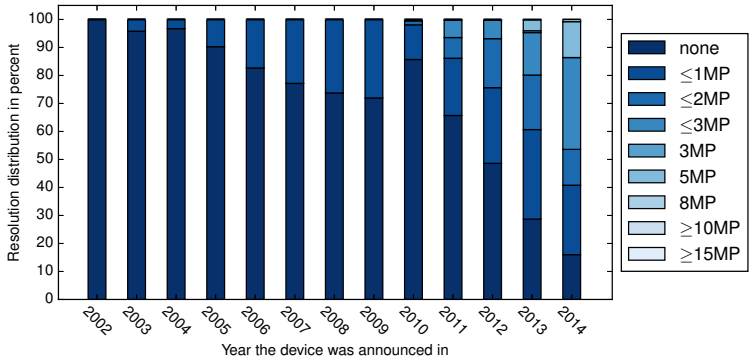

(b) Reflection of single key for $30 - 60cm$ distance on sunglasses

Figure 3.2: Size of an objects in the recorded image based on the distance between the phone and the face as well as the utilized reflective surface.

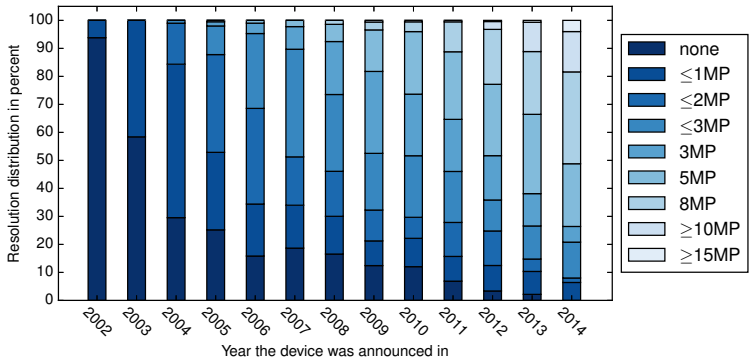
### 3.3.2 Experimental Verification

We conducted a set of experiments with the OPPO N1 and its 13MP camera (as described in Table 3.1) to demonstrate that a manual extraction is as feasible as the automated process already documented in the literature.

For this experiments, we created an application which provides a user with a numerical password input field. On each keypress, the application records an image with the front camera in the background. These images were stored on the phone for later extraction. A second subject was then tasked with determining the entered pins based on the provided images, using only the reflections found in the user's eyes. The results of this process can be found in Table 3.2. While the correct pincode was easily established for two out of four cases, the two other cases demonstrate that a permutation over the most probable as well as second most probable input may be necessary to establish the correct pin.




(a) Front camera resolution development




(b) Rear-facing camera resolution development

Figure 3.3: Development of the resolutions for front- (a) and rear-facing (b) cameras in announced devices between 2002 and 2014. The data has been obtained from gsmarena.com end of Feburary 2014.

### 3.3.3 Previous Mitigation Techniques

In the context of facial reflection shoulder surfing, some mitigation strategies were suggested by Xu et al. (Y. Xu et al., 2013). However, all three techniques proposed by them do not sufficiently prevent the attacks we just described. The first technique they propose is a

| Entered PIN | First Guess | Second Guess |
|---|---|---|
| 78135 | 48435 | 78128 |
| 90134 | 60121 | 99254 |
| 5102 | 5102 | / |
| 159397 | 159397 | / |

Table 3.2: Results of the manual pin-code recovery.

privacy screen limiting the view-port of a device, the second one are gaze-based passwords and the third are randomized keyboards. The privacy screen does not provide additional protection against our method, as the reflections are created in the direct view port, i.e. the face of the user. Gaze based passwords would be entered via eye-tracking with the front camera, the sensor which we already utilize for our attack.

Finally, randomized keyboards do provide some protection, but only as long as the recorded image does not have a resolution high enough to actually read the random keyboard. However, according to our analysis in the previous sub-section, even slight increases in the resolution of user-facing cameras in conjunction with worn sunglasses will provide sufficient images. Based on the data presented in Figure 3.3(a)(a) and (b), such devices can be expected in the near future. For lower resolution cameras, it might be feasible to measure the time a user needs to press a key on the randomized keyboard. If that time is closer to the subjects native typing speed, it may be an indication that the letter for that key is on its native keyboard position. Over time, it would hence be possible to extract a password, each character when the corresponding key is in its native position.

## 3.4 Rear-Facing Camera: Fingerprint Extraction

In this section we introduce a new method to extract a user's fingerprints and demonstrate how these extractions can be used to create forgeries sufficient to break the most recent mobile fingerprint readers. During these experiments we used the OPPO N1 as introduced in Section 3.3. As we exploit the camera of a mobile device for this, the whole process requires no physical contact to the victim. Our successful creation of forgeries demonstrates that cameras in smartphones are a threat to biometric authentication mechanisms.

### 3.4.1 Fingerprint Extraction

The first to effectively clone a fingerprint usable on a fingerprint sensor was Matsumoto in 2002 (Matsumoto et al., 2002), although the forensic literature holds indications of forgeries being conducted well before that (Geller et al., 1999). Matsumoto used gummy and rubber replicas to create forgeries from either mold-prints of real fingers or laser printed negatives of scanned fingerprints produced by pressing an inked finger on a sheet of paper. We discovered that the resolution of modern smartphone cameras suffices to obtain images of fingers that can be used for the same process. In an attack scenario an adversary has to obtain an image of the main-hand (either left or right) index finger, the one mostly used for biometric authentication. We achieved the best results when the phone was put down with the front side facing the table. During the subsequent pick-up by the user, ideal images of the main hand index finger may be created. This process is depicted in Figure 3.4.

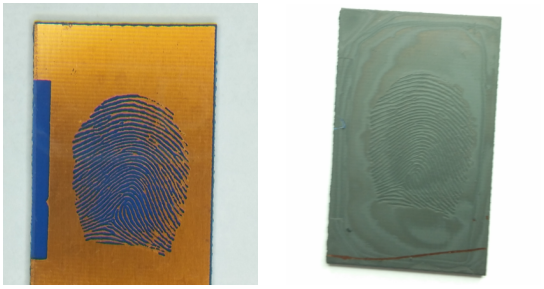
Figure 3.4: While a user picks up a device the right index finger moves through the rear-facing camera view-port (red). An attacker can use this moment to create an image of the user's fingerprints.

### 3.4.2 Cloning Process

The images taken with the technique described above are the starting point for cloning. As depicted in Figure 3.6(a) and 3.6(b) this process consists of first identifying the section of the image which contains the fingerprint to extract. Then the image is manipulated by a binary filter, which transforms the darker valleys to black and the lighter ridges to white. Furthermore all peripheral parts of the image are cropped and replaced by a black area surrounding the print. As this does not necessarily yield perfect results, manual intervention can be required to adjust fine-grained parts of the print.

In contrast to direct scans of latent prints, the real size of an object taken by a camera is not known. It depends on the zoom and distance between finger and camera. To get an estimation of the finger's size, additional information like zoom level or the auto-focus settings could be used. As biometric features change over time and even between single measurements subtle changes occur, most systems allow some tolerance. The fingerprint system from Digital Persona we used for testing tolerated a size variation of $+ - 10\%$ during our empirical evaluation.

To create a mold for the dummies, we first enhanced the pictures in contrast and brightness until the ridges and valleys are distinguishable for the binarisation step. By splitting the values of the brightness channel valleys turn black and ridges white as depicted in Figure 3.6(a) and 3.6(b). Depending on the quality of the image some manual post-processing has to be



(a) Etched PCB negative (b) Graphite applied

(c) Wood-glue applied (d) Ready forgery

Figure 3.5: The four stages of creating a fingerprint forgery.

performed. As a picture of the finger is taken, the resulting images must be mirrored before they can be printed onto a transparent foil using a laser printer. Modern sensors have a resolution of at least 500dpi, but up to 1,000dpi (Jain et al., 2007), so the print-out should have at least this resolution. The toner particles form a three dimensional structure with a height of around 15 microns, which is sufficient to fool most types of sensor.

Thermal sensors use the different cooling-time between the air in the valleys and the skin of the ridges to create an image of a fingerprint. Modern capacitive sensors emit an additional RF field into the finger, which allows them to measure deeper skin layers. To account for these features the dummies for such sensors have to be created with deeper molding structures. To create these we used the print-out as an etching-mask on a photo sensitive printed circuit board (PCB). These boards come with a copper layer of 35 or 70 microns which is approximately the height of the ridges in a human fingerprint. Subsequently, the PCB is etched to remove the undesired areas. Finally, the resulting dummy can be used to create replicas by applying a thin layer of common wood-glue on it. To increase the capacity of the replica and for easier removal of the glue the PCB is covered with graphite spray before the wood glue is applied. After the thin layer of half a millimeter wood glue is set, the replica can be carefully peeled off the PCB. The whole process is depicted in Figure 3.5. First a negative is etched from a PCB (Figure 3.5(a)). Subsequently graphite spray is applied to allow for easier peeling of the forgery, and to adjust the capacity of the wood-glue, as seen in Figure 3.5(b). The applied wood glue on the negative has to set,

this usually takes around one hour (Figure 3.5(c)). Finally, as shown in Figure 3.5(d), the created forgery can be used on the designated target.

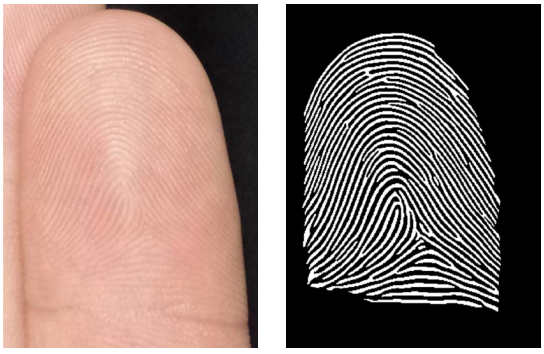
### 3.4.3 Evaluation

The created forgeries have been successfully tested on all recent flagship mobile phone finger print sensors, which were trained with corresponding original fingerprints. In addition to that we also successfully evaluated them on a legacy sensor from 2004. The forgeries suffice to fool the sensors and it can also be assumed reasonable that the recorded images can be used to track users. This could be done with a small piece of malware that performs the steps taken to extract the fingerprints automatically. By applying one of the well known fingerprint recognition algorithms, for example (Jiang and Yau, 2000), on that extract a user can be reliably identified and tracked

## 3.5 Discussion

With the method we presented an attacker can use reflections in the user's face to perform keylogging with a smartphone's front camera. We could successfully demonstrate that the accuracy for this technique outperforms previous methods. An additional feature of our technique is that, in contrast to the work of Xu et al. (Y. Xu et al., 2013), it can be utilized remotely.

By evaluating the mitigation strategies proposed in that work we could furthermore demonstrate that and why they are not effective in the context of our method. Only a fully randomized keyboard provides some security. However, this approach will either fall to statistical analysis, or the constantly increasing resolution of user facing cameras. The experiments



(a) Picture from camera (b) Extracted negative

Figure 3.6: Using binary-imaging techniques an image extract as in Figure 3.4 can be transformed to a negative to be used during fingerprint forgery.

we conducted to verify our technique furthermore demonstrate, that it is not necessary to implement a complex analysis framework in a real-world case, as the input extraction can be as easily conducted manually.

Furthermore we have created a method that allows an attacker to extract its victims fingerprints with a normal smartphone's camera in a quality high enough to create usable forgeries. An attacker can use these forgeries to circumvent stationary access controls for secured areas or use them to plant false evidence at a crime scene. With our research we could also demonstrate that it is possible to use these forgeries to circumvent the most advanced sensors for mobile phones recently introduced to the market (Simonite, 2014).

This provides an important opportunity during targeted attacks in contrast to traditional fingerprint extraction methods (CCC, 2013). Possible attackers do not have to rely on prints being present on an obtained phone. Hence they may have the opportunity to circumvent local fingerprint authentication on a phone to steal confidential data before the victim can issue a remote wipe. Additionally, if the fingerprint itself is used in a biometric remote authentication scheme as first proposed by Boyen et al. (Boyen et al., 2005), such a system is effectively broken by the introduced method, as the cryptographic secrets in that case are bound to the extracted fingerprint.

Finally, a wide range of privacy violations follows from these attacks. Authors of malicious software may use the fingerprints of a user to track the user across multiple devices or distinguish multiple users of one device. These techniques are also relevant, if an attacker has to reliably establish the identity of a user to make sure that the right device has been compromised, for example in case of high profile target.

## Mitigation

The presented attack vectors create severe challenges for a users' security and privacy. As already discussed in the introduction of this paper, permission systems do not provide sufficient protection, as long as a user can grant those permissions to applications asking for them. Therefore we will focus on mitigation strategies that do not require additional decisions from the user.

The most convenient technique imaginable is a dedicated hardware lid, which physically disables the camera. This can either happen with a shutter or by separating the power connection. To enhance the security of this technique, a dedicated sensor in the trusted computing environment indicating the state of the disable button could be implemented, that can be checked by applications. Hence an application could refuse logins, if the camera is not effectively disabled. In fact, such a sensor, disabling all non essential sensory inputs on a device would also effectively mitigate any other sensor side-channel based keylogger method.

The extraction of fingerprints can not be mitigated as easily. While such a button might help in restricting the amount of situations in which a fingerprint can be extracted, it does not

prevent all of these situations. Especially if a user forgets to close the lid before putting the phone down. This leaves two possibilities. For high security phones removing the cameras all together is certainly an option, and due to the uncovered issues advisable. For normal end-user, however, it is not. To mitigate the presented attacks on those systems an in-camera algorithm that reduces the resolution of parts of an image that have been identified as fingerprints, before the image leaves the sensor may be applicable. Similar to the lid the power-supply for the camera can be coupled to the one of the screen. Hence, if the screen is off, as the phone has been put down, the camera is necessarily off. Other methods include using biometric features that are invisible in normal light, for example deep vein patterns in the human finger.

## 3.6 Summary

In this chapter we demonstrate how an attacker can exploit the camera of a user's smartphone to obtain sensitive information. We established that the cameras found in modern smartphones constitute a serious security threat. Not only could the camera be abused for a keylogger, it could also be the extraction point for the user's fingerprints. Furthermore, we demonstrate how attackers can use these weaknesses to steal sensitive information and penetrate high security environments.

To mitigate such attacks, we investigated multiple mitigation strategies, which would prevent attackers from exploiting the camera of a device for keylogging and severely hampered attempts to extract fingerprint information if they were widely adopted. However, given the small corner case of users that might be targeted by our attack, it is unlikely that these mitigations will be adopted in mainstream devices. This leads to the conclusion that phones used in high security environments should avoid having cameras.

Stepping back, this chapter highlights what a dedicated attacker can archive by abusing a smartphone's camera. However, the kind of attack that we present is non trivial and requires significant work from the attacker. Thus, it is unlikely to be executed at large scale.

# 4

# Protocol Definitions as Misconfiguration Facilitating Factors

As pointed out Chapter 2, security incidents in database systems due to operational malpractice are common events, e.g., (Binary Edge, 2015a; Ghena et al., 2014; Mathews, 2016; MongoDB website, 2015). This may lead to the assumption that a common factor in database related services makes them especially susceptible to misconfigurations. Thus, in this chapter, we investigate if the way *protocols* they use were designed may facilitate misconfigurations.

Today's Internet utilizes a multitude of different protocols. While some of these protocols were first implemented and used and later documented, others were first specified and then implemented. Regardless of how protocols came to be, their definitions can contain traps that lead to insecure implementations or deployments. Insufficiently strict authentication requirements in a protocol specification are a prime example of this.

The resulting misconfigurations, i.e., not enabling strong authentication, are common root causes for Internet security incidents. Indeed, Internet protocols have been commonly designed without security in mind which leads to a multitude of misconfiguration traps. While this is slowly changing, to strict security considerations can have a similarly bad effect. Due to complex implementations and insufficient documentation, security features may remain unused, leaving deployments vulnerable.

Hence, this chapter attempts to systematically group and classify network protocols to identify common traits that make them especially susceptible to misconfiguration. The obtained insights together with observations about end-user centric usability and security by default are then used to derive recommendations for improving existing and designing new protocols—without such security sensitive traps for operators, implementors and users.

## 4.1 Motivation

Security incidents involving Internet services have become regular events. Examples include: (a) disclosures of information, e.g., petabytes of personal data stored in unprotected key-value stores - NoSQL databases (Binary Edge, 2015a; MongoDB website, 2015). (b)

Unauthorized access via the Internet to systems, e.g., supervisory control and data acquisition systems (SCADA) (Meixell and Forner, 2013) which are used to control systems which range from light installations to oil platforms. (c) Undesired publication of information, e.g., health data from the United Kingdom on an HTTP root (Solon, 2014), or an UEFI signing key on an internal FTP server which was available via the Internet (Caudill, 2013).

While programming vulnerabilities are well-studied, e.g., see (McGraw, 2004, 2006), insufficient attention has been paid to poorly designed and hard to configure protocols. We focus on protocol security practices with their varying naïvete, complexity, and weaknesses. Following our definitions from Chapter 1, we indeed find misconfigurations among the root causes of the above severe incidents. While this is, in principle, well known, e.g., (Cuppens et al., 2005; T. Xu, J. Zhang, et al., 2013), to date, the main and only explanation has been human error. We claim that there are more fundamental reasons for such misconfigurations which stem from the design of the Internet's protocols themselves, the security assumptions or configuration choices offered by protocols. These lead to services which are prone to misconfiguration.

Our study complements the multitude of individual incidents documented in the scientific literature and the anecdotes in systems lore with a macroscopic systematic survey of Internet protocols and their corresponding services. More precisely, we investigate which of the underlying assumptions during protocol design lead to misconfigurations during service deployment. We refer to this as misconfiguration prone protocols/services.

There are many reasons for misconfigurations: (a) the operator does not follow best practices regarding network settings by which the service is deployed, (b) the operator does not use the default configuration settings leading to tainted CIA, or (c) the operator uses the default configuration settings and they lead to tainted CIA.

Given the newest Internet trends, services in the cloud, Internet of Things including Industry 4.0, autonomous systems, e.g., self-driving cars, and mobile applications, we expect even more diversity and complexity and, thus, more security incidents. Considering the link between mismanagement and maliciousness (Liu et al., 2015; J. Zhang, Durumeric, et al., 2014), many of these incidents will involve misconfigurations. Thus, we claim that a systematic review of why misconfiguration occurs is needed.

In summary, in this chapter we review the assumptions under which protocols have been designed along two dimensions, (a) the assumed strength of the attacker - weak vs. strong - and (b) the defense paradigm - good enough vs. perfect security. By using these dimensions to group protocols we find four major clusters, one in each quadrant. We name these clusters ***Early Internet*** for weak attacker/good enough, ***Emerging Threats*** for weak attacker/perfect security, ***Complex Security*** for strong attacker/perfect security, and ***A new Simplicity*** for strong attacker/good enough. The names capture the design essentials as well as the mindset of the protocols in each of the classes. Furthermore, the names express how the security mindset of protocol design evolved over time. From this systematization

of misconfiguration prone protocols we derive a set of specific action items. To keep protocols secure and misconfiguration resilient these must be consider when introducing new or updating old protocol specifications.

## 4.2 Systematization Method

Thousands of Internet protocols have been proposed, developed, and deployed on the Internet; therefore an exhaustive analysis is beyond the scope of this thesis. Rather we extract essential features from a subset of misconfiguration prone protocols and use these to derive our systematization.

### 4.2.1 Example Protocol Selection

Our choice of protocols is driven by the following considerations. First, we choose protocols that are commonly used and/or are new and upcoming. Next, we choose some were misconfigurations can potentially have a large impact or those where system lore states that they are easily misconfigured. We augment this list by protocols that capture corner cases. From these we selected a set of protocols that are most iconic for the relevant class. Due to the size limitations of the work at hand, only a subset can be introduced in detail and displayed in Table 4.2. We focus on the server, not client side. Thus, pure end-user focused protocols as well as client misconfigurations are beyond the scope of this thesis.

### 4.2.2 Security Relevant Misconfigurations

If a service is deployed in such a way that its CIA is tainted because of a misconfiguration we call the misconfiguration security relevant. There are three main reasons for such misconfigurations. The first is when the operator of the service does not follow best common practice (BCP). The second is when the operator uses their own configuration which taints the CIA of the service. The third is when the operator uses the default configuration, but the CIA is still tainted, likely due to incorrect defaults.

We refer to misconfiguration prone protocols and services. But in the end it is the service that is misconfigured, often facilitated by design choices in the protocol. Examples for common misconfigurations are that the service is deployed in a network setting which deviates from the one for which it, or its default configuration, was designed.

### 4.2.3 Security Guidelines for Protocol Design

Request For Comments (RFCs) document Internet protocols and services. Since 1992, each RFC must address the topic of security, according to, IETF processes, (RFC1311 (Postel, 1992), RFC1543 (Postel, 1993), RFC2223 (Postel and Reynolds, 1997), RFC7322 (Flanagan and Ginoza, 2014)), which document what an RFC must contain. Indeed, RFC1311 (Postel, 1992) states: *"All STD RFCs must contain a section that discusses the security considerations of the procedures that are the main topic of the RFC."*

Over time, the community has realized that this statement by itself is not sufficient and the specification of the security requirements have gotten stricter, see RFC3552 (Rescorla and Korver, 2003) from 2003, the Best Current Practices for *"Writing RFC Text on Security Considerations"*. The goal of this requirement is to make all protocol designers and implementers aware of possible security implications. Given that this basic requirement existed in 1992, we conclude that the importance of security has been recognized for at least 23 years.

### 4.2.4 Review of Security Threats for Protocols and Services

One of the motivations for including a security section in each RFC is to make the protocol designer consider the following two questions: (a) against whom to defend and (b) how to defend.

We find that protocol designers consider different kinds of attackers, ranging from very weak to very strong. The weak attacker is either unskilled or is resource limited. The strong attacker is very skilled and has all necessary resources in their hands.

Defining how to defend is more difficult as it depends on the eyes of the beholder. Some argue that the cost of breaking security should be larger than the value of the protected asset. This goes back to Pfleeger and Pfleeger (C. P. Pfleeger and S. L. Pfleeger, 2002) and specifies that one should put up a wall against threats at least high enough that most attacks will not break the wall. Moreover, the wall should not cost more than the protected asset. We call this the "good enough" approach to security. Others, especially the field of cryptography (Krämer, 2015), follow the approach of "perfect security". Perfect security refers to using every possible mean to achieve security. We refer to these two approaches as the defense paradigm.

### 4.2.5 Classification of Protocols and Services

Thus, we have two-dimensions, namely, the capabilities of the attacker and the defense paradigm. Using these dimensions we classify our selected set of protocols and identify four major clusters. These clusters correspond to the four quadrants of the two dimensional space. We refer to them as: ***Early Internet***, ***Emerging Threats***, ***Complex Security***, ***A new Simplicity***.

**Weak attacker - good enough**
This class contains those protocols that are designed for a friendly, collaborative environment - the Internet when security was not yet a major concern. This class, in particular, contains those protocols that initially were designed under the assumptions that there is no attacker, or still carry artifacts from that idyllic time. Such an attacker is the weakest one possible. We refer to this class as ***Early Internet***.

**Weak attacker - perfect security**
This class no longer assumes that the environment is entirely friendly. Rather it recognizes that there are threats, but not yet by sophisticated attackers. However, since significant assets can be at stake, even attacks that are only theoretically possible are considered. We refer to this class as ***Emerging Threats***.

**Strong attacker - perfect security**
This class captures the protocols that consider security a necessity at all costs. As a result, the protocols in this class are designed to handle strong attackers and be safe against all, even theoretically conceived, attack vectors. However, as a result, they are often complex and hard to deploy, maintain, and difficult to use. We refer to this class as ***Complex Security***.

**Strong attacker - good enough**
This class contains those protocols who's designers recognize that strong attackers exist but also value protocols that "just work" out of the box. Therefore, the designer does not try to defend the asset against every possible attack by reducing the attack surface. In this class we see a conscious choice between security and operational ease, favoring the latter. We refer to this class as ***A new Simplicity***.

Interestingly, when one considers when most protocols in each of the above classes were designed, we find that Internet protocol designers have started with protocols in the ***Early Internet*** category, and moved to ones from ***Emerging Threats*** when they realized that the Internet was no longer nice. However, while the core ideas did not change, the protocols were hidden behind fences such as DMZs. Since this did not suffice, they moved to ***Complex Security***. As these were hard to maintain or difficult to use, we see a new trend towards ***A new Simplicity***. Please see Figure 4.1 for a visualization of where the example protocols we discuss in the remainder of this chapter can be found in that 2-dimensional plane. An overview of these examples can be found in Table 4.2.

### 4.2.6 Systematization

In Sections 4.3–4.6 we take a closer look at each of the above classes. For each class, we identify a set of representative protocols which we analyze according to five sets of features.

**Security Features:** Among the essential security features that protocols should support are authentication, authorization, and use of encryption (TLS for transport layer encryption).

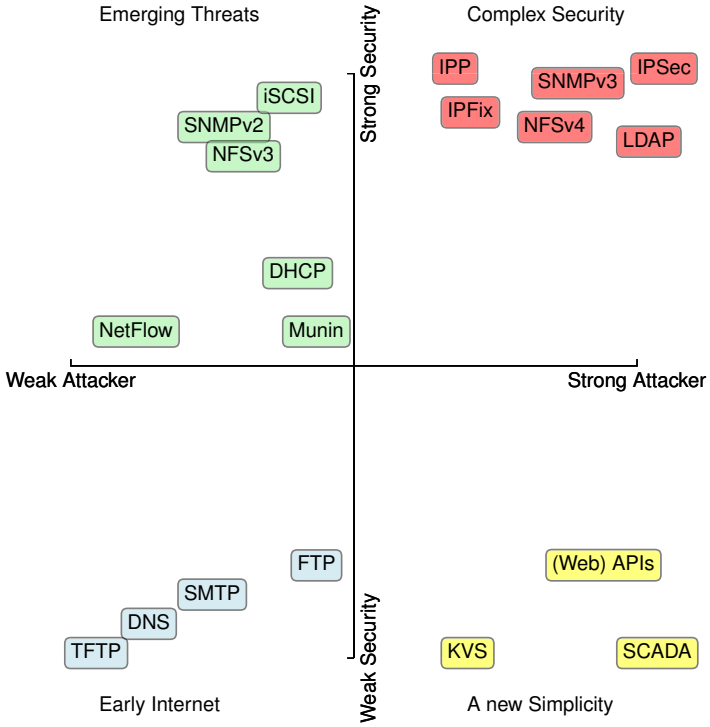



Figure 4.1: Example protocols used in this chapter, arranged on the protocol design plane following our main-classification.

We mark for each protocol/service which of these features is (a) in common use (●), (b) implemented but not commonly used (○), (c) not implemented (-). If the protocol does not support a security feature we leave the space blank.

**Misconfiguration traps:** We consider the possible misconfiguration traps which a protocol/service may have. Under **NoAuth** we capture if (a) authentication is not offered by the protocol, (b) typically not implemented, or (c) typically not configured in the deployed service. Under **Credentials** we note if the service is commonly deployed with weak default credentials. Under **Artifacts** we note if protocol features common at design time lead to misconfigurations when used today. With **Fencing** we note those cases that depend on firewalling, etc. to ensure that they are not reachable from the Internet. We mark those protocols **NoUse** that are hardly deployed even though they replace earlier protocols, e.g., prior versions, with major misconfiguration traps.

**Support:** As misconfigurations often occur due to poor technical support for operators, we look closely at that for each of our representative protocols. More precisely, we look at the documentation for securely deploying the service and check if it is mostly (○) or always (●) misleading, lacking, too complex, or otherwise insufficient. Another common aspect that leads to misconfigurations are bad defaults. But rather than looking at what can go wrong we check if the defaults are always (●) or sometimes not "sane" (○). Sane in the sense that

they enable or enforce enabling the supported security features of the protocol/service to ensure confidentially, integrity, and availability by default.

**Publications:** Here we capture if problems that can lead to misconfigurations discussed under misconfiguration traps or support are already well known either in the academic world or the security community. If it is known in the academic world, we refer to a representative paper. If it is known in the community we note approximately when it became part of the system engineering lore.

**Visible Instances:** Next we try to estimate the number of systems, or rather IP addresses, that offer the service/protocol under discussion. We rely on different data sources, among them (a) publicly available data sets or representative papers, (b) zMap (Durumeric et al., 2013) scans by the authors, and (c) search results from Shodan (Bodenheim et al., 2014; Shodan, 2015). Shodan is "the world's first search engine for Internet-connected devices". While Shodan does provide an estimate of the possible number of IPs offering a service it is neither complete, covers all available ports (Bodenheim et al., 2014), nor are all instances per se vulnerable.

## 4.3 The Early Internet

This class includes those protocols that were designed in the context of the early Internet, roughly from 1960 – 1988, where attacks had yet to be considered and therefore protocols were designed without security considerations. The paper by David Clark about "The Design Philosophy of the DARPA Internet Protocols" (Clark, 1988) does not even contain the term security even though availability in the sense of survivability is a major goal. After all, the main goal of the Internet was to interconnect existing networks with the implicit assumption that all participants worked towards the common goal of communication.

As a result attacks were not yet common. So, the need for security either did not exist or was extremely limited. Towards the end of the era, attacks against operating systems became more prominent and the first major Internet worm, namely the Morris worm, was let loose (Orman, 2003; Spafford, 1989).

Most popular protocols from that era have been updated to remain usable in today's hostile Internet. However, this does not remove all misconfiguration traps. Today many still have artifacts of their design for a friendly Internet.

Thus, the threat model of this class is: "weak attacker" with "good enough". The representative protocols we examine are: SMTP and DNS. Other protocols in this class include: TFTP, FTP, Finger, rexec, Chargen, NIS, RIP, NTP, WHOIS, Ident, XDMCP/X11, Syslog, rsync and IRC.

### 4.3.1 Example Protocols

**FTP:** The file transfer protocol (FTP) is an application layer protocol for Internet file transfer between hosts. FTP is one of the earliest Internet protocols and was first documented by RFC 114 (Bhushan, 1971) in 1971. It provides authentication within the protocol and authorization via the operating system's file system access controls. It does not offer encryption. However, it can be used over a TLS tunnel, see RFC4217 (Ford-Hutchinson, 2005) from 2005.

The major misconfiguration pitfalls for FTP servers are related to either missing authentication or insufficient authorization and directory limits. These are: (a) Enable anonymous logins to share files publicly, see RFC1635 (Deutsch et al., 1994). Hence, files uploaded to a server that allows anonymous access become public. Recent examples show that, e.g., private keys (Caudill, 2013) can be exposed this way. (b) If, in addition, write access is enabled files can be deleted and/or overwritten. The FTP servers can also be abused to share malicious content. This issue has been discussed as common example by Uppuluri and Skar (Uppuluri and Sekar, 2001). (c) Faulty configuration of the root-directory of an FTP server may expose system files. If, e.g., a UNIX machine exposes its global root directory, FTP users can access all files that are accessible to the user operating the FTP server. (d) FTP's dedicated data-channel enables an attacker to send files containing service commands to a remote server, e.g., SMTP. This attack can be used for amplification and firewall evasion (Helmer et al., 2002). RFC2577 (Allman and Ostermann, 1999) recommends disallowing data-channel connections to low-ports as mitigation.

To counteract some of the above threats, some modern FTP implementations, such as vsFTPd, ship a systematically locked down default configuration. It requires extensive user action to enable anonymous, write, and, non-directory-restricted access. The documentation of vsFTPd is short and precise. Other widespread FTP implementations, e.g., the versatile solution ProFTPd have a more complex documentation due to their larger feature set.

**TFTP:** The Trivial File Transfer Protocol (TFTP) is a "very simple protocol used to transfer files" and is documented in RFC783 (Sollins, 1981) dated 1981. TFTP only supports reading and writing files and lacks most of the advanced features of FTP. It uses UDP as transport layer protocol. TFTP is often used for bootstrapping by providing access to files needed for system boot such as boot images, firmware updates, or network device configurations files. Revision-2, RFC1350 (Sollins, 1992), fixed the "Sorcerer's Apprentice" protocol bug - a major data retransmission problem which leads to packet amplification.

TFTP itself does not provide authentication or authorization. It does offer limited protection by means of the operating system's file system access controls. Modifying files on an TFTP server can be restricted by most implementations. TFTP does not support encryption. TFTP is by design insecure which has also led to its most common use case, to allow bootstrapping of unprovisioned systems that therefore have no security credentials.

TFTP is also among the first protocols to suffer from unintended amplification attacks, see RFC1350 (Sollins, 1992). This is one of the earliest amplification attacks of stateless

protocols. TFTP servers, unless shielded from general access, are subject to disclosure attacks (Handley et al., 2006). Indeed, they often enable transitive attacks, in which an attacker first retrieves the confidential configuration files, including encryption, authentication, and authorization secrets. Then the attacker uses this information to access the systems newly configured over TFTP.

If the TFTP server allows write access, attackers can, in principle, overwrite any of the configuration files with their own, resetting passwords or configuring additional known credentials for super users. Similarly, the attacker can alter the available system images - boot as well as full system images - to include backdoors. Once provisioned they enable the attacker to take over the corresponding systems.

The documentation of the most common TFTP server implementations is short and precise and includes a sensible discussion of the security issues. Moreover, the suggested configurations are appropriate.

To counteract most of the above threats, almost all TFTP servers must be isolated from public access and subject to strict access rules or firewalls. Thus, the main misconfiguration trap is missing fences. Indeed, in 2014, MacFarlane et al. (Macfarlane and Buchanan, 2015) found more than 600,000 publicly accessible TFTP servers. However, fencing is insufficient against inside attackers.

**DNS:** Since the early 1980's the Domain Name System (DNS), RFC882 and RFC883 (Mockapetris, 1983a,b), is used to map hostnames to IP addresses and vice versa.

DNS is a hierarchical distributed database organized in independently administered DNS zones. These zones are implemented as subtrees in the hierarchy of the DNS. Each zone has at least one "authoritative" DNS server while one server can be authoritative for multiple zones. In addition, a nameserver can also be queried by client hosts and provide name resolution for arbitrary domains for which it is not necessarily authoritative. Most non-authoritative DNS servers only serve clients within their administrative domain, e.g., an Internet Service Provider (ISP) providing name resolution for its customers. However, services like OpenDNS and the Google public DNS provide public available name servers.

While some services are intentionally open to the public there is a large mass of misconfigured servers unintentionally providing public name resolution. DNS by default uses connectionless UDP and usually the responses are larger than the queries as the query is contained in the response. Thus, DNS, by design, can be abused for amplification attacks, in particular both with open resolvers (Rossow, 2014) and resolvers that return large answers.

Mitigation strategies against amplification attacks exist and are usually deployed by the large providers of open DNS servers. Nevertheless, this kind of abuse can not be completely prevented due to inherent protocol limitations (Rossow, 2014). While DNSSEC is currently being discussed as mitigation for various other problems in the DNS protocol suite it exacerbates this abuse as it often produces very large answers (Rijswijk-Deij et al., 2014).

Another attack vector is information disclosure in reverse DNS lookups. Using these an attacker may infer which hosts offer which services, even inside a firewalled network, or disclose the organizational structure. Some misconfigured systems may still allow zone transfers of full DNS Zones (Kalafut et al., 2008), which was historically the default (Bernstein, nodate).

Currently, we find more than 10,000,000 DNS servers on the Internet. A substantial fraction, more than 5,000 (Streibelt et al., 2013), of these are open DNS servers of which it is unclear to what extent they deploy even the available limited amplification mitigation.

**SMTP:** The objective of the Simple Mail Transfer Protocol (SMTP) as documented in 1982 in RFC821 (Postel, 1982) and updated by RFC5231 (Segmuller and Leiba, 2008) is to reliably and efficiently transfer email. Among the important features of SMTP is the ability to relay email across multiple networks.

The base architecture of SMTP used open relays for forwarding messages between Mail Transfer Agents (MTAs) without authentication or authorization, see RFC822 (Crocker, 1982). Authentication was suggested by an Internet draft in 1995 and finally added with RFC2554 (Myers, 1999) in 1999. TLS was also added in 1999 with RFC2487 (Hoffman, 1999).

Attackers realized early on, that open relays are great for amplifying the effects of worms, viruses, and in particular SPAM (Klensin, Freed, et al., 1995). Even with the CIA features SPAM is a major daily annoyance.

These problems are usually mitigated by providing strict and well documented default configurations (Jung and Sit, 2004). In today's deployments almost all SMTP servers only accept emails for their configured domains. Thus, the possibility of amplification has been reduced. If MTA to MTA relay is allowed it is only with credentials and TLS.

Another problem with SMTP is that an attacker may take over an SMTP server and send rogue data which is not easily mitigated. The defense here is blacklisting, whitelisting, sender verification, etc. see, e.g., (Cormack, 2007).

However, it is still possible - in an attempt to "Make Things Work" - to misconfigure SMTP servers. After all, problematic configurations do still have applicable use-cases on the Internet, e.g., an outbound email relay for a large network which every machine should use. In the wild, open SMTP relays are still observed from time to time. But most are quickly found and closed down.

### 4.3.2 Discussion

A common misconfiguration trap in this class is the assumption that neither client authentication nor encryption is needed as the services are in a trustworthy environment. Indeed, access without authentication is considered a feature (FTP, SMTP, and DNS). Other abuses

of misconfiguration are stateless amplifications attacks (TFTP, DNS) where the root cause is that the server sends data without checking if the client wants it.

Based on these observations one would presume that protocols in this class have seen the end of their live cycle. However, almost all of the above protocols are still very popular. The reason is (a) the Internet relies on the services (DNS, SMTP) (b) their convenience (FTP), (c) the fact that there is no good alternative (TFTP), and (d) the service happens to be running and is a legacy service. Worse, the implementation of, e.g., TFTP requires a small code base which makes it common in millions of customer premise equipment (CPE). Indeed, these protocols are unlikely to disappear as they are the foundation of the Internet.

The reason why such services are still in operation is twofold: Either it is presumed possible to hide the services behind firewalls (TFTP, NIS, RIP) or work on alternative protocols has started, but these protocols are not yet in Internet-wide use (DNS, SMTP). But, misconfigurations occur if either the firewall fails or the protocols are not used as originally designed.

The first major security incident which exploited a misconfiguration was an email amplification attack - namely the Morris worm in 1988 (Spafford, 1989). At about the same time, the community started to realize that major misconfigurations can occur, see RFC1222 (Braun and Rekhter, 1991) and RFC1223 (Halpern, 1991) that "provide guidance for vendors, implementors, and users of Internet communication software".

## 4.4 Emerging Threats

Security incidents such as the Morris worm changed the way that Internet protocols are perceived. Instead of designing them for the Internet at large, they became explicitly designed with firewalls in mind. Thus, network firewalls, more precisely packet filters (Chapman, 1992), became the typical way of fencing off network services.

Bellovin and Cheswick (Bellovin and Cheswick, 1994) state that the motivation for Network Firewalls is: *"Computer security is a hard problem. Security on networked computers is much harder. Firewalls (barriers between two networks), when used properly, can provide a significant increase in computer security."* In addition, the approach from 1988 onwards is, according to Bellovin and Cheswick (Bellovin and Cheswick, 1994): *"Everything is guilty until proven innocent. Thus, we configure our firewalls to reject everything, unless we have explicitly made the choice - and accepted the risk - to permit it"*.

Protocol designers find network firewalls to be a convenient way to handle security. In their minds, firewalls enable them to basically ignore security threats as they presume that the firewall rejects everything that is "untrusted". The design assumption of most protocols is that since the attacker is not strong enough to get past the firewall the protocol itself can be designed for a trusted environment.

However, as stated by Wool (Wool, 2004): *"The protection that firewalls provide is only as good as the policy they are configured to implement. Analysis of real configuration data shows that corporate firewalls are often enforcing rule sets that violate well established security guidelines."* His conclusion is to keep firewall configurations simple but efficient to avoid misconfiguration.

Thus, the threat model for this class is: "weak attacker" with perfect security. The representative protocols we take a closer look at are: NetFlow, DHCP, and, iSCSI. Other protocols in this class include: SNMPv2, Munin, NFSv3, Wake on Lan, Remote DMA, NBD, rsyslog, SCADA, early versions of CIFS/SMB, and Mapping of Airline Traffic over Internet Protocol (MATIP).

### 4.4.1 Example Protocols

**NetFlow:** Cisco Systems NetFlow service allows network administrators to collect IP flow information from their network. A flow is a summary of a set of packets that pass through a device that have some common property. NetFlow uses UDP as its transport protocol. NetFlow is widely used in many ISP and enterprise networks. Indeed, many resource accounting as well as security incident systems are built on this data source. Early versions are documented as Cisco white papers. Version 9 is documented in RFC3954 (Claise, 2004).

NetFlow itself does not provide authentication, authorization, or encryption support. This was a conscious choice by the protocol designers, to cite RFC3954 (Claise, 2004): *"The designers of NetFlow Version 9 did not impose any confidentiality, integrity or authentication requirements on the protocol because this reduced the efficiency of the implementation and it was believed at the time that the majority of deployments would confine the Flow Records to private networks, with the Collector(s) and Exporter(s) in close proximity."* Indeed, RFC3954 specifically redirects the issue of security to the subsequent IPFix security requirements in RFC3917 (Quittek et al., 2004).

RFC3954 outlines possible attacks including disclosure of flow information data, forgery of flow records or template records, and DoS attacks on NetFlow collectors. The latter enables an attacker to exceed the collector's storage or computational capacity and, thus, can disable the monitoring of the network. Using forgery, an attacker can inject flow information that (a) may redirect network-forensic investigations by incriminates another party or (b) lead to wrongful charges if NetFlow is the basis of accounting.

These attacks can, in principle, be mitigated by, e.g., moving to TCP and enforcing TLS/DTLS and mutual authentication. An example of such a mitigation strategy is the proposal in the IPFIX security requirements RFC3917 (Quittek et al., 2004), see Section4.5.

The documentation of NetFlow is given mainly by Cisco White papers and Cisco device configuration examples. We observe that the documentation does not even mention how to secure NetFlow or that it is necessary. It does, however, point out that NetFlow data can be used as security enhancement to investigate network anomalies.

In summary, NetFlows main misconfiguration trap is insufficient fencing. Since NetFlow is a stateless write only protocol it is unfeasible to estimate the misconfigured number of NetFlow collectors by active scans.

**DHCPv4:** The Dynamic Host Configuration Protocol (DHCP) is a stateful client-server protocol that can be used to provide configuration parameters to hosts - the clients - connected to the Internet. In practice, it is often used by clients to retrieve their IP address configuration as well as additional parameters including nameservers, domainnames, or local TFTP servers for diskless clients.

DHCP is based on the Bootstrap Protocol (BOOTP) and is documented in RFC1531 (Droms, 1993) in 1993. DHCPv6, standardized in RFC3316 (Droms, Bound, et al., 2003) in 2003, tackles many of the security issues of DHCPv4. Since DHCPv4 is still commonly used for configuring IPv4 networks we discuss it in this section. In the following when we refer to DHCP we mean DHCP for IPv4 address configuration.

Regarding security RFC1531 (Droms, 1993) claims that " *DHCP is built directly on UDP and IP which are as yet inherently insecure*". Since DHCP does not add any security features itself this means that the protocol lacks all basic security features. It has been designed without client authentication, server authentication, or encryption. Client Authentication was added in 2001 (Droms and Arbaugh, 2001), but is not widely implemented, especially in the common embedded DHCP servers for CPEs.

As a result, any attacker can exploit the possibility for a single client to request all leases held by a DHCP server. This effectively blocks all other clients from obtaining an address. This attack is critical as it can be executed locally as well as remotely if the DHCP server is accessible from the Internet.

The next problem is that servers do not have to authenticate themselves towards the clients. Therefore, any host can pretend to be the authoritative DHCP server for its network segment. This allows an attacker to impersonate a DHCP server and send malicious information to the clients, e.g., to use (a) a different gateway which is hijacked by a monkey in the middle or (b) a different DNS server to spoof internal websites and access credentials.

Among the common DHCP servers are ISC-DHCP and dnsmasq. Their documentation is reasonable but ignores the topic of security. The main misconfiguration trap is yet again insufficient fencing. Indeed, during the 28th Chaos Communication Congress in December 2011 the network operations team observed a DoS against their publicly reachable DHCP server. A virtual machine hosted in Amazon EC2 performed a lease starvation attack on that system (Rechthien and Hargrave, 2011).

Since DHCP is the common protocol for assigning dynamic IP addresses it is in common use almost everywhere. Indeed, even many home users use DHCP due to the large scale introduction of Network Address Translation (NAT) enabled home routers.

**SNMPv2:** The Simple Network Management Protocol (SNMP) dates back to 1988, when it was first specified in RFC1067 (Case, Fedor, et al., 1988). It is the standard protocol

for managing IP network devices, including routers, switches, workstations. It is typically shipped on the device. Small command extensions led to Community-based SNMPv2 (Case, McCloghrie, et al., 1996) the version supported by most vendors, e.g., Cisco and Juniper. While other versions of SNMPv2 already support extensive security features most became prominent with SNMPv3 (Case, Mundy, et al., 2002; Wijnen et al., 1999), which we discuss in Section 4.5. The only authentication mechanism of SNMPv2c is the so called community string. Moreover, SNMPv2c does not support transport layer security.

Common misconfigurations are the use of weak (default) credentials, e.g., the community string `public` for read access and string `private` for the read/write access (Moonen and BV, 2012). While SNMP enabled devices should be shielded by proper firewall configurations they often are not. Moreover, SNMP proxies which are designed to handle ACLs can easily be misconfigured as well. Note, weak credentials can lead to disclosure of information. With the `private` community it is possible to take over the device. Moreover, open SNMPv2 servers have been used for amplifications attacks (Rossow, 2014).

By default, a lot of devices come with communities pre-configured, e.g., `public` or/and `private`. Even if the operator configures their own communities they often forget to remove the pre-configured ones - leaving the door open to attackers. Moreover, with a network sniffer it is possible to extract community strings. Depending on the class of the device, the documentation differs significantly from good for high-end devices to almost none for low-end customer premise devices. This is, in particular, problematic as the customer premise devices are often directly connected to the Internet. The potential number of devices that may be subject to this class of misconfiguration is, according to Shodan scans, more than 3,800,000 devices with the `public` community string.

**Munin:** Munin (Munin, 2002) is an open source networked resource monitoring tool. It is a simple service allowing retrieval of server statistics for monitoring purposes. The monitored server listens on an open port for inbound connections. The Munin monitor polls each of the targets by connecting to the port and requesting the status data. Other similar services use the same basic schema, e.g., NCSA, Ganglia, Collectd etc. Authentication is implemented by whitelisting IP addresses or address ranges of Munin servers. Authorization and transport security are not available.

Common misconfigurations are weak firewalls together with too liberal ACLs. This allows attackers to obtain detailed information about the infrastructure as well as fine-grained usage information. This, in turn, can enable a whole range of security critical side channel attacks on the cryptography of other protocols (Y. Zhang et al., 2012). Moreover, given recent side-channel attacks that use acoustic signals from the CPU (Genkin et al., 2014), it is not unlikely that attackers can use such data to extract, e.g., secret keys from the monitored servers.

While the documentation states that ACLs have to be clearly limited to authorized hosts, we still find more than 6,000 systems in the Shodan data.

**NFSv3:** The Network File System (NFSv3) offers transparent access to remote files. It has become one of the common UNIX network file systems and has first been documented in RFC1813 (Callaghan et al., 1995) from 1995. NFSv4 or more general NFS with Kerberos support is discussed in Section 4.5.

NFSv3 offers host-based authentication on network-wide names but not per principal authentication. Moreover, NFSv3 relies on the client OS for authorization. On the wire encryption is not supported (Callaghan et al., 1995). While NFSv3 in principle supports Kerberos secret keys it does not mandate them and most deployments do not use them. System lore warns to use NFS without Kerberos, see Section 4.6, if strong security is needed, e.g., "Kerberos is key for secure data access and not NFSv4" (Troppens, 2014).

Common misconfigurations for NFSv3 involve the Access Control Lists: They can be too liberal, e.g., network wide, or incorrectly specified, e.g., wrong subtree of the file system. In either case, an attacker can mount an NFS share and read or modify arbitrary files. While the available documentation stresses the importance of ACLs, misconfigured servers can be found in the wild. These problems are widely known both in systems lore (Eisler, 1999) as well as academia (Tanenbaum and Van Steen, 2007).

**iSCSI:** The Internet Small Computer System Interface (iSCSI) is a protocol for remotely accessing block devices over the Internet using SCSI commands first documented 2004 in RFC3720 (Satran et al., 2004). Since iSCSI was designed for a hostile Internet, a dedicated RFC (Aboba et al., 2004) exists, that spells out the security requirements for iSCSI and similar network accessible block storage protocols. This RFC has been updated most recently in April 2014 by RFC7146 (Black and Koning, 2014).

These RFCs require iSCSI to include authentication and authorization but delegate encryption and integrity to IPSec. Moreover, RFC3723 (Aboba et al., 2004) acknowledges common threats for iSCSI deployments under the assumption that authentication and authorization are working, i.e., that the attacker is not able to initiate a valid connection. Moreover, the base security assumption is that there is no monkey in the middle either due to IPSec or an isolated network segment.

However, if due to a misconfiguration the iSCSI target is reachable via the Internet without authentication iSCSI becomes a severe security liability. An attacker with access to an iSCSI volume can tamper with all data thereon and can take over all machines with root file systems on those volumes.

Indeed, it is easily possible - and many large enterprise applications and howtos for setting up UNIX based targets recommend - to configure iSCSI without any authentication, neither for the whole target set nor the individual targets. Usually, this is done to cater to operational needs, especially in the cases where iSCSI volumes are used as boot devices or if a dynamic set of virtual machines has to have access to a volume. In contrast to the above common malpractice, the Payment Card Industry Data Security Standards (PCI-DSS) (Payment Card Industry, 2014) explicitly requires client authentication, in particular, it requires it for each

volume individually. Nevertheless, the common misconfiguration trap for iSCSI is missing authentication coupled with reliance on fencing.

So far, these security issues have been recognized in the industry (Dwivedi, 2005) but not necessarily in academic references. With a quick zMap scan, we find roughly 9,000 iSCSI targets reachable via the Internet of which 1,000 do not require authentication. Among these are various major organizations as well as academic institutions.

### 4.4.2 Discussion

The base assumption of all protocols/services in this class is that the local LAN is safe. Thus, a common misconception is that they can be used with "convenient" security settings. This often leads to major security incidents when the fencing mechanisms fail. This is the major misconfiguration trap for all protocols in this class.

NetFlow is a blatant example of the low security considerations within this class. Its security concept relies entirely on fencing. DHCP goes even further: first anyone can run a rogue DHCP server and second engineers think about DHCP as a link layer protocol. SNMPv2 is one of the protocols in this class that first added security but then reduced it. NFSv3 does do authorization using OS ACLs, but without authentication. This means that anyone can impersonate anyone. While iSCSI, in principle, supports authorization and authentication some deployments do not enable it as iSCSI should be restricted to the storage network and, if used as boot device, is difficult to supply the clients with secured credentials for the iSCSI volume. However, if fencing breaks down this is a major misconfiguration as large amounts of sensible data are leaked.

However, the assumption that everything can be fenced in does not necessarily hold as specifying security policies is difficult and realizing them in a firewall is rather difficult and prone to errors (Cuppens et al., 2005; Garcia-Alfaro et al., 2013; Mayer et al., 2000; Al-Shaer and Hamed, 2003; Yuan et al., 2006). Among the complications are that the designer of the security policies are not necessarily the ones that configure the firewalls and those are not necessarily the ones that deploy the network services. Moreover, updating and maintaining such rules is quite error prone. This opens up the network service for all kinds of attacks that bypass firewalls or access services that are thought to not be reachable from the Internet.

Other means of fencing include: (a) not connecting the service to the Internet at all (air gap) (b) VLANs and sub-networks (c) Virtual routing and forwarding (VRF). But, there are known attacks to all of them. Examples of how firewalls are circumvented via VPNs, hidden dialups/UMTS and other covert channels are described in the Maroochy water breach discussion (Slay and Miller, 2008). Stuxnet (Falliere et al., 2011) is the prime example for bypassing an air gap. One example for broken VRF is accidentally announcing a BGP full table into the VRF engine. Overall, the industrial lore states, from the Security Issues and

Best Practices for Water/Wastewater Facilities (Hayes, 2013): *"Industrial networks are often shared with the business side of the operation. VLANs, sub-networks, firewalls all help to create a layer defense, but are not impervious."*.

This is particularly the case for services that were first envisioned for enterprises and then commonly used in Small Office/Home Office (SoHos) and home networks. In these settings security by default configurations are essential as the users often lack the knowledge and means to properly address security and network challenges.

## 4.5 Complex Security Solutions

At the end of the 20th century the awareness that firewalls were not "the" security solution became prominent. For example, RFC3365 (Schiller, 2002) dated 2002 states: *"History has shown that applications that operate using the TCP/IP Protocol Suite wind up being used over the Internet. This is true even when the original application was not envisioned to be used in a "wide area" Internet environment. If an application isn't designed to provide security, users of the application discover that they are vulnerable to attack."*

As a result, protocol designers realized that (a) there was a need for improved protection architectures, e.g., the work on SANE (Casado et al., 2006) or DoS-limiting network architecture (Yang et al., 2005), and (b) that security had to be an essential feature of future protocols and services. Moreover, just adding another component to ensure security to existing protocols, e.g., firewalls, did not suffice. This fits the increased need for security in the society due to the increasing economic relevance of the Internet (Mahadevan, 2000).

At the same time the diversity of the scenarios also increased with home users, SoHos, enterprises, infrastructure providers, company mergers and splits, etc. Indeed, road worriers started to appear. As a result, more assets were at stake which had to be accessible in many different ways. Thus, versatile security solutions to model complex organizational structures, e.g., via role-based access control (RBAC), were needed.

Indeed, the Danvers Doctrine (Schiller, 2002) stated that the *"IETF should standardize on the use of the best security available"*. Thus, the threat model for this class is: "strong attacker" with "perfect security". The representative protocols we take a closer look at are: IPP, SNMPv3, and IPFIX. Other protocols in this class include: LDAP-ACL, NFSv4, AFS, Postgresq, FTPs, RADIUS/WPA2Enterprise, s/MIME encryption, SSL/TLS, PGP, and seLinux, as an example from system security.

### 4.5.1 Example Protocols

**IPSec:** IPSec, first introduced in RFC1825–1829 (Atkinson, 1995) and updated by RFC4301–4309 (Kent and Seo, 2005), is a suite of protocols that promise to seamlessly extend IP with

authentication, data integrity, confidentiality, non-repudiation, and protection against replay attacks.

To cite Ferguson and Schneier (Ferguson and Schneier, 2000): *"Our main criticism of IPsec is its complexity. IPsec contains too many options and too much flexibility; there are often several ways of doing the same or similar things. This is a typical committee effect."* Thus, this is a prime example of this class. However, IPSec is often used as an argument why it is possible to leave out certain security features in other protocols (Aboba et al., 2004; Romanow et al., 2005).

IPSec is in use for corner cases such as LTE backend network security (Bikos and Sklavos, 2013). Indeed, as Ferguson and Schneier state (Ferguson and Schneier, 2000): *"Even with all the serious criticisms that we have on IPsec, it is probably the best IP security protocol available at the moment."* Still, IPSec has not yet seen widespread deployment, e.g., (Richter, Chatzis, et al., 2015). One possible reason is usability. Here we cite Gutmann (Gutmann and Grigg, 2005) *"If we consider security usability at all, we place it firmly in second place, and anyone wishing to dispute this claim is invited to try setting up an IPsec tunnel via a firewall or securing their email with S/MIME."*

**LDAP:** LDAP, the Lightweight Directory Access Protocol, is a protocol for accessing and maintaining distributed directory information. It builds upon the ideas of X.500 but differs, in particular, with regards to security features (Hassler, 1999) and simplicity. LDAP is designed to be extensible and flexible, see the many LDAP related RFCs, including RFC2251 (Wahl et al., 1997) dated 1997, and RFC4510 (Zeilenga, 2006) to RFC4519 (Sciberras, 2006). LDAP organizes its data in a tree like ASN.1 (*Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation* 2002). It is often used to organize organizational information, groups, and, in particular, user data, including account information, personal information, and authentication data.

LDAP offers transport security. It also provides various forms of authentication, including SASL and Kerberos, see RFC4513 (Harrison, 2006). The authorization concept of LDAP is probably one of the most complex, yet, also most powerful systems currently available. In LDAP, the ACL roughly follows role-based access control (Harrison, 2006). It distinguishes between anonymous, authenticated, and specific connections. Specific connections are defined by properties on the object that holds the connection. This may be, but is not limited to, group membership, subtree membership, tree position, attributes in the object, etc.

The main misconfiguration opportunities for LDAP are (a) that operators use LDAP over the Internet without transport layer encryption and (b) that operators make mistakes while setting up access control (Findlay, 2011; T. Xu, J. Zhang, et al., 2013).

Especially when used for authentication, the bootstrap process is hard. Clients have to be configured and authenticated correctly, as well as authorized to access the information in the LDAP tree, and use it to perform authentication and authorization within their own applications. Furthermore, no reasonable default values can be created for ACLs, as

these depend on the root nodes of the LDAP tree, which is organization dependent. To cite RFC4513 (Harrison, 2006): *" Operational experience shows that clients can (and frequently do) misuse the unauthenticated authentication mechanism of the simple Bind method see (Unauthenticated Authentication Mechanism of Simple Bind)"*

Commercial distributions come with reasonable pre-configured ACLs. The non-commercial ones usually come with one rule, no write access for users that are not system administrators.

**IPP:** The Internet Printing Protocol (IPP) is documented in RFC2565 (Herriot et al., 1999) dated 1999 and its companion RFCs. IPP is an application level protocol suite for distributed printing using Internet tools and technology. It uses HTTP, namely 1.0 or 1.1 as its transport protocol.

IPP itself implements the relevant mechanisms to perform strong authentication - by default against members of a local UNIX group via PAM (Pluggable Authentication Module) and supports the use of transport encryption. For IPP (Herriot et al., 1999) authentication and authorization are critical due to an unrelated security topic, accounting. Printing, or rather, use of paper and ink, must be accounted for in most companies as well as universities. Thus, it is not surprising that the most prominent UNIX based IPP server, CUPS, currently uses TLS for all connections containing credentials. Moreover, authentication is required by default for all administrative actions.

The main misconfiguration trap with IPP is that printing on a device is - by default - allowed for unauthenticated clients (Institute, 2003). Thus, a remote attacker can print on all printers they learn about. While this is usually not a major security problem, it may become an interesting basis for social engineering attacks. In addition, it enables DoS attacks, e.g., if an attacker prints endless numbers of fully black pages. Lastly, it is a nasty way of large scale resource waste. Keep in mind that the IPP service is offered by most major network attached printers by default as well as apple devices if they share their home printer.

**NFSv4 with Kerberos:** NFSv4 is another distributed file system protocol. Unlike its predecessor NFSv3, see Section 4.4, NFSv4 has support for strong security and its negotiation built in. NFSv4 uses a principal-based authentication model rather than machine-based as prior versions of NFS did.

The NFS standard (Haynes and Noveck, 2015) mandates strong security using Kerberos (Zhu et al., 2005). Kerberos, according to Neuman et al. (Neuman and Ts' O, 1994), is a distributed authentication service that allows processes of a principle to prove their identity to an application server without sending data across the network that might allow an attacker to impersonate the principal. Kerberos also provides integrity and confidentiality.

This sounds great. However, while the Storage Networking Industry Association (SNIA) states (Ethernet Storage Forum, 2015) that *"With careful planning, migration to NFSv4.1 and NFSv4.2 from prior versions can be accomplished without modification to applications or the supporting operational infrastructure, for a wide range of applications; home*

*directories, HPC storage servers, backup jobs and so on."* system lore says that NFSv4 deployment is lacking.

Indeed, NFSv4 may come with a performance penalty (M. Chen et al., 2015) and, as McDonald points out *"One area of great confusion is that many believe that NFSv4 requires the use of strong security. The NFSv4 specification simply states that implementation of strong RPC security by servers and clients is mandatory, not the use of strong RPC security. This misunderstanding may explain the reluctance of users to migrate to NFSv4, due to the additional work in implementing or modifying their existing Kerberos security."*

Thus, we conclude that even though strong security options exist, often system administrators choose to not deploy them. One reason is the inherent complexity of Kerberos. Indeed, Bouillon in his Black Hat EU 2009 talk stated (Bouillon, 2009) *"However, lots of system administrators still make dramatic mistakes while configuring it (those mistakes are made more likely by buggy GUIs and their poor documentation)..."*. Indeed, from discussions with several operators we learned that they often hesitate to deploy Kerberos due to its complexity, lack of documentation, and difficulty debugging its deployment.

As a consequence it is not surprising that the Shodan scans find many NFSv3 deployments (100,000) but no NFSv4 deployments. Even though NFSv4 should be easier to detect as it uses the IANA port 2049 and does not rely on portmap.

**SNMPv3:** SNMPv3 (Case, Mundy, et al., 2002; Stallings, 1998) is the successor to the Simple Network Management Protocol v2, see Section 4.4. The motivation for SNMPv3, RFC3410 (Case, Mundy, et al., 2002) was to fix: *"The unmet goals included provision of security and administration delivering so-called "commercial grade" security with: authentication . . . , privacy . . . ; authorization and access control; and suitable remote configuration and administration capabilities for these features."* Thus, SNMPv3 makes few changes to the protocol aside from adding the option of on-the-wire encryption. Rather, it focuses on two main aspects, security and administration (Corrente and Tura, 2004). SNMPv3 supports the notion of users and authorization.

The main misconfiguration trap with SNMPv3 is using SNMPv2 instead or in parallel. Indeed, SNMPv2 and SNMPv3 are not exclusive. A host offering SNMPv2 can also offer SNMPv3 even on the same port. Today many hosts indeed offer SNMPv2 and SNMPv3 simultaneously (Frye et al., 2000, 2003). Since SNMPv2 is often enabled by default the hosts remain vulnerable even though they support SNMPv3. Therefore, it is not surprising that we still see more than 3,800,000 hosts with SNMPv2 enabled. Indeed, this is supported by data from Cisco Advanced Services from May 2013 (Bollinger, 2015) which reports on the SNMP configuration status of 1,724,827 device configurations in 2013, see Table 4.1. Even though the adoption of SNMPv3 has increased it has been at a lower rate than SNMPv2. Part of the reason is the complexity of SNMPv3 as outlined, e.g., by Cisco training material (Bollinger, 2015).

However, even if SNMPv3 is correctly deployed, the corresponding RFCs come with another misconfiguration trap. The original RFCs, RFC2264 (Blumenthal and Wijnen, 1998)

| Protocol Version | Customers | Devices | Change from 2012—2013 |
|---|---|---|---|
| SNMPv2 | 98.5% | 88.2% | up 9% |
| SNMPv3 | 34.6% | 10.4% | up 4% |

Table 4.1: Adaption of SNMPv3 over time following (Bollinger, 2015).

to RFC3414 (Blumenthal and Wijnen, 2002), only specify DES as a cipher. AES was added significantly later, RFC3826 (Blumenthal, Maino, et al., 2004). Similarly, the only supported digest algorithms are MD5 and SHA1. This leads to implementations with weak crypto and, thus, are susceptible to advanced attacks (Lawrence and Traynor, 2012). SSHv1 suffers from similar problems as CRC32 is fixed in the protocol specification (D. J. Barrett et al., 2005).

**IPFix:** The purpose of the IP Flow Information Export (IPFIX) protocol, please see RFC5101 (Claise, 2008), is to transfer IP Traffic Flow information from an exporter to a collector. IPFIX is the vendor-independent successor of NetFlow version 9, see Section 4.4. IPFIX supports flexible definitions of network flows via a template based extensible information model. The design goal was to make the protocol future proof as well as applicable to all network protocols.

Given the inherent insecurity of NetFlow the goal of IPFIX was to incorporate strong security in its design, see RFC3917 (Quittek et al., 2004). This resulted in RFC5101 (Claise, 2008) which states that IPFIX must ensure confidentiality and integrity of the transferred IPFIX data and authentication for the exporter and collector.

However, none of the major vendors, including Alcatel, Cisco, and Juniper implement any of the CIA mechanisms of the IPFIX RFC. Thus, the same exploits and misconfigurations that apply to NetFlow also apply to IPFIX, see Section 4.4. Hence, the major misconfiguration trap is insufficient fencing.

### 4.5.2 Discussion

The common design of all protocols within this class is that they are designed according to the Danvers Doctrine (Schiller, 2002). Therefore, they all offer full CIA support. Unfortunately, when looking at the deployment base, we either find few indications of actual use of the protocol or that the deployed instances do not utilize or implement the full CIA support.

Among the reasons are difficult setup (LDAP, NFSv4) or limited perceived benefit of latest version of the protocol (NFSv4, SNMPv3). Also, they are often used within the SoHo, rather than the enterprise, where there is a lack of required security infrastructure (IPP without Kerberos). Furthermore, implementations do not support required protocol security features (IPFIX). Furthermore, third party documentation may recommend simpler solutions, e.g., Postgres.

Another aspect is that the system administrator and the IT security professionals are typically not the same person, e.g., (Botta et al., 2007; Furnell et al., 2009). Moreover, these systems come with substantial complexity which leads to many misconfiguration traps. For example, Xu et al. (T. Xu, Jin, et al., 2015; T. Xu, J. Zhang, et al., 2013) in their papers *"Do Not Blame Users for Misconfigurations"* and *"Hey, You Have Given Me Too Many Knobs! Understanding and Dealing with Over-Designed Configuration in System Software"* point out that major causes of today's system failures are misconfigurations due to complexity. However, it is not necessarily the user or the system administrator who is to blame, but rather the inherent complexity and mismatch of the tools (R. Barrett et al., 2004; Haber and Bailey, 2007) as well as poor usability for system operators (T. Xu, Jin, et al., 2015; T. Xu, J. Zhang, et al., 2013).

Given the above complexity, let's review what might or might not motivate a system operator to deploy the latest secure protocol suites, see, e.g., the discussion of West as well as Schneier in their papers on *"The psychology of security"* (Schneier, 2008; West, 2008): (a) No or little reward for secure behavior. Indeed, there are hardly any monetary incentives for deploying the secure versions. Some claim that such incentives might change this (S. J. Greenwald et al., 2004). (b) The misconception of operators that there is a low risk of being attacked. (c) The "laziness" of the operator that wants to get the main service working first and then worry about security if there is time left. (d) The time pressure by management that forces the operator to get a service working in no time. (e) The presumption that no one is likely to get caught for not deploying the secure version of the service. (f) The maintainability of the complex security infrastructure.

## 4.6 A new Simplicity

The inherent complexity of protocol suites of the previous class provided the motivation for trade-off based security. This class is about "strong attackers" and "good enough" security. As noted by Bruce Schneier (Schneier, 2008): *Security is a trade-off. This is something I have written about extensively, and is a notion critical to understanding the psychology of security. There's no such thing as absolute security, and any gain in security always involves some sort of trade-off.*

The protocol designs in this class must be seen in the context that we now have clouds as well as many start-up companies with various (mobile) applications. Indeed, the Internet is experiencing yet another growth explosion in terms of services (Armbrust et al., 2010; Pallis, 2010). In terms of infrastructure, we are now in a "world of HTTP everywhere", virtualization, the cloud, large scale as well as microservices architecture, and configuration orchestration.

Let us consider Internet application developers. Among the easiest ways to develop new applications is to do it in the cloud using a microservice architecture. They can rely on "Node.js" or similar programming languages and reuse existing code. The reuse of existing code has been made easy by the equivalent of appstores. As transport protocol they will

most likely use HTTP/HTTPs. Moreover, the application grows as the feature sets and/or user base increases. A common observation is that the security review of such services is often lacking and the assumption is that it is possible to fence the service in a private cloud as it is just another Web service.

With "simplicity" we refer to the security concept rather than the feature sets of the protocols or, rather, protocol suites. The threat model for this class is: "strong attacker" with "good enough". The representative protocols/protocol suites we examine are: Telnet, key-value stores, VNC, and the many incarnations of (HTTP/Socket) APIs. Other protocols in this class include: PulseAudio's and Systemd's internal protocols as well as control protocols for tools such as Nessus, Aircrack, etc. Since most of these use HTTP as transport layer protocols the same misconfiguration traps apply as discussed in the API subsection.

### 4.6.1 Example Protocols

**Telnet:** Back in 1969 the idea of the Telnet protocol, e.g., RFC15, RFC137, RFC854, RFC5198 (Carr, 1969; Klensin and Padlipsky, 2008; O'Sullivan, 1971; Postel and Reynolds, 1983), was to make a terminal usable by a remote host as if it was local. The result was a bi-directional, byte-oriented communication protocol. Since SSH (Ylonen and Lonvick, 2006) was designed, in 1995, as a secure replacement for Telnet, rlogin, etc. (Ylönen, 1996) Telnet usage should have declined to almost zero. Therefore, if at all we should have discussed Telnet in Section 4.3.

Unfortunately, today for many application and infrastructure systems, e.g., Customer Premise Equipment (CPEs), Storage Area Network (SAN) Devices, and what is now the Internet of Things (IoT) and Industry 4.0, Telnet is again the default choice for accessing devices (Foster et al., 2015) - with attacks skyrocketing in the past 18 months (Pa et al., 2015). After all, Telnet is relatively simple. Thus, most of these devices already come with a built-in daemon from the original design/equipment manufacturer (ODM/OEM) that supports a subset of the Telnet protocol in their firmware templates (Costin et al., 2014). This is the reason we discuss Telnet in this section.

The original versions of Telnet did not offer authentication/authorization or encryption. Telnet authentication defaulted to the operating system's login. Various authentication and authorization mechanisms including Kerberos and RSA were added to Telnet in 1993, see RFC1409 (Borman, 1993). Adding TLS to Telnet was suggested by an Internet draft in 2000 (Boe and Altman, 2002). A recent publication on vulnerabilities in telematic systems found that for all investigated devices authentication was not enabled for the Telnet interface (Foster et al., 2015).

CPEs differ from operator equipment in the sense that they are actually operated by the end users. They also differ from the typical end-user equipment in the sense that they are often pre-configured and directly reachable via the Internet. Given vendors pre-provisioning their products with known default credentials, these CPE devices can be vulnerable as soon as they are accessible over the Internet. The extent to which this can be a problem was

demonstrated in 2012 by the Carna botnet (Carna Botnet, 2013; Krenc, Hohlfeld, et al., 2014). Another major attack used these devices to change the DNS settings of the end-users and, thus, did a monkey in the middle attack (Assolini, 2012).

Mitigation is relatively easy. Vendors should rethink if Telnet access is necessary at all. Even if it is, it should be possible to physically turn it on/off with a small dip switch on the device - with default off. Furthermore, they should not come with no, default, or guessable credentials. Indeed, the initial credentials should be unique for each device and should not be computable from anything that is also related to the device. Various vendors already use this approach. (Lorente et al., 2015)

The documentation of the Telnet service is usually very limited as the documentation usually focuses on the devices themselves and only mentions that Telnet is supported. In the past most CPE services used default configurations with weak credentials.

Today, we find more than 10,600,000 devices with an open Telnet port according to Shodan. Indeed, the Carna botnet exploited about 1,200,000 of these devices (Carna Botnet, 2013). Furthermore, a recent study by Pa et al. (Pa et al., 2015) finds that the attack volume on Telnet enabled IoT devices has increased by many orders of magnitude since 2014.

**Key-Value Stores:** A useful Internet service is provided by key-value stores which are widely used by companies such as Amazon, Facebook, Digg, and Twitter (Atikoglu et al., 2012). Key-value stores provide a mapping between keys and their associated data and can, if implemented *In-memory*, avoid classic I/O bottlenecks. Thus, they allow quick non-relational data storage. Examples include Dynamo, MongoDB, Redis, and memcached. Initial development started around 2004. A first paper (DeCandia et al., 2007) reporting on "Dynamo, Amazon's highly available key-value store" appeared in 2007. While not a protocol in themselves key-value stores are an almost universal network service and often rely on HTTP/HTTPs or JSON for their communication.

Initially, most of the key-value services did not support authentication, authorization, and/or encryption. However, over time most were augmented with support for authentication, authorization, and encryption.

To highlight the underlying assumption of the design of all of key-value stores we cite the Dynamo paper (DeCandia et al., 2007): *"Other Assumptions: Dynamo is used only by Amazon's internal services. Its operation environment is assumed to be non-hostile and there are no security related requirements such as authentication and authorization."*

Most key-value stores allow for transitive attacks. When an attacker can access a central web session storage they can impersonate users and or administrators. When using key-value stores for caching, e.g., of SQL requests or of pre-compiled Just-In-Time byte code, access to the cache can result in information disclosure or even attack code execution.

Next we give a few examples, using memcached, of the defaults when deploying key-value stores. For memcached strong authentication on the basis of SASL was introduced in 2009. Moreover, memcached now supports transport layer encryption. However, this support

is lacking by default in most Unix-derivatives as the implementation of SASL was flaky in memcached and led to bugs. We surveyed the following distributions for SASL support: *Gentoo, Debian Wheezy, Ubuntu 14.04.1 LTS, Arch Linux, Centos 6, OpenBSD 5.6, FreeBSD 10*. We found that only since 2014 *Debian Wheezy* and *Ubuntu 14.04.1 LTS* started to link against SASL by default.

There are two options regarding the configuration of the listen socket, restrictive or global. Restrictive, means a specific IP, e.g., the localhost IP. `127.0.0.1`, or non restrictive using `0.0.0.0`. We find that for memcached only *Debian Wheezy* and *Ubuntu 14.04.1 LTS* use the restrictive default.

We find, not surprisingly that the documentation is large given the feature set of the services. Instructions for enabling the optional security features are hidden in Dynamo, misleading in memcached, and clear and simple in Redis. It seems that initially the security documentation of MongoDB was also hidden. On the 10th of February 2015 more than 40.000 MongoDB databases were unprotected on the Internet. Since then a major update to the documentation took place (MongoDB website, 2015).

Indeed, the weaknesses in the security models of NoSQL databases are known. While Srinivas and Nair (Srinivas and Nair, 2015) conclude that they are on a good path forward with regard to providing CIA, Okman et al. (Okman et al., 2011) point out that *"Clearly the future generations of such DBMSs need considerable development and hardening in order to provide secure environment for sensitive data which is being stored by applications (such as social networks) using them."*.

Binaryedge (Binary Edge, 2015a) finds that there are still more than 175K unprotected Redis/MongoDB/Memcache instances in the Internet that can be contacted from any host.

**VNC and the Remote Frame Buffer Protocol:** Virtual Network Computing (VNC) describes the programs and application providing the Remote Frame Buffer (RFB) Protocol, RFC6143 (Richardson and Levine, 2011) dated 2011. RFB allows a networked computer to use a graphical user interface on a remote machine. It has two common use cases (a) remote support for end-user systems by support staff and (b) remote administration of physically inaccessible servers. The latter is, in particular, used in Linux based virtualization solutions that provide access to the VGA output of the VMs via VNC (Hammel, 2011). The latter is the reason why we consider VNC in this Section.

To cite from the RFB RFC6143 (Richardson and Levine, 2011): *"The RFB protocol as defined here provides no security beyond the optional and cryptographically weak password check described in Section 7.2.2. In particular, it provides no protection against observation of or tampering with the data stream. It has typically been used on secure physical or virtual networks."* However, it can be enhanced with IPsec or SSH for encryption and some implementations support plain and certificate based authentication.

Since VNC is used for remote administration, it offers many opportunities for misconfiguration. An obvious end-user system misconfiguration, weak or no passwords, opens the end-systems to easy attack. Of course, since these are usually company provided, they

"should not" be unprotected in the Internet. For remote administration of SCADA systems this is worse, as access to them implies access to industrial control system (Binary Edge, 2015b; Stevenson, 2014).

VNC for remote access to virtual machine consoles is often used by Linux based virtualization solutions, e.g., libvirt and Xen. These enable VNC without authentication by default. Moreover, while VNC is typically bound to localhost libvirt offers the "convenience" option of using `0.0.0.0` by uncommenting a more restrictive binding to localhost. If an attacker gets control of a VNC port they can use keyboard commands to reboot the system (Richardson and Levine, 2011) and boot into single user mode where the credentials for the administrative account may be changed.

To mitigate this, strong, ideally key-based, authentication and built-in encryption is needed. Therefore, the protocol must be adjusted. Furthermore, the implementations should provide these security mechanisms in an easy to use manner. Shodan lists roughly 600,000 publicly available IPs running VNC. Indeed, roughly 2,000 unprotected VNC instances have been used as the basis of a "security" roulette, see the report by Stevenson (Stevenson, 2014).

**APIs and Microservices:** Remote procedure calls (RPCs) are a fundamental concept introduced before 1981, with the goal to make execution of code on a remote machine as simple and straightforward as on the local machine (Birrell and Nelson, 1984; Nelson, 1981). Today RPCs are everywhere, but often hidden behind a different name, examples include JSON-RPC, XML-RPC, Java-RMI, SOAP, REST. Indeed, one often refers to them as application programming interfaces (APIs).

APIs are commonly used in, e.g, (a) microservices, (b) configuration interfaces, (c) mobile application services, and (d) Web applications. Microservices (Halteren and Pawar, 2006; Newman, 2015; Pratistha et al., 2003) reuse the traditional Unix philosophy of combining "small, sharp tools" (A. Hunt and Thomas, 2000). The communication is delegated to APIs. An example of a configuration interface is the Docker API which allows operators to orchestrate applications built in docker containers via Web services (Docker.com, 2015). Examples of mobile applications are the various APIs used in a multitude of Android and iOS apps (Fiebig, W. Katz, et al., 2013; Masse, 2011; Polakis et al., 2015). Web applications often rely on client-based JavaScript code that calls back to APIs on the server-side (Cantelon et al., 2014; Charland and Leroux, 2011). While APIs often use RPCs they do not have to. Some use HTTP as a transport protocol and are either REST-full or use JSON or XML as basis. Others use plain JSON or plain XMLRPC. Yet another group defines their own protocol often using binary encoding.

Some APIs provide some form of authentication, authorization, and encryption. However, APIs are often used in what the application designer thinks is a fenced environment. Hence, the typical use case has most security elements disabled (Alarcón and Wilde, 2010; Halteren and Pawar, 2006). This is the reason that APIs are in this class as they basically repeat the misconceptions of ***Early Internet***, namely to trust every client.

Most attacks against APIs are ones that "just" use the service. This can result in non-intended side-effects due to missing authentication and authorization, e.g., abusing the docker API, or information leakage, e.g., in mobile application APIs (Fiebig, W. Katz, et al., 2013; Polakis et al., 2015), or using the back-end API rather than the client interface to overcome rate limits against brute force attacks on the original service (Fallon, 2015). These attacks are enabled by misconfigurations such as APIs that are Internet-wide accessible due to (a) holes in the firewall, (b) misconfigured bouncer, and (c) inside attacks via another compromised microservice (Breen, 2015).

We again find that, as with almost all new cool ideas, security aspects are the ones that are considered last. Thus, many APIs come with unclear documentation and/or global binds for their management APIs, e.g., docker, tomcat, JBoss (Breen, 2015). Even though the meta-documentation tells reasonably well that APIs have to be secured, the practice does not adhere to this goal. Indeed, the problems are known for microservices since 2003 (Pratistha et al., 2003). Moreover, good practices are known but hardly followed (Newman, 2015).

### 4.6.2 Discussion

In this class, we often do not have a single protocol but concepts that are realized by different protocols that all suffer from the same misconfigurations, e.g., key-value stores and APIs. Overall, we observe a clear trend towards using HTTP as application layer protocol, likely because HTTP allows middlebox traversal (Qazi et al., 2013). In theory, this opens up the possibility of "just securing HTTP" rather than having to secure a large number of Internet protocols (Richter, Chatzis, et al., 2015). Still, the problems of misconfiguration remain. Even if an API uses SSL for encryption, it can still be abused by an attacker (Breen, 2015) to disclose information. The origin of this trend is the need for complex, yet easy to deploy services.

Apart from delegating security issues to HTTP these modern services again rely on fencing. During software and system development, the main focus is on getting the service ready and not necessarily on security. Hence, *"[...] Its operation environment is assumed to be non-hostile [...]"* as stated in the Amazon Dynamo paper (DeCandia et al., 2007).

The underlying misconception is that services can be securely fenced off as they are only used within "internal systems". That this can lead to misconfigurations has already been shown in previous discussions. Nevertheless, many recent security incidents have lead to prominent examples of data leakage (key-value stores, APIs) (Binary Edge, 2015a).

The motivation for delegating security to a fenced environment is the notion that security without fencing is a complex and time consuming task, impossible to achieve, and making the (backend) services unusable. Hence, the spirit during development follows the start-up culture, which includes reusing available protocols as they are (Telnet/VNC).

Fencing techniques have advanced as well, and since 2013 include the concept of containers (T. Kim and Zeldovich, 2013). A common misconfiguration is captured by the following

idea: "If we put a service in its own container in its own dedicated VM in its own VLAN behind a firewall nothing can happen". However, since services have to be reachable they have to allow some access. This access often turns out to be the entry point for attacks.

Containers, e.g., Docker, are a deployment and testing mechanism. The motivation is that traditional means of software distribution do not scale to the cloud as traditional testing/release procedures do not fit the rapid development cycle. This often leads to monolithic deployment of formerly modular software components. This is unmanageable if some component has to be updated, e.g., to patch a security hole. Furthermore, containers usually come pre-configured which adds additional misconfiguration traps.

Due to their experience with ***Complex Security*** several developers state that one should not be too strict about security and that security is "in the way" of innovation. This goes as far as e.g. Ren et al. (Ren et al., 2012) claiming that: *"Security and privacy is one fundamental obstacle to cloud computing's success."*.

Old mistakes are redone as the default assumption of the software developer is again "this is an esoteric scenario and not the intended use case". APIs may release more information than intended (Fiebig, W. Katz, et al., 2013; Polakis et al., 2015). Moreover, the cloud adds the complexities of multi-party trust and the need for mutual auditability (Y. Chen et al., 2010). Additionally, developer guidelines with a focus on security are hard to find even if they exist.

Overall, we note that, after having had too complex security and unusable systems, we are now in a world with easily deployable solutions. However, security is often outsourced. This results in many misconfiguration traps.

## 4.7 Lessons Learned

From our observations in this paper we can compile a set of action points and requirements protocol designers MUST consider when they create new or update old protocols.

***Early Internet***: The major lesson from this class is that security requirements and environments change. However, this class also demonstrates that such issues can be tackled. Hence, ***when updating a protocol, one MUST purge problematic use-cases and design choices.*** Only then can appeals for fixing bad configurations combined with sanctioning of insecure practices lead to improved overall security (see, e.g., (Jung and Sit, 2004) and (Kührer et al., 2014)). While the community succeeded with SMTP, the Internet still suffers from, e.g., DNS and NTP amplification attacks. In addition, other protocols from this era still linger and urgently require revisiting.

***Emerging Threats***: Given that the non-hostile Internet is gone, a new paradigm emerged. Instead of adjusting the protocols to the new insecure environment, the environment is redefined to be fenced. Thus, the assumption is that the service is behind a firewall or that lower layers offer security guarantees. Yet, it is common knowledge that firewalls tend to

| Example | | Security Features | | | Misconfiguration traps | | | | | Support | | Publications | | Visible Instances | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Introduced | Authentication | Authorization | TLS | NoAuth | Credentials | Artifacts | Fencing | NoUse | Bad Defaults | Bad Documentation | Academia | Lore | Total | Affected |
| **Early Internet** | | | | | | | | | | | | | | | |
| FTP | 1971 | ● | ○ | | ● | ● | | | | ○ | ○ | | 1999 | 3,000,000[c] | |
| TFTP | 1981 | | | | | | | ● | | | | 2015 (Macfarlane and Buchanan, 2015) | 2006 | | 600,000[a] (Macfarlane and Buchanan, 2015) |
| SMTP | 1982 | ○ | | ○ | ● | | ● | | | | | 2004 (Jung and Sit, 2004) | 1995 | 5,600,000[c] | |
| DNS | 1984 | - | - | | ● | | ● | | | | ○ | 2004 (Specht and R. B. Lee, 2004) | ~2000 | 10,000,000[c] | >5,000[a] (Streibelt et al., 2013) |
| **Emerging Threats** | | | | | | | | | | | | | | | |
| NetFlow | ~1990 | | | | | | | ● | | | ● | | 2004 | | |
| DHCP | 1993 | - | | | ● | | | ● | | ● | ● | | 1993 | | |
| SNMPv2 | 1993 | ○ | ○ | | | ● | | ● | | ● | ● | 2014 (Rossow, 2014) | ~2012 | | 3,800,000[c] |
| NFSv3 | 1995 | ○ | ○ | ○ | ● | | | ● | | ○ | ○ | 1999 (Harris and R. Hunt, 1999) | 2015 | 100,000[c] | |
| Munin | ~2000 | | | | | | | ● | | | | | | | 6,000[c] |
| iSCSI | 2002 | ○ | ○ | | ● | | | ● | | ● | ○ | | 2009 | | 1,000[b] |
| **Complex Security** | | | | | | | | | | | | | | | |
| IPSec | ~1995 | ● | ● | ● | | | | | ● | ○ | ○ | 2000 (Ferguson and Schneier, 2000) | 2005 | | |
| LDAP | 1997 | ○ | ○ | ● | ● | | | | | ○ | ○ | | 2006 | 200,000[c] | |
| IPP | 1999 | ○ | ○ | ○ | ● | | | | | ● | ● | | 2003 | | 400,000[c] |
| NFSv4+Krb5 | 2000 | ● | ● | ● | | | | | ● | | ● | | 2009 | | |
| SNMPv3 | 2002 | ● | ● | ● | | | ● | | ● | ○ | ○ | | 2013 | | |
| IPFix | 2008 | - | - | - | ● | | | | | ○ | ○ | | 2014 | | |
| **A new Simplicity** | | | | | | | | | | | | | | | |
| Telnet | 1969 | ● | ○ | | ● | ● | | ● | | ● | ● | 2014 (Krenc, Hohlfeld, et al., 2014) | 1993 | 10,600,000[c] | 400,000[a] (Carna Botnet, 2013) |
| KV Stores | 2006 | ○ | ○ | ○ | | | | ● | | ○ | ○ | 2011 (Okman et al., 2011) | 2015 | | 175,000[a] (Binary Edge, 2015a) |
| APIs | 2000 | ○ | ○ | ● | ● | | | ● | | ● | ● | 2003 (Pratistha et al., 2003) | 2013 | 200,000[c] | |
| VNC | 1998 | ○ | | ○ | ● | ● | | ● | | ○ | ○ | 2011 (Holm et al., 2011) | 2014 | 600,000[c] | >2,000[a] (Binary Edge, 2015b) |
| | | ●: Commonly Used<br>○: Uncommon<br>-: Not Implemented<br>: Not Specified | | | ●: misconfiguration traps | | | | | ●: Most<br>○: Some | | Year Published<br>[example ref.] | | [a] cited data<br>[b] zMap<br>[c] Shodan | |

Table 4.2: Summary of all selected representative protocols.

fail eventually and lower layers do not hold up to their promises. Thus, ***when designing a protocol, one MUST not assume that it will only be used in the environment it was designed for.*** Moreover, ***when designing a protocol, one MUST include authentication, authorization and confidentiality preserving methods***.

***Complex Security***: Here the community tried to address some of the above recommendations. However, the resulting protocols come with other complications. Security features designed for an enterprise setup may not be appropriate in SoHo scenarios, e.g., recall IPP. IPSec is so complex that it is not in widespread use. Moreover, implementations do not necessarily interoperate. In an attempt to ensure good cryptographic algorithms, SNMPv3 and SSHv1 specified algorithms that were good then but not necessarily now. Hence this class provides us with the following three lessons. First, ***when designing a protocol, one MUST ensure that the security is scalable.*** Scalable in the sense that it is applicable to large as well as small setups. Even the simplest setup should provide confidentiality, integrity, and availability by default. Second, ***when designing a protocol, one MUST keep it simple and concise***, to allow multiple parties to implement interoperable solutions. Third, ***when designing a protocol, one MUST not mandate the use of specific ciphers but one MUST exclude plain and weak algorithms*** since cryptographic algorithms become obsolete over time.

***A new Simplicity***: Overwhelmed by the complexity of the previous protocol generation a new protocol design trend emerged. Protocols become simple again, and designer focus on getting things done. New technologies, such as "Cloud" and "SDN" also bring new paradigms. We find that large enterprises develop protocols very similar to those of the era of emerging threats while ignoring lessons learned often for the sake of performance. It works well for the designed environment, e.g., large enterprises where the perimeter is secure, but fails in different settings. The concept of "everything over HTTP" leads to "protocols" which ignore and/or misuse basic concepts such as authentication and in particular authorization. In addition, we find that often new actors, e.g., the automotive industry, adapt IP based technologies and re-do old mistakes. Hence, the most important takeaway from this class is that ***when designing a protocol, one MUST take past lessons into account***. While this sounds simple history tells us it is not. Indeed, almost all misconfigurations pointed out by this survey are covered in even the most basic security textbooks. Yet, they remain common and repeatedly cause major data leaks and/or outages. Thus, the prime message of this survey is to finally apply what we have learned.

***General Observations:*** Protocol designers MUST be strict in requiring all security features from implementations and this must be reflected in the RFCs or related standards. To advance the use of TLS, a decent infrastructure, e.g., DANE (Barnes, 2011; Hoffman and Schlyter, 2012) is necessary. If such an infrastructure is unavailable encryption should at least be performed opportunistically (Roth et al., 2005; Ylönen, 1996).

Opportunistic encryption raises the question of usability. Usability has to become a key part of the protocol design process: The goal has to be to make the protocol secure while designing the system such that it is simple to comprehend and use. The correct way is either

the only way or at least the easiest way (Balfanz et al., 2004; Bernstein et al., 2012; Roth et al., 2005).

Good examples of such design practices exist in the context of user centric protocols, e.g., SSH (Ylönen, 1996) and the large set of encrypted (mobile) messaging protocols (Unger et al., 2015). The rise of the latter is motivated by the—emotional—demands of end-users (Kraus et al., 2015) and professionals (McGregor et al., 2015) for easy but secure communication. Important security features include strong mutual authentication and opportunistic encryption. The same kind of protocol design is needed for all Internet protocols and, in particular, those used within the infrastructure.

***Misconfigurations and Data Emancipation:*** Besides the issues in protocol design itself we need usable security. That is, usable security in the context of operators that have to deploy and maintain services, rather than in the context of end-user. While this has been discussed since at least 2001 (Gutmann and Grigg, 2005; Schultz et al., 2001), few steps in this direction can be observed.

However, this problem has an by far larger perspective in the sense of privacy. In ***Complex Security*** we can observe how complex enterprise security solutions are not easily adapted by smaller setups, e.g., for IPP. Similarly, in ***A new Simplicity*** we can see how the concepts of large cloud providers, e.g., key-value stores, is easily misconfigured in small setups. While in the first case a private home installation of IPP lack the presence of a full fledged kerberos setup, small setups using key-value stores lack a dedicated fenced of backend network. These problems also stretch on if one considers software security update policies and the constant changes in current best practices.

Large companies, like Facebook, Google, and Amazon, have large and usually skilled system administration teams. These can handle complex security methods and have the resources to provide fenced of backend infrastructure. User-data stored there is relatively secure against access that is not sanctioned by the respective large organization. In contrast, smaller, distributed setups, suffer from misconfigurations, forgotten software updates and neglected servers.

The privacy issue here is between these two extremes. On the one hand large corporations storing data securely—but sharing it with government agencies an harvesting user-data to sell it for marketing purposes. On the other hand the decentralized approach propagated by many NGO's (Non Government Organizations) like the EFF (Electronic Frontier Foundation), motivating—untrained—users to configure and run infrastructure connected to the Internet.

***State of the Internet:*** Motivated by our classification, we did take a closer look at several protocols and find the following new attack vectors that have not yet been well documented or who's impact may have been underestimated:

- **iSCSI:** Misconfigured firewalls and missing authentication lead to more than 5,000 iSCSI hosts exporting block devices that are accessible Internet-wide. This includes

large commercial institutions, several major universities and even nuclear research facilities around the world.

- **TFTP:** We identified several information disclosure and manipulation vulnerabilities. These allow further attacks on hosts using a TFTP server. This affects more than 600,000 hosts. Namely, those that already suffer from the previously known amplification attack (Macfarlane and Buchanan, 2015).
- **LDAP:** By documenting the misconfiguration traps for LDAP we highlight the potential attack vectors of more than 200,000 publicly reachable LDAP servers.
- **IPP:** We document the misconfiguration traps for IPP that lead to more than 400,000 publicly accessible IPP servers.

These vulnerable systems certainly demonstrate the unprecedented amount of effort that will be required in order to prevent *future* misconfiguration traps, many vulnerable and misconfigured services should already be deployed on the Internet.

***Misconfiguration Mitigation:*** While our work provides a clear indication that the uncovered issues are relevant in practice, prior work has been mostly concerned with investigating the amount of vulnerable systems as a gross total, not investigating how efficient mitigation could be performed. Hence, in the next chapter, we will investigate how the operations community handles mitigation of known to be frequent misconfigurations.

We have to find a way to systematically identify misconfigured services and implementations on the Internet to start mitigation. Internet-wide scanning and large scale misconfiguration mitigation have been successfully demonstrated in the past (Durumeric et al., 2013; Rossow, 2014). Still, limitations exist, especially in the context of IPv6. It remains a challenging engineering, administrative, and ethical problem, to cover all misconfiguration prone protocols.

While challenging, the task of securing our infrastructures, especially the misconfiguration prone cloud environments (D. Chen and Zhao, 2012; Y. Chen et al., 2010), has to be tackled. Even if users are arbitrarily cautious it may not suffice if the back-end protocols of a service that they use have been misconfigured, their user data will spill all over the Internet. Moreover, even if we have secured our protocols and services we still need secure infrastructures, in particular, secure cloud environments (D. Chen and Zhao, 2012; Y. Chen et al., 2010). As demonstrated in Section 4.6, this also requires secure back-end protocols within these environments.

## 4.8 Summary

Our systematization of misconfiguration prone Internet protocols and services is based on assumptions about the attackers—weak vs. strong attacker—as well as security trade-offs—good enough vs. perfect security—in the protocols. We find that misconfiguration prone Internet protocols and their services fall into one of our four classes: ***Early Internet***, ***Emerging Threats***, ***Complex Security***, and ***A new Simplicity***.

We observe that protocol design and service development have come full circle with regard to security. In the sense that initially the Internet was a cooperative environment, see ***Early Internet***. Once it was realized that the Internet was hostile it was presumed that it was possible to fence of services, see ***Emerging Threats***. However, attackers are getting stronger and assets more valuable and, thus, there is the attempt to get security "right", see ***Complex Security***. The drawback is complexity and, thus, we complete the circle back to a simple security model in a presumed friendly environment guaranteed by enhanced fencing mechanisms, see ***A new Simplicity***. Overall, we find many misconfiguration traps in all of the above classes.

Fencing is seen as a major security solution and often used as only barrier protecting major assets. However, fencing is a major misconfiguration trap in itself. After all, with regards to fencing there is the statement from RFC3365 (Schiller, 2002): *"History has shown that applications that operate using the TCP/IP Protocol Suite wind up being used over the Internet. This is true even when the original application was not envisioned to be used in a "wide area" Internet environment."*

In summary, we can make specific observations for the different players involved. We find that protocols that are easily deployable are appreciated by operators. Having to configure security mechanisms can lead to frustration in operators, if they are unable to deploy a service due to their complexity. Hence, not including security measures reduces the chance that an operator becomes frustrated with a service implementation.

Hence, we observe that protocol/service designers move from strong but too complex security to hardly secure but easily deployable software that works "out of the box". Moreover, for many decision makers (i.e. managers) the operational focus is on fulfilling business needs and generating revenue. However, in contrast to a conveniently deployed, but insecure service, frustrated operations personnel that is unable to deploy services does not provide a business benefit. This is only true in the short term, as in the long term a security incident commonly proves to be more costly than proactive security considerations would have been. Nevertheless, these costs are not immediately visible, and, hence, are often overlooked.

In summary, we find that the considerations taken into account during the design of a protocol do indeed inflict upon how easily implementations of a protocol can be operated in a secure manner. We propose various practices that can reduce the impact of this for existing as well as newly developed protocols. Nevertheless, given the considerable number of already existing, misconfigured implementations connected to the Internet, and protocol design not being the sole root-cause for misconfigurations in practice, we have to continually monitor misconfigured services on the Internet, e.g., via scans. Hence, in the next part of this thesis, we investigate how misconfiguration mitigation using scans is conducted in practice.

# Part II

# Misconfiguration Scanning for IPv6

# 5

# Investigating Security Misconfigurations in the Wild

In the previous chapter of this thesis, we discussed security misconfigurations and their possible impact on security properties in Internet services. For misconfigurations it is always relevant, how many systems are actually affected, e.g., see Table 4.2. Thus, in this chapter, we investigate how security misconfigurations can be measured in the wild and which organizations participate in the process of mitigating security misconfigurations on the Internet.

Security misconfigurations, much like any other vulnerable system, have to be found on the Internet. The standard method of finding vulnerable hosts on the Internet is connecting to every possible host[1] and checking, if the service one wants to investigate is running on that host. The process of connecting to all possible hosts, probing for running services, is generally referred to as "scanning."

In this chapter, we introduce the basic background, i.e., IPv4 and IPv6 addresses, and subsequently scan technologies for Internet wide scans. Finally, we provide an overview on how security incidents are currently approached on an Internet scale.

## 5.1 Background

Recall, the Internet consists of a set of computers communicating using either IPv4 or IPv6. Internet services then use protocols that allow communication atop IP. Usually, an Internet service will use a protocol which is implemented atop of another layer of transmission protocols. Common transmission protocols are the User Datagram Protocol (UDP) (Postel, 1980) and the Transmission Control Protocol (TCP) (Postel, 1981c). Both protocols introduce additional challenges to the scanning process.

In general, the scanning process has the following form:

1. **Address liveliness detection** Before a host can be evaluated for the presence of running Internet services, liveliness has to be established, i.e., if a given IP address is currently assigned to a host.

[1] $2^{32}$ for IPv4, $2^{128}$ for IPv6

2. **Transport protocol handshake scan** If an address is assigned to a host, a scan has to establish if any service listens on a given port, or a list of ports. This differs based on the used transport protocol of the service one scans for.
3. **Application protocol handshake scan** Services can be run on different ports. An open port does not necessitate that the service that is assigned to this port by IANA runs. Hence, in addition one has to perform additional requests following the service-under-test's protocol. This is even more critical if one wants to establish if the service is vulnerable to a given misconfiguration or security issue.

Below, we give an introduction to challenges of scanning on a given layer.

### 5.1.1 IPv4

Addresses in IPv4 are $32bit$. They are organized in networks defined by the common prefix of a given size a set of addresses shares (Fuller and T. Li, 2006; Fuller et al., 1993). Of the possible $2^{32}$ addresses in IPv4, a significant portion is not used (Richter, Smaragdakis, et al., 2016). A common technique to establish if an IPv4 address is used by a host is sending an Internet Control Message Protocol (ICMP) type 8 (echo request) message (Postel, 1981a) to the address. If the address is used by a host, it *should* reply with an echo response. However, in practice ICMP echo requests are often filtered (Richter, Smaragdakis, et al., 2016).

### 5.1.2 IPv6

As the possible address space of IPv4 is being exhausted (Richter, Smaragdakis, et al., 2016), IPv6 was designed (Deering and Hinden, 1998). With $2^{128}$ possible addresses, IPv6 offers a significantly larger address space, and is, therefore, unlikely to be exhausted in the foreseeable future. Similar to IPv4, IPv6 has its own version of the ICMP, ICMPv6 (Conta et al., 2006), which also allows the echo request/reply method to be used to detect hosts.

### 5.1.3 Scanning for TCP Services

TCP is a stateful protocol which allows the detection of packet loss (Postel, 1981c). The common way of establishing if any service using a specific TCP port is running on a host is completing a full three-way handshake with TCP. This carries the advantage of providing ground truth, whether a specific TCP port is open. However, for this, the scanning client has to keep state for the connection. This may slow down scanning, if no host uses the target address, and no destination unreachable message is send by the last hop router. A similar effect occurs if a firewall simply drops the SYN packets from the scanning client, or, if the port is not open, but no connection refused ICMP message is sent.

### 5.1.4 Scanning for UDP Services

Contrary to TCP, UDP lacks a concept of state, and, thereby also a notion of acknowledging received packets (Postel, 1980). This means that a network scanner does not have ground truth data on UDP ports being open. Instead, various heuristic approaches are feasible (Lyon, 2007): (i) Using requests on the application level to facilitate an answer, or, (ii) observing ICMP responses to make an educated guess if a port is open, i.e., receipt of "connection refused" ICMP responses.

### 5.1.5 Application Protocol Scanning

As soon as the transport protocol's port has been established to be open, a connection (in case of TCP) or exchange following the Internet service's applications level protocol can take place. This is especially important when scanning for vulnerable systems or security misconfigurations, as it has to be established if the system is, indeed, misconfigured or vulnerable. This can be done by issuing requests for the possibly misconfigured feature, or by sending requests that return, e.g., information on the running software version.

## 5.2 Internet Wide Scans Before zMap

The multi-step process outlined before introduces severe challenges for Internet scale scans. Even for just scanning the $2^{32}$ possible addresses of IPv4, a scanning system has to keep significant state. The following observations used to be to some extent common knowledge, yet, Durumeric et al. (Durumeric et al., 2013) were the first to collectively recognize and approach.

First, general reachability has to be determined. Common scanning tools do not only use ICMP, but also issue TCP connection attempts. This is, for example, the case with nmap (Lyon, 2007), which by default tries to establish various TCP connections, including to the HTTP(s) ports 80 and 443. This process can not be executed for to many hosts in parallel. Hence, it is prone to being stalled by timeouts, lost packets, and slow-to-respond hosts (Padmanabhan et al., 2015).

The subsequent TCP or UDP scan suffers from the same challenge. For each connection state has to be kept, and the scanning process will be slowed down by hosts timing out or responding slowly.

Last, the application protocol layer *again* suffers from slow to respond and timing out hosts. In addition, contrary to the transport layer, it is not necessarily clear which protocol is used by the service. While the service's port number gives some estimation on the implemented service, it is not a definitive indicator (Richter, Chatzis, et al., 2015). In addition, various services exist that actively tar-pit, that is, intentionally slow down possibly malicious clients (Eggendorfer, 2007).

Malicious actors as well as researchers conducting active measurements approached this challenge by distributing their scans over multiple machines which all take care of specific subsets of the Internet. Specific examples include Provos and Honeyman (2001) who used

a large state pool in their specially crafted scan tool to handle TCP SYN staleness, and the CARNA botnet (Krenc, Hohlfeld, et al., 2014). The CARNA botnet took over several thousand embedded devices to use them as a distributed scanning platform, collecting data on the Internet.

## 5.3 zMap: Scanning the Internet in Minutes

As indicated earlier, Durumeric et al. (Durumeric et al., 2013) identified state keeping as *the* major time consumer in Internet wide scans. Correspondingly, they developed zMap, a tool for Internet wide scans. Contrary to established scanning tools, zMap focuses on a single port, trying to identify *all* IPv4 hosts which have this port open.

For this purpose, zMap, especially in later versions (Adrian et al., 2014), has been build with an advanced packet generation function which then en-bulk pushes packets via initially a raw socket and later a PF_RING Zero Copy (ZC) interface to bypass the Linux kernel's TCP/IP interface. Hence, technically, zMap pushes all SYN packets out as fast as possible, and then records all returning SYNACK flagged packets. Receiving a SYNACK flagged packet indicates that the remote host received the SYN packet, and is listening on that port, as it tries to establish a TCP connection with a three-way-handshake. Service discovery in a zMap style scan setup is then performed only on those hosts which have already been established to have an open target port.

## 5.4 Clearing Houses and CERTs/CSIRTs

The development of quicker scanning techniques has enabled the identification of misconfigured or vulnerable systems. That this is, in general, a useful task has been demonstrated by Provos and Honeyman (Provos and Honeyman, 2001). However, the mitigation of security misconfigurations is commonly not handled by researchers. Instead, a community of so called Computer Security Incident Response Teams (CSIRTs) (West-Brown et al., 2003) and clearing houses has formed (Koivunen, 2010). CERT, the Computer Emergency Response Team at Carnegie Mellon University, being responsible for the US, is one of the most famous CSIRTs, but most larger organizations now implement their own teams (Killcrece et al., 2003). These NGOs focus on reducing abuse and security issues by disseminating information, scanning the Internet and notifying affected parties. One of the most prominent scanning operations in this context is the ShadowServer Foundation, which conducts large-scale daily scans of the Internet and provides digests to network operators (ShadowServer Foundation, 2014).

# 6

# A Case Study on Exposed Key-Value Stores

In the last chapter, we reviewed how security misconfigurations are currently identified for mitigation in the wild. In this chapter, we describe a case-study using these techniques. This serves as a foundation to better understand the requirements of these scans, so we can appropriately address the question of how to conduct IPv6 security scans.

For our case study we focus on two commonly misconfigured implementations of in-memory key-value stores. Such in-memory key-value stores are an integral part of today's highly-scalable low-latency Web services. While they are essential to improve Web service performance they should not be exposed to the Internet. Security problems range from data leakage to remote code execution. In addition, following our observations in Chapter 4, in-memory key-value stores are among the most commonly misconfigured Internet services at the time of writing: Novel database implementations whose authentication component is either not present or not configured while the Internet service has been exposed on the Internet.

In this chapter we use a year long data set of exposed Redis and memcached instances collected with regular security scans. Using it, we highlight the magnitude (about 200K exposed instances) of the problem, document new transitive attacks, and explore misconfiguration patterns. We find that the number of exposed instances is constantly on the rise and that even severe problems only lead to temporal decreases. However, by correlating misconfiguration patterns we can explain significant changes in the number of exposed systems. Furthermore, considering this case study in the context of misconfiguration mitigation, we obtain valuable insights into state-of-the-art misconfiguration mitigation.

## 6.1 Motivation

Key-value stores have emerged as a vital technology to support highly-scalable low-latency Web services. They were first documented in research literature by Amazon with Dynamo (DeCandia et al., 2007). To offer high performance and ease of use key-value stores remove features of traditional Relational Database Management Systems (RDBMS) such as inter-object relations and type-dependent data fields. As consequence, key-value stores are

excellent systems for services which need to store small values associated with an unique index key. Their intended deployment environment is a dedicated, well protected, and isolated backend network segment (DeCandia et al., 2007).

To achieve their performance, usability, and slim design goals key-value stores often omit basic security features. Usually, modern key-value stores only offer limited authentication schemes, hardly any authorization, and regularly lack transport encryption. This is a proper trade-off within the intended operational use cases. Originally, key-value store systems were supposed to be deployed in isolated network segments, however, this is no longer the case.

Today, due to their ease of use, high performance, and widespread availability in almost all cloud environments almost every Web 2.0 service relies on key-value stores (Marston et al., 2011). However, this increasing popularity has also lead to a wide spread deployment of unprotected key-value stores. This was first noticed in early 2015 (Srinivas and Nair, 2015) and affects more than 200,000 systems. Indeed, several key-value store implementations are commonly run without authentication or authorization while being Internet-wide reachable. Since then, various authors have investigated what kind of data is stored in such publicly reachable key-value stores (Binary Edge, 2015a; John Matherly, 2015a,b). The data includes personal data, credit card information, medical data, cached Web pages and byte-code, etc. Thus, in principle the data confidentiality and integrity of tens of thousands of Web applications at risk.

Moreover, deployed key-value stores— exposed to the Internet — suffer from a range of attacks beyond the confidentiality, integrity and availability of the data stored within them. These include object code cache injection, persistent cross-site-scripting
(XSS), Web session manipulation, and even remote shell access. Therefore, one would presume that once it is known that key-value stores deployments have to be properly isolated from the Internet the Web services' administrators take action and, indeed, isolate them.

In an ideal world today no key-value stores services should be reachable via the Internet. However, as we find in Chapter 4 that this is not the case. Internet services are commonly misconfigured. With unprotected key-value stores being known as an issue to the operations community for some time (John Matherly, 2015a), they make an interesting subject matter for our case study. Nevertheless, they are yet known long enough for qualitative datasets being available (ShadowServer Foundation, 2014). Furthermore, while overall the number of unprotected instances is growing, Redis, exhibited an additional security issue in the recent past (Sanfilippo, 2015), allowing for observations on the impact such an incident has on the number of exposed instances. This motivates our year-long survey of unprotected instances of two prominent key-value stores, namely *Redis* and *memcached*.

## 6.2 Key-Value Stores

Over the last decade, we have seen a dramatic rise in the use of NoSQL (Not Only SQL) databases. NoSQL, in contrast to relational databases, does not require the use of an explicit schema and trades consistency for flexibility and performance. One type of NoSQL database with particularly good performance are in-memory key-value stores. Key-value stores provide a mapping between keys and their associated values. *In-memory* key-value stores avoid I/O bottlenecks by storing the entire database in memory.

Among the prominent in-memory key-value stores are *Redis* and *Memcached*. These systems are commonly used for low-latency access to shared or cached data (Atikoglu et al., 2012). Similar use cases apply to other key-value stores including MongoDB and ElasticSearch (Binary Edge, 2015a). To understand the security implications of exposing deployed key-value stores to the Internet we highlight what they are used for. We then outline security implications of accessing/tainting data in key-value stores.

### 6.2.1 Use-Cases

Amazon, Facebook, Digg, and Twitter (Atikoglu et al., 2012) commonly use key-value stores to realize their services. Moreover, key-value stores are also used by many smaller companies and startups. Common use cases are caching (Fitzpatrick, 2004), queuing (Turnbull, 2013) and providing featured lists, e.g., lists of the $N$ most recent items (Macedo and F. Oliveira, 2011).

**Caching:** Caching is a principle mechanism for improving performance. Hereby, key-value stores are often used to store previous or intermediate results. One example is traditional database queries where the client first checks if the result is already cached in the key-value stores. If yes it uses the result if not it issues the query against the original database and stores the result in the cache. Due to their in-memory design and key-based lookup pattern key-value stores significantly accelerate performances over traditional relational databases (Atikoglu et al., 2012; Ousterhout et al., 2010).

Another example involves PHP code if *Zend* or *PHP-FPM* is used. Both compile PHP code to byte-code and cache the resulting byte-code, e.g., in a key-value store. Thus, the application server first checks in the key-value store cache if the PHP code is already available as byte-code, and, if yes, skips the compilation. This significantly accelerates application performance.

**Lists:** Key-value stores often include support for ordered lists (Macedo and F. Oliveira, 2011) and in particular for determining the top-N of such a sorted list which is an expensive operation in traditional RDBMSs (Atikoglu et al., 2012). Such lists are often used to hold volatile authentication data. Examples include lists containing web-sessions, file-share links with associated tokens and expiration time, and temporary access tokens handed out to devices after login. Such tokens are needed by all involved application servers as they are

used to authenticate each request. Previous work (Atikoglu et al., 2012; DeCandia et al., 2007) shows that this is one of the major use-cases for key-value stores.

**Queues:** Key-value stores also often support queue types which are are useful for a wide range of applications from email and messaging to workload queues. Worker processes can access those queues to obtain data for distributed compute jobs. This can involve meta-data on the processed job, i.e., a global tracking of the progress on a single job or the actual data to be processed distributively. Distributed system log message management frameworks like *Logstash* rely on key-value stores to handle the log messages of thousands of systems (Turnbull, 2013). Note, that log messages often hold sensitive information on users' actions or may have to be covertly altered by an attacker to hide the evidence of a compromise (Schneier and Kelsey, 1999).

### 6.2.2 Transitive attacks

As highlighted above key-value stores are often used to store sensitive data. Thus, access to or tainting of this data leads to confidentiality, integrity, and availability problems for the key-value stores itself, see, e.g., Zahid et al. (Zahid et al., 2014) for an overview. However, there are also severe consequences for the service using the key-value store. Below we outline several, so far unreported, transitive attacks on such services via their key-value stores.

**Object Code Cache Injection:** In case key-value stores are used as caches, e.g., for PHP object code, an attacker can replace the cached object with her own. Hereby, the attacker can force the application server to execute her object code the next time the corresponding Web-page is accessed. This allows full remote code execution by an attacker.

**Stored XSS:** In a similar way an attacker can abuse the key-value store for persistent XSS (Cross-Site-Scripting). By injecting, e.g., JavaScript, into cached HTML templates an attacker can distribute her code to all clients which access web-sites that use these templates. This constitutes an easily executed drive-by attack.

**Web-Session Manipulation:** If, for example, an authentication token (SessionID) with the session data of an application is stored in the key-value stores (Atikoglu et al., 2012) an attacker can impersonate users or elevate the privileges of her own session within the application's context.

**Redis Remote Shell:** In October 2015 Salvatore "antirez" Sanfilippo, the lead developer of *Redis*, demonstrated a *Redis* specific exploit in a blog article (Sanfilippo, 2015) which leads to a remote shell. Since *Redis* allows an attacker to alter its configuration the attacker can instruct it to use `~/.ssh/author ized_keys` as database. By entering a ssh-public-key enclosed by "\n" as data for an arbitrary key to that database the attacker can obtain shell access.

## 6.3 Dataset

One of the Shadowserver Foundation's projects (ShadowServer Foundation, 2014) aims at reducing the set of affected systems for various protocols that are potentially vulnerable, misconfigured, or packet amplification sources. Among these potentially vulnerable systems are, e.g., Redis and memcached.

The Shadowserver foundation utilizes a distributed scanning cluster to regularly scan the Internet for affected systems. It checks if a service is reachable and for key-value stores if they reply to status requests. Due to the missing privilege separation, this indicates full access to a key-value store. Thus, this key-value store is considered affected. Hence, these scans follow the zMap-style (Durumeric et al., 2013) two-step approach: First the general reachability of a port is established. Then, a service specific protocol handshake is performed. The scans check for Redis on port TCP/6379 and memcached on port TCP/11211. The data is then aggregated on two levels: the Autonomous System (AS) and CC (Country Code). Thus, for each AS/CC we can estimate the number of Internet-wide reachable systems on a daily basis. This information is made publicly available each day. However, no historical summary of the key-value stores scans are provided.

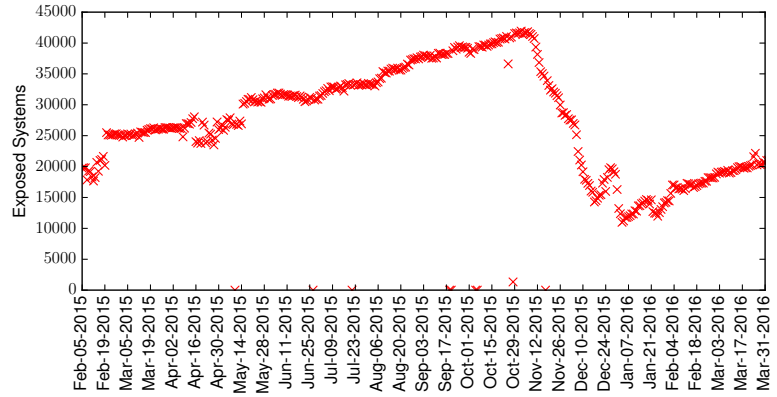
In late January 2015 the Shadowserver Foundation added Redis and memcached to their scanning project (ShadowServer Foundation, 2014). Other key-value stores's were added later, e.g., MongoDB mid-February 2015 and ElasticSearch in June 2015. We, for the purpose of this thesis, choose Redis and memcached as our research focus is on in-memory key-value stores. We have gathered a year-long dataset of all exposed Redis and memcached instances on the Internet, starting on the 5th of February 2015 using the Shadowserver Foundation's public data feed.

As the ShadowServer datasets are already highly aggregated we augment them with additional data. For this purpose, we perform our own targeted low bandwidth (5mbit) zMap (Durumeric et al., 2013) SYN only scan. But rather than targeting all ASes we focus on a subset of the ASes which we highlight in our case studies, see Section 6.5. To identify which prefixes to scan we use the HE BGP tool (Hurricane Electric, 2016) to identify all de-aggregated /24 prefixes an AS originates at the time of a scan.

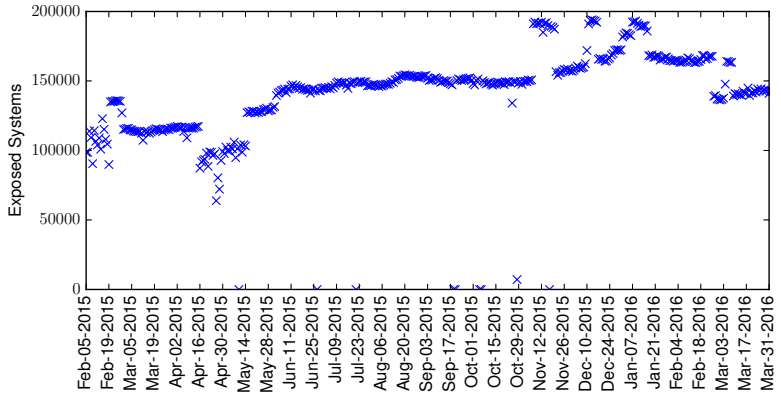
We find that the total number of responding systems in our scan is comparable to the numbers reported by the Shadowserver Foundation. Indeed, as these do match nearly exactly, we consider this as validation of both our own data as well as the data by the ShadowServer foundation. It is a validation of our own data as we do an SYN only scan. It is a validation of the data by the ShadowServer foundation as an independent scan yields similar results.

## 6.4 Observations

To get an overview of how the number of exposed Redis and memcached instances on the Internet evolved over time, Figure 6.1(a) and 6.1(b) show the number of exposed instances




(a) Redis




(b) Memcached

Figure 6.1: Redis and memcached: Number of exposed instances over time.

for each day. Surprisingly, given the severe security implications, we still see a steady increase in the number of exposed instances in both datasets. However, there is one major difference between the two key-value stores. In case of Redis the number of exposed instances is substantially reduced during the time period from early November 2015 to the beginning of January 2016.

Taking a closer look at the Redis dataset we identify three phases, see Figure 6.1(a). From the start of our observation period, Feb-05-2015, till about Nov-06-2015 we observe a steady increase of exposed instances. Here, the average increase of exposed instances is 62/434 per day/week. Starting from Nov-07-2015 to Jan-03-2016 we observe a huge drop by 30,575 exposed instances. The average decrease rate is 449 instances per day. This decrease is briefly interrupted by a short increase over the holiday season at the end of 2015. However, starting around Jan-04-2015 till the end of our observation period, Apr-01-2015,

| Start | End | Change | Contributors |
|---|---|---|---|
| Feb-27-2015 | Feb-28-2015 | -20,272 | $AS_{m_2}$ |
| May-30-2015 | Jun-03-2015 | +12,020 | $AS_{m_2}$ |
| Nov-06-2015 | Nov-07-2015 | +40,327 | $AS_{m_1}$, $AS_{m_2}$ |
| Nov-20-2015 | Nov-21-2015 | -30,632 | $AS_{m_1}$ |
| Dec-09-2015 | Dec-11-2015 | +28,615 | $AS_{m_1}$ |
| Dec-16-2015 | Dec-17-2015 | -26,598 | $AS_{m_1}$ |
| Dec-31-2015 | Jan-03-2016 | +12,186 | $AS_{m_1}$ |
| Jan-06-2016 | Jan-09-2016 | +10,586 | $AS_{m_1}$ |
| Jan-15-2016 | Jan-17-2016 | -21,344 | $AS_{m_2}$ |
| Feb-25-2016 | Feb-26-2016 | -28,796 | $AS_{m_1}$ |
| Mar-03-2016 | Mar-05-2016 | +26,517 | $AS_{m_1}$ |
| Mar-08-2016 | Mar-09-2016 | -24,175 | $AS_{m_1}$ |

Table 6.1: Memcached: Periods with significant change to $No.$ of exposed instances.

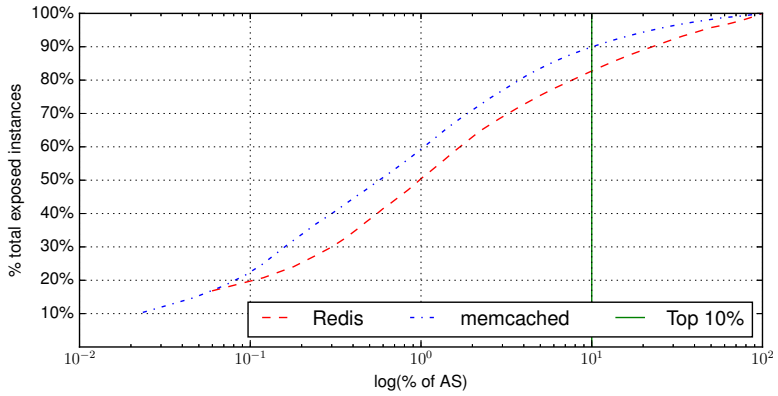



Figure 6.2: Redis and memcached: CDF per AS from Mar-14-2016.

the number of exposed instances is on the rise again with an average increase of 109 per day.

In the memcached dataset, see Figure 6.1(b), we do not see such a significant decrease in the number of exposed instances as for the Redis dataset. Here, the increase of exposed instances is roughly 170/1130 per day/week over the whole observation period. However, note that the magnitude of memcached instances is significantly larger with a maximum of almost 200,000 However, Figure 6.1(b) shows some strange behavior towards the end of 2015 and in March 2016. Indeed, Table 6.1 summarizes short time periods when the number of exposed systems changed by more than 9,000. In Section 6.5 we take a closer look at them.

Next, we explore how the number of exposed system is distributed across the autonomous systems. Overall, our dataset finds exposed systems in 4,705 / 7,714 ASes for Redis /

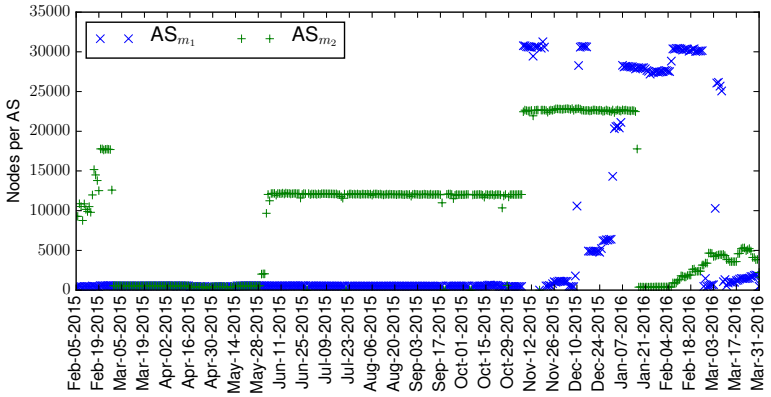



Figure 6.3: Memcached: Number of exposed instance in selected ASes over time.

memcached. This corresponds to a large fraction of total number of ASes. In addition, as one may expect, not all ASes are affected equally. Accordingly, Figure 6.2 shows for Mar-14-2016 the concentration of the number of exposed key-value stores per AS for both Redis and memcached using a cumulative distribution function (CDF) with a log-scale x-axis. To ease interpretation of the results we added a support line for the Top 10% of the ASes. For memcached they host 90% of the exposed instance while for Redis they only host 82% of exposed instances. The Top 1% contribute 60%/50% of all exposed memcached/Redis instances. For other days we find similar distributions.

We do see some evidence of irregularities in the two datasets: up to Feb-19-2015 and from Apr-09-2015 to Apr-14-2015. Moreover, there are a few days (less than ten) in which no or a very small number of exposed instances are reported for both Redis and Memcached. Since these are common to both datasets we presume that these are due to some problems with the Shadowserver Foundation's infrastructure and ignore those time periods. Indeed, these are the only irregularities that we find that are present in both datasets. While on first glance the data for Memcached in the period from Nov-06-2015 to Jan-15-2016 may look like an irregularity it is not as we discuss below.

## 6.5 Examples

Next, we take a look at the possible causes for significant changes in the number of affected systems for both Redis and memcached.

### 6.5.1 Redis Decrease/Reincrease

Recall, that Redis has an initial phase with steady increase in the number of exposed instances followed by a second phase of substantial decline which starts around Nov-07-2015 where the number of exposed instances platoons followed by few days of minor decline and then a sharp decline around Nov-12-2015 which leads to a reduction in the number of exposed instances by a factor of more than three within the next month.

On Oct-25-2015 Sanfilippo described how an exposed Redis server can be used to obtain remote shell access to a server (Sanfilippo, 2015), see Section 6.2. This story remained more or less dormant for the following nine days. On Nov-12-2015 various programming community-related news sites started to report on the possible exploit. Up to this time the increase of the exposed instances levels which indicate that some administrators already reacted based on the original blog post or an attack. Shortly afterwards, mainstream media picked up the issue. This corresponds to the sudden and forceful decline in the number of exposed Redis instances. Moreover, we find first evidence of the use of this exploit—stackoverflow posts (Upmanyu, 2015)—on Nov-13-2015. The decline is for Redis only and did not carry over to other key-value stores, e.g., memcached. Given that this specific attack only affects Redis this may not be surprising. However, given the other attacks pointed out in Section 6.2, this is a severe issue which should be approached by the responsible parties.

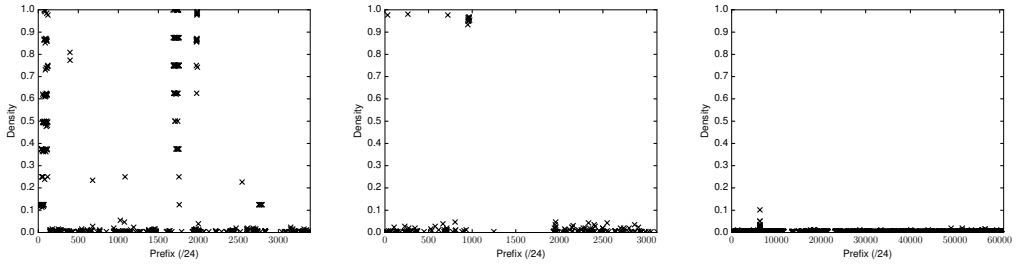
The decline lasted until early January 2016 with a brief interruption during the holiday season at the end of 2015. Our conjecture is that people used that time to evaluate new software stacks. As such, test systems were set-up during the holiday season and discontinued afterwards. If the project involved key-value stores Redis was a likely choice due to its previous prominence in the media.

Disturbingly, after the brief period of decrease, the number of exposed Redis instances rises at nearly twice the previous rate. This indicates that the incident awareness within the system administration community of a severe security problem is limited to roughly one month. Thus, continuous mitigation is severely lacking.

### 6.5.2 Memcached Events

As pointed out the number of exposed memcached systems is varying, see Table 6.1. To understand its root cause we use the distribution per AS and identified two major contributing ASes: $AS_{m_1}$ focuses on large web-hoster and also offers dedicated/virtual-private server provider; $AS_{m_2}$ is another large hoster. In contrast to $AS_{m_1}$ $AS_{m_2}$ focuses mainly on co-location as well as dedicated/virtual-private servers. For comparison, we also consider $AS_{m_3}$: A large end-user PaaS cloud network.

To get an overview of how the number of exposed memcached instances for $AS_{m_1}$ and $AS_{m_2}$ changes over time Figure 6.3 shows for each day the number of exposed instances by these ASes. By comparing Figure 6.3 with Figure 6.1(b) we find that both ASes are




(a) $AS_{m_1}$, Feb-23-2016 (b) $AS_{m_2}$, Mar-16-2016 (c) $AS_{m_3}$, Feb-23-2016

Figure 6.4: Memcached: Density of exposed instances in $AS_{m_1}$, $AS_{m_2}$ and $AS_{m_3}$.

responsible for most of the events of Table 6.1. One can see that most identified irregular events between Nov-2015 and Apr-2016 can be attributed to one of the two AS even given that $AS_{m_2}$ partly (de)amplified the observable effects. Therefore we label each of the events with the AS that is the major contributor in Table 6.1.

Given the large number of exposed systems and their sudden, apparently coordinated, appearance we hypothesize that they are, in fact, under a single administrative control. Here, administrative control is not meant in the sense of determining the BGP policies but rather the firewall and/or systems configuration of the exposed systems. In contrast to BGP in dedicated server setups the control of the systems and firewall level can be diversified up to a per-IPv4 address level.

Therefore, we check if the exposed systems are distributed evenly across the address range of the ASes or if they are clustered in specific prefixes. Using our zMap SYN scan for all prefixes originated by our three sample ASes we determine the density of the exposed systems per de-aggregated /24 prefixes originated by that AS. A density of 1 corresponds to an exposed memcached service on all IPv4 addresses within the /24 prefix. A density of 0 corresponds to no exposed memcached services.

Figure 6.4 shows the densities for our three ASes for Feb-23-2016 respectively Mar-16-2016. We picked Feb-23-2016 for $AS_{m_1}$ and $AS_{m_3}$ as there are a lot of exposed memcached instances within $AS_{m_1}$. We picked Mar-16-2016 for $AS_{m_2}$ as the number of exposed memcached instances with $AS_{m_2}$ is again around 5,000 instances[1]. $AS_{m_3}$, a large end-user PaaS cloud network, serves as baseline and hosts more than 4,000 exposed memcached instances. The density plot for February, Figure 6.4(c), and March, not shown, for $AS_{m_3}$ shows that the exposed instances are more or less evenly distributed across the address space of the AS.

For $AS_{m_1}$ and $AS_{m_2}$, see Figures 6.4(a) and 6.4(b), the densities support our hypothesis that here the exposed instances are under common administrative control. Indeed, for $AS_{m_1}$ more than hundred adjacent /24s have significant densities. Indeed, these densities vary in

[1] Unfortunately, we started this kind of detailed analysis too late to get insights into the events before Feb-2016.

multiples of 32 IPv4 addresses. It appears the allocation of addresses is done en-block on /28 granularity.

We contacted the operator of $AS_{m_1}$ in mid Feb-2016. After some discussion, the operator started to mitigate the issue. Mitigation was completed by end of February. Surprisingly, our dataset shows that the problem re-surfaced at the beginning of March. This time, the operator mitigated the problem without further contact.

$AS_{m_2}$ is responsible for the first large drop event in Feb-2015 of the exposed memcached instances. It re-introduced a large number of exposed memcached instances in the beginning of Jun-2015. This event was, however, counteracted by a simultaneous decrease in another AS and does, therefore, not significantly stick out in Figure 6.1(b). Its instances further increase at the same time when $AS_{m_1}$ first exposed its large number of instances. However, in contrast to $AS_{m_1}$, $AS_{m_2}$ mitigated the issue for nearly all exposed instances by mid January 2016. For $AS_{m_2}$, see Figure 6.4(b), we find 56 different /24 with a density higher than 0.9. This indicates that the systems in these subnets are under common administrative control.

Background checks showed that the corresponding network segments are most probably rented out. While $AS_{m_2}$ already has a somewhat questionable reputation due to being regularly listed on various blacklists, these third parties are commonly known for shady practices. Indeed, all affected prefixes are on the Spamhaus blacklist for use in so-called snowshoe spamming. In snowshoe spamming, spammers reserve large prefix sets to send spam from all addresses within the network to bypass spam filtering (Eeten et al., 2010). Overall, we conclude that the system(s) behind these addresses are most likely under the control of a sub-sub-customer of $AS_{m_2}$.

While we cannot per-se generalize to earlier events in $AS_{m_2}$ it may be possible to cross-check it with previous spam-campaigns which is beyond the scope of this paper. Nevertheless, from these exposed memcached instances we could have gathered internal information from these large, commercial spammers, if we had ignored ethical and legal considerations.

In principle, we can turn this observation around. By looking for prefixes with high densities of exposed memcached instances and their reverse DNS entries we can identify systems that are misconfigured due to a common root cause and are likely under one administrative control. Thus, as long as the spammers do not fix their memcached deployment we should be able to track them even if they move across the Internet or change their incorporated affiliation. Moreover, it may be possible to use related historic datasets to better understand spammers.

### 6.5.3 Redis Anomalies

Next, we use the concept of prefixes with high-density of exposed instances per /24 to revisit the Redis data. Among the identified ASes with such prefixes are two relatively

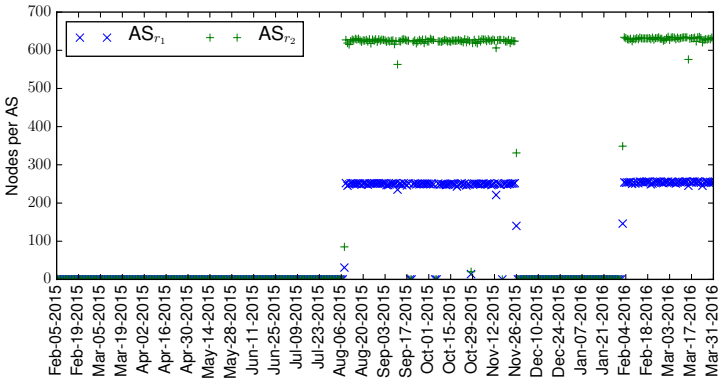



Figure 6.5: Redis: Timeseries for $AS_{r_1}$ and $AS_{r_2}$.

small ASes: $AS_{r_1}$ is registered in Azerbaijan and announces one /21 and $AS_{r_2}$ is registered in Kazakhstan which announces one /22 and one /20. Indeed, a detailed look at these ASes across time, Figure 6.5, shows the simultaneous sudden appearance and disappearance of exposed Redis instances.

Here, the most striking observation is that the appearance / disappearance of exposed key-value stores in these two ASes fully coincides. This indicates a shared administrative body. Indeed, the RIPE database entries for the two ASes list the same MNT-BY. This maintainer object belongs to a hoster that advertises that abuse complains are treated in a relaxed manner. Again, we have leads that may allow us to identify and link shady ASes via a shared misconfiguration pattern.

## 6.6 Summary

In this chapter we take a closer look at the protection of two popular in-memory key-value stores, Redis and memcached. Both are designed to be deployed in isolated network segments. If exposed to the Internet such system pose significant dangers to the data within them. In addition to the general abuse potential and the Redis remote shell exploits, we document so far unknown transitive attacks.

Using a year-long scan dataset, we point out that there is an alarming and ever increasing number of exposed Redis as well as memcached systems on the Internet. Moreover, even significant security problems only lead to a temporary, even if significant, reduction in the number of effected systems. Unfortunately, the memory of the systems administration community seems to be less than a few months and even ready reports, such as those by the Shadowserver foundation do not seem to have any lasting impact in reducing them.

Moreover, as a significant number of these key-value stores systems are hosted in cloud environments individual as well as organization misconfigurations can drastically increase

the number of exposed systems. We highlight this using exposed memcached instances in two ASes as examples. Indeed, such misconfigurations if cross-correlated with reverse DNS may even enable us to track spammers. Moreover, we show that it is possible to identify underlying organizational structures with concurrent misconfiguration patterns. We use Redis with two ASes as example to link two ASes with questionable reputation to a common operator.

Furthermore, we use this case study to investigate how misconfigurations are mitigated. For our case study we use scan-data that is provided on a daily basis by an NGO (Non Government Organization) to operators world wide. Our case study demonstrates, that directly approaching affected parties does help in mitigating specific misconfigurations with one operator. In contrast, we do see a huge impact of specific security incidents with high press coverage. In our example case, the publication of an exploit that tainted not only confidentiality and availability but also the integrity of dependent systems, lead to a severe decrease in exposed systems.

# Part III

# Scanning for Misconfigurations in IPv6

# 7

# Existing Approaches for IPv6 Security Scans

In the last part of this thesis, we took a look at scanning for misconfigurations on the Internet focusing on IPv4 connected hosts. However, IPv6 currently gains more and more traction (Plonka and Berger, 2015). Hence, it is imperative to also investigate security misconfigurations on the IPv6 Internet. However, due to the $2^{128}$ possible IPv6 addresses which could—theoretically—be allocated, the classical discovery approaches as outlined in Chapter 5 do not scale to IPv6 and alternative methods are needed.

The most promising approaches use datasets of IPv6 addresses to pre-determine scanning targets. In this chapter, we introduce the most commonly used datasets. Furthermore, we will reflect on the properties of these datasets, and how they influence their applicability for large-scale scans for IPv6 security misconfigurations.

## 7.1 Properties of Datasets

In the remainder of this chapter, we compare datasets along the following three properties.

**Dataset Composition:** For each dataset, we evaluate which type of host they mostly cover. We quantify this by assessing if a dataset excludes certain types of hosts based on the way the dataset is gathered. For the purpose of scanning for security misconfigurations—following our definition of security misconfigurations in Chapter 2—we want to use datasets that mainly include servers and on-path network infrastructure devices rather than clients.

**Public Availability:** Especially in Internet measurement the availability of datasets is a constant challenge (Alllman and Paxson, 2007). Researchers may be unable to publish datasets used in studies due to privacy concerns, or because the dataset contains proprietary information, e.g., on a company's performance. However, for the purpose of scanning for security misconfigurations, the dataset has to be available to security researchers, ideally *without* the need to have access to specific vantage points.

**Dataset Size:** In addition to its general availability, a dataset for IPv6 security scans should be as large as possible. This aspect is straight forward: The more uniquely allocated IPv6 addresses a dataset contains, the better.

Next, we will introduce various datasets which have been used in the Internet measurement and IT security literature to observe and measure the IPv6 Internet.

## 7.2 Proprietary Datasets

### 7.2.1 IXP Dataset

Using Internet eXchange Points (IXPs) as datasources was first introduced by Chatzis et al. (Chatzis et al., 2013). More recently, Gasser et al. (Gasser et al., 2016) use a major European IXP to collect IPv6 datasets for scanning. They were able to collect over 79M unique IPv6 addresses on their vantage point. However, due to the nature of IXPs, their dataset is focused on client and server systems. On-path network equipment can not be discovered, as the information extracted from the IXP only contains IPv6 addresses of the two end-hosts of a given connection (Chatzis et al., 2013; Gasser et al., 2016). However, due to the significant privacy implications of this dataset, it is not available to the general public.

### 7.2.2 CDN Dataset

The Content Delivery Network (CDN) dataset has been used for several recent studies on IPv6 (Foremski et al., 2016; Plonka and Berger, 2015). The dataset is collected from the access logs of a major CDN. With over 1.85B IPv6 addresses, this dataset dwarfs all other contemporary datasets in size. However, again due to privacy implications, live access to the dataset is not possible. Furthermore, due to CDNs being almost exclusively contacted by clients, the CDN dataset rarely contains on-path network devices and server systems.

## 7.3 Public Datasets

### 7.3.1 Traceroute Dataset

The CAIDA traceroute dataset (Luckie et al., 2008) has been created for topology discovery. The dataset is created by tracerouting various destinations from a large set of probes scattered over the Internet. From these traceroutes, the IPv6 addresses of intermediate hops can be collected. The dataset has the advantage that it is published by the authors, and can also be collected by independent researchers. However, as it uses traceroutes to discover network paths, it is mostly composed of on-path network equipment's IPv6 addresses. Furthermore, with a size of around 110k unique addresses at the time of writing, this dataset is relatively small.

| Source | Systems | | | Size | Public | Citation |
|---|---|---|---|---|---|---|
| | Clients | Net.-Devs. | Servers | | | |
| IXP | ✓ | | ✓ | 79M | | (Gasser et al., 2016) |
| CDN | ✓ | | | ≥1.85B | | (Plonka and Berger, 2015) |
| CAIDA TR | | ✓ | | 110k | ✓ | (Luckie et al., 2008) |
| IPv4 rDNS | | ✓ | ✓ | 545k | ✓ | (Czyz, Luckie, et al., 2016) |

Table 7.1: Overview of example datasets' features.

### 7.3.2 IPv4 Reverse DNS Forward Confirmation Dataset

Another method to collect an IPv6 address dataset has been documented by Czyz et al. (Czyz, Luckie, et al., 2016). They brute-force resolve all possible IPv4 reverse DNS (rDNS) entries and then perform AAAA forward lookups for the associated IPv6 address on them. Due to this collection methodology, the dataset is mostly comprised of on-path networking equipment and server systems. Furthermore, the technique is well documented and the datasource the publicly queryable DNS. Still, Czyz et al. only report on finding 545k addresses with this technique.

## 7.4 Other Datasets

Apart from those datasets discussed in this section, other datasets may be applicable. Examples include the Rapid7 ANY dataset, which could also be collected by individual researchers, and the Farsight DNS log dataset. However, to the best of our knowledge, neither of these two has been studied with respect to the contained IPv6 addresses'.

## 7.5 Summary

Given the set of available datasets, see Table 7.1, we find that only one of those datasets fulfills the requirements we set earlier: A focus on servers/network devices and public availability. However, the IPv4 rDNS dataset used by Czyz et al. is comparatively small. It contains less than a million hosts, and is unlikely to be representative of the Internet as a whole. This opens the challenge to find ways to gather large datasets which are publicly accessible.

# 8

# Global Collection of IPv6 Scan-Targets From DNS

As we document in Part II of this thesis, the mitigation of Internet-wide security sensitive misconfiguration heavily relies on the availability of exhaustive security scans. This is a solved challenge for IPv4, see Chapter 5, where a brute-force approach is feasible. However, in IPv6 such a brute-force approach is not feasible due to the large address space of IPv6 in comparison to IPv4. Instead, possible targets have to be identified before the security scan. Current approaches for the large-scale identification of IPv6 targets depend on restricted and proprietary datasets. The datasets that are available are either small or domain specific, e.g., traceroute-based datasets are biased toward network equipment, see Chapter 7.

However, as seen in Chapter 5 and 6, security researchers require *publicly accessible* datasets which cover systems used for running *Internet services*, not network equipment or clients. Thus, there is demand for techniques to gather public IPv6 datasets. Furthermore, a review of the literature already indicates various promising techniques (Gont and Chown, 2016), which however lack global applicability. Hence, in this chapter, we present our adjustments to a methodology for collecting IPv6 address datasets for subsequent scans, which make them globally applicable. Contrary to prior work, our methodology does not require access to restricted network vantage points.

Even though the core technique, enumerating the IPv6 reverse zone through DNS' denial of existence semantics (NXDOMAIN), has been around for some months (Gont and Chown, 2016), we encountered several obstacles in using this technique on a global scale. In this chapter, we present how we have successfully overcome these obstacles and how we collected a new dataset spanning more than 5.8 million IPv6 addresses. Subsequently, these can be used for large-scale IPv6 security scans.

To demonstrate the feasibility of our approach, we include three case-studies in this chapter. The first one deals with investigating the infrastructure of a large Software as a Service (SaaS) provider. Our technique permits us insights into the way the operator organizes its infrastructure. Indeed, based on our data, we are able to determine that core-operations in the network are performed manually, and, hence, are prone to errors. In addition, we investigate the security impact of our technique itself, as well as security scans that evaluate hosts enumerated with our technique. Our technique permits us to investigate well-firewalled networks, that are hidden from traditional scan techniques. Herein, we focus on how this

information provides insights on the software stacks used by the associated organizations. Last, using security scans of IPv6 networks, we also investigate a so far unknown security misconfiguration. We find that large Internet backbone routers of a major vendor exposes internal backplane communication APIs to the Internet exclusively via IPv6. Note, that this may be the first misconfiguration that is exclusive to IPv6 connected systems.

## 8.1 Motivation

The adoption of IPv6 has been steadily increasing in recent years (Czyz et al., 2014). Unsurprisingly, simultaneously, the research question of efficiently identifying allocated and active IPv6 addresses in the wild has received more and more attention from the scientific community. However, unfortunately for the common researcher these studies have—so far—been dominated by the analysis of large, restricted, and proprietary datasets. For instance, the well-known content delivery network (CDN) dataset used for most contemporary IPv6 analyses (Foremski et al., 2016; Plonka and Berger, 2015), Internet exchange point (IXP) datasets which were used regularly by some other research groups (Chatzis et al., 2013; Gasser et al., 2016), or, slightly less restrictive, the Farsight DNS recursor dataset (Vixie, 2016). Although public datasets do exist, they are traceroute-based datasets from various sources, including the RIPE Atlas project (Ripe NCC, 2016a), which are limited due to their nature: they are biased towards addresses of networking equipment, and, in turn, bear their own set of problems for meaningful analyses.

Correspondingly, in this chapter, we aim to address the problem of obtaining a dataset of currently-allocated IPv6 addresses for subsequent security scanning that is accessible to the common researcher: We present a new methodology with which we were able to collect more than 5.8 million unique IPv6 addresses that can be employed by every researcher with network access. The underlying concept is the enumeration of IPv6 reverse zones (PTR) leveraging the semantics of DNS' denial of existence records (NXDOMAIN). Although the general concept has been discussed in RFC 7707 (Gont and Chown, 2016), we identified and overcame various challenges that prevent the use of this technique on a global scale. Therefore, we document how we can leverage the semantics of NXDOMAIN on a global scale to collect allocated IPv6 addresses for a new IPv6 dataset. Our detailed algorithmic documentation allows researchers everywhere to implement this technique, reproduce our results, and collect similar datasets for their own research.

## 8.2 Background on Reverse DNS

Following, we briefly introduce rDNS, the most common method to resolve an IP address to a name, which follows the concept that every host reachable on the Internet should have a name (Barr, 1996). In practice, rDNS is implemented in the DNS ecosystem using pointer (PTR) resource records (RR) (Mockapetris, 1987a). These records are assigned to names

under designated zones in-addr.arpa for IPv4 and ip6.arpa for IPv6 (Mockapetris, 1987a; Thomson et al., 2003). The specific DNS name under these zones is directly derived from the IP address, and can be queried for the PTR record.

**Reverse DNS in Practice** At almost all points where IP addresses are displayed to users, rDNS can be used instead, to provide more information on the address, a practice that is especially important for large networks and for human-readable log files (Nicholas and Huntington, 2003). For example, it is common practice for Secure Shell (SSH) servers to look up the FQDN of a connecting host to include it in the authentication logs it generates. Similarly, many intrusion detection systems (IDS) perform rDNS lookups to enrich their logs.

Furthermore, reverse DNS is also used by email servers to mitigate spam: mail exchange (MX) servers verify that the rDNS record of a connecting client IP matches the client's HELO string and that the corresponding forward record (A or AAAA) resolves to the IP address again (Cormack, 2007). Similarly, it is considered best practice to indicate the purpose of exhaustive Internet security scans in the rDNS entries of the scanning systems (Durumeric et al., 2013). Operators might also encode information on the logical and physical parameters of IP addresses into rDNS entries. For example, it might include information on the physical interface the address is on, the location of the device, the purpose of the link the address is on, or, data on the software versions and function of the device (Fiebig, Borgolte, et al., 2017). Especially due to such data being encoded, rDNS has become an important tool for researchers to investigate network topologies (R. V. Oliveira et al., 2008; M. Zhang et al., 2006).

**IPv4 Reverse DNS: in-addr.arpa** IPv4 rDNS is organized under in-addr.arpa (Mockapetris, 1987a). An IPv4 address consists of four octets, represented by decimal numbers delimited by dots. To generate the rDNS name for any given address, e.g., 198.51.100.23, the address is reversed at a per-octet level and suffixed with in-addr.arpa. For example, the rDNS record to look up the name for 198.51.100.23 is 23.100.51.198.in-addr.arpa. This octet-level separation means that the respective DNS zones are split at the octet-level as well and can only be delegated with that granularity.

Unfortunately, this approach caters toward the requirements of classful interdomain routing that split networks at octet borders (Fuller et al., 1992). However, classfull routing has been obsoleted (Rekhter, 1995) and replaced with classless interdomain routing (CIDR) (Rekhter and T. Li, 1993). In turn, for IPv4 rDNS it introduced various problems. Specifically, CIDR allows networks to be split outside of octet borders, including into smaller networks than the smallest delegatable rDNS zone size (/24). The network 198.51.100.16/28, which includes 198.51.100.16 to 198.51.100.31, is an example for this. For this network, no rDNS zone small enough exists that could be delegated to the end-user. To work around this issue, RFC2317 (Eidnes et al., 1998) suggests to use CNAME records to delegate single pointer RRs from the in-addr.arpa zone to a domain controlled by or delegated to the end-user.

**IPv6 Reverse DNS: ip6.arpa** The rDNS zone for IPv6 is ip6.arpa (Thomson et al., 2003). In general, IPv6 addresses are represented by up to 32 nibbles, hexadecimal digits (0-f), grouped in blocks of four nibbles each. For improve readability, leading zeros may be omitted in each group when displaying IPv6 addresses. Furthermore, the longest continuous set of groups filled only with zeros may be replaced with a double colon (::) (Hinden and Deering, 2006).

The experiences gathered with IPv4 rDNS lead to a slightly different and improved approach for rDNS in IPv6 (Thomson et al., 2003): The IPv6 rDNS zone is split at the nibble-boundary. Effectively, the reverse name for a given IPv6 address, e.g., 2001:db8:2342:-4567:89ac:23::1 corresponds to the ip6.arpa name 1.0.0.0.0.0.0.0.3.2.0.0.c.a.9.8.7.6.5.4.2-.4.3.2.8.b.d.0.1.0.0.2.ip6.arpa

The split at the nibble-boundary has the advantage that rDNS zones can be delegated with a significantly finer granularity than IPv4 reverse zones. Specifically, the smallest delegatable zone for IPv6 rDNS contains only 16 addresses.

## 8.3 DNS Enumeration Techniques

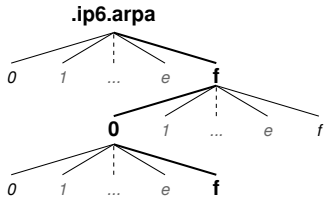



Figure 8.1: Enumerating **f.0.f.ip6.arpa.**

Complimentary to prior approaches, van Dijk proposed enumerating IPv6 reverse records by utilizing the specific semantics of denial of existence records (NXDOMAIN) (Bortzmeyer and Huque, 2016; Gont and Chown, 2016): When correctly implementing RFC1034 (Mockapetris, 1987a), as clarified in RFC8020 (Bortzmeyer and Huque, 2016), the Name Error response code (NXDOMAIN in practice) has the semantic of *there is nothing **here** or anywhere **thereunder** in the name tree*. Making this notion explicit in RFC8020 (Bortzmeyer and Huque, 2016) is a relatively recent development. Combined with the IPv6 PTR DNS tree, where each sub-zone has 16 (0-f, one for each IPv6 nibble) children up to a depth of 32 levels, provides the possibility to exploit standard-compliant nameservers to enumerate the zone.

Specifically: Starting at the root (or any other known subtree), a request for each of the possible child nodes is performed. If the authoritative server returns NXDOMAIN, the entire possible subtree can be ignored, as it indicates that no entries below the queried node exist. Algorithm 1 shows the corresponding algorithmic description. Figure 8.1 provides a simplified visualization, e.g., if a queries for *0-e.ip6.arpa.* return NXDOMAIN, but *f.ip6.arpa.* returns NOERROR, we can ignore these subtrees, and continue at *f.ip6.arpa.*, finally finding *f.0.f.ip6.arpa.* as the only existing record.

Algorithm 1: Algorithm for iterating over ip6.arpa., based on RFC7707 (Gont and Chown, 2016).

```
// Base-Case: max.ip6.arpa.len = 128/4 * 2 + len("ip6.arpa.");
Function enumerate(base, records={ }, max.ip6.arpa.len)
    for i in 0..f do
        newbase ← i+"."+base;
        qryresult ← getptr(newbase);
        if qryresult != NXDOMAIN then
            if len(newbase) == max.ip6.arpa.len then
                add(records, newbase);
            else
                enumerate(newbase,records,max.ip6.arpa.len);
```

## 8.4 Methodology and Algorithmic Implementation

The approach outlined in Section 8.3 has been used on small scales in the past: Foremski et al. (Foremski et al., 2016) used it to collect a sample of 30,000 records from selected networks for their study. In this section, we analyze the challenges of a global application of the technique and describe how we can overcome these limitations.

### 8.4.1 Non RFC8020-Compliant Systems

The current technique requires that RFC8020 (Bortzmeyer and Huque, 2016) is correctly implemented, i.e., that the nameserver behaves standard-compliant. However, following RFC7707 (Gont and Chown, 2016), this is not the case for all authoritative DNS nameserver software found in the wild (Bortzmeyer and Huque, 2016). Specifically, if higher level servers (from a DNS tree point of view) are not enumerable by any of the presented techniques, then this can mask the enumerable zones below them. For example, if a regional network registry, like APNIC or, RIPE would use a DNS server that cannot be exploited to enumerate the zone, then all networks for which they delegate the reverse zones would become *invisible* to our methodology.

To approach this challenge, we seed the algorithm with potentially valid bases, i.e., known to exist *ip6.arpa.* zones. Our implementation obtains the most recent Routeviews (University of Oregon, 2016), and the latest RIPE Routing Information Service (RIS) (Ripe NCC, 2016b) Border Gateway Protocol (BGP) tables as a source. Particularly important to allow the approach to be easily reproducible: both are public BGP view datasets, available to any researcher.

Based on the data, we create a collapsed list of prefixes. Following prior work, we consider the generated list a valid view on the Global Routing Table (GRT) (B. Zhang et al., 2005). For each of the collapsed prefixes we calculate the corresponding ip6.arpa. DNS record. The resulting list is then used as the input seed for our algorithm. Alternative public seed datasets are the Alexa Top 1,000,000 (Czyz et al., 2014; Czyz, Luckie, et al., 2016) or

traceroute datasets (Foremski et al., 2016) (which, as aforementioned, are biased by nature; thus, special care must be taken for traceroute datasets). If available, other non-public datasets like the Farsight DNS recursor dataset (Vixie, 2016) could also be used.

Complimentary approaches to collect ip6.arpa. addresses or subtrees from systems that implement RFC8020 incorrectly are those with which one can obtain (part of) a DNS zone. For example, by employing insufficiently protected domain transfers (AXFRs), which are a prominent misconfiguration of authoritative nameservers (Atkins and Austein, 2004).

### 8.4.2 Breadth-First vs. Depth-First Enumeration

For our data collection, we employ Algorithm 1. Unfortunately, the algorithm leverages depth-first search to explore the ip6.arpa. tree. This search strategy becomes problematic if any of the earlier subtrees is either rather full (non-sparse) or if the authoritative nameservers are relatively slow to respond to our queries. Slow responses are particularly problematic: they allow an "early" subtree to delay the address collection process significantly.

Substituting depth-first search with breadth-first search is non-trivial unfortunately. Therefore, we integrate features of breadth-first search into the depth-first algorithm (Algorithm 1), which requires a multi-step approach: Starting from the seed set, we first use Algorithm 1 to enumerate valid ip6.arpa. zones below the records up to a corresponding prefix-length of 32 bits. If we encounter input-records that are more specific than 32 bits, we add the input record and the input record's 32-bit prefix to the result set. Once this step has completed for all input records, we conduct the same process on the result set, but with a maximum prefix-length of 48 bits, followed by one more iteration for 64-bit prefixes. We opted to use 64 bits as the smallest aggregation step because it is the commonly suggested smallest allocation size and designated network size for user networks (Hinden and Deering, 2006). Algorithm 2 provides a brief description of the cook_down algorithm. The last step uses Algorithm 1 on these /64 networks with a target prefix size of 128 bits, effectively enumerating full ip6.arpa. zones up to their leaf nodes.

To not overload a single authoritative server, the ip6.arpa. record sets are sorted by the least significant nibble of the corresponding IPv6 address first before they are further enumerated. Sorting them by the least significant nibble spreads zones with the same significant bits as broadly as possible.

Combined with the observed low overall traffic that our modified technique generates, we can prevent generating unreasonably high load on single authoritative nameserver. Our approach, contrary to prior work, does not generates high load on the authoritative nameservers before moving on to the next one. Otherwise it would launch a denial of service attack against the nameserver. If our approach is more widely adopted by researchers, future work should investigate how distributed load patterns can be prevented, i.e., thousands of researchers querying the same nameserver simultaneously (see Section 8.4.5).

Algorithm 2: Algorithm cooking down the initial seed records.

```
Function cook_down (records)
    for prefix.len in 32,48,64 do
        records.new ← { };
        cur.ip6.arpa.len ← prefix.len/4 * 2 + len("ip6.arpa.");
        for base in records do
            // See Section 8.4.3 Dynamically-generated Zones/Prefix Exclusion/Opt-Out for details;
            if checks(base) == False then
                pass
            else if len(base) ≥ cur.ip6.arpa.len then
                add(records.new, base);
                crop.base = croptolength(base,cur.ip6.arpa.len);
                add(records.new, crop.base);
            else
                add(records.new, enumerate(base, cur.ip6.arpa.len));
```

### 8.4.3 Detecting Dynamically-generated Zones

Dynamically generating the reverse IP address zone, i.e., creating a PTR record just-in-time when it is requested, has been popular in the IPv4 world for some time (Richter, Smaragdakis, et al., 2016). Unsurprisingly, utilizing dynamically generated IPv6 reverse zones has become even more common over time as well. Especially access networks tend to utilize dynamically-generated reverse records. While this provides a significant ease-of-use to the network operators, our algorithm will try to fully enumerate the respective subtrees. For a single dynamically-generated /64 network it leads to $2^{64}$ records to explore, which is clearly impractical. Therefore, we introduce a heuristic to detect if a zone is dynamically-generated, so that we can take appropriate action.

To detect dynamically-generated reverse zones, we can rely on the semantic properties of reverse zones. The first heuristic that we use is the repeatability of returned FQDNs. Techniques for dynamically-generated reverse zones usually aim at providing either the same or similar fully-qualified domain names (FQDNs) for the reverse PTR records. For the former detection is trivial. In the latter case, one often finds the IPv6 address encoded in the returned FQDN. In turn, two or more subsequent records in an dynamically generated reverse zone file should only differ by a few characters. Therefore, a viable solution to evaluate if a zone is dynamically-generated is the Damerau-Levenshtein distance (DLD) (Fiebig, Danisevskis, et al., 2014).

Unfortunately, we encountered various cases where such a simplistic view is insufficient in practice. For instance, zones may also be dynamically-generated to facilitate covert channels via DNS tunneling (Nussbaum et al., 2009). In that case, the returned FQDNs appear random. Similarly in other cases, the source record or IPv6 address are hashed, and then incorporated into the reverse record. In those cases the change between two records can be as high as the full hash-length of the utilized hash digest.

We devised another heuristic based on the assumption that if a zone is dynamically-generated, then all records in the zone should be present. Following prior work by Plonka et al. and Foremski et al. (Foremski et al., 2016; Plonka and Berger, 2015), we determined that certain records are extremely unlikely to exist in one zone all together. Specifically these are all possible terminal records of a base that utilize only one character repeatedly. For example, for the base *0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.e.f.ip6.arpa* such a record would be *f.f.f.f.f.f.f.f.f.f.f.f.f.f.f.f.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.e.f.ip6.arpa*. Only random records would be slightly less like to exist, but would also make our algorithm more complex, due to depending on entropy for randomness. Therefore, we build and query all sixteen possible records from the character set 0..f. Due to these records being highly unlikely (Foremski et al., 2016), and the use of packet-loss sensitive UDP throughout DNS, we require only three records to resolve within a one second timeout to classify a zone as dynamically-generated. We omit the heuristic's algorithmic description for brevity, as the implementation is straight forward.

### 8.4.4 Prefix Exclusion

Naturally, in addition to excluding dynamically-generated zones, a network operator may ask to be excluded from her networks being scanned. During our evaluation, multiple network operators requested being excluded from our scans. Furthermore, we blacklisted two network operators that did use dynamically-generated zones, but for which our heuristic did not trigger, either due to rate-limiting of our requests on their side, or bad connectivity toward their infrastructure. Similarly, our algorithm missed a case for a US Based University which used /96 network access allocations, which we did not detect as dynamically-generated due to the preselected step-sizes for Algorithm 2. In total, we blacklisted five ISPs' networks and one University network.

### 8.4.5 Ethical Considerations and Opt-Out Standard

To encourage best practice, for our experiments and evaluation, the outbound throughput was always limited to a maximum of 10 MBit/s in total and specifically to 2MBit/s for any single target system at a time following our least-significant byte sorting for ip6.arpa zones. Although the load we incurred was negligible for the vast majority of authoritative nameservers, we acknowledge that the load this methodology may put onto authoritative servers may become severe, particularly if more researchers utilize the same approach simultaneously or do not limit their outbound throughput. Hence, we suggest to adopt and communicate the practice of first checking for the existence of a PTR record in the form of *4.4.4.f.4.e.5.4.5.3.4.3.4.1.4.e. ... .ip6.arpa.*. The respective IPv6 record encodes the ASCII representation of DONTSCAN for /64 networks. For networks larger than /64, we suggest to repeat the string. We do not use a non-PTR conform record, as this would exclude users utilizing, e.g., restrictive DNS zone administration software possibly sanitizing input. We will carry this proposal toward the relevant industry bodies, to provide operators a simple method to opt out of scans.

Algorithm 3: Call-order in final script.

```
seeds ← get_seeds();
enum.records ← cook_down(seeds);
final.result ← { };
for base in enum.records do
    // See Section 8.4.3 Dynamically-generated Zones/Prefix Exclusion/Opt-Out for details;
    if checks(base) == False then
        return { } ;
    tmp.results ← enumerate(base, 128);
    final.result ← final.result + tmp.results;
```

### 8.4.6 CNAMEs

We note that our investigation later revealed cases in which we discovered empty terminals in the DNS tree, i.e., records of 32 nibble length without an associated PTR resource record that do not return NXDOMAIN. Upon removing of these records, and by focusing on non-empty terminals in these address bases, we still obtain valid results. In addition to cases where the terminals are fully empty, CNAME records (Mockapetris, 1987b) may exist instead of PTR records, which is why it is necessary to resolve CNAME records if a PTR record does not exist.

### 8.4.7 Parallelization

Combining the previously presented algorithms, we can enumerate the IPv6 PTR space (see Algorithm 3). Due to our algorithm's nature, parallelization is ideally introduced in the *for* loop starting at line 5 of Algorithm 2 and the *for* loop at line 4 in Algorithm 3. Technically, it would also be possible to introduce parallelization in the first *for* loop of Algorithm 1. However, then parallelization would be performed over a single subnetwork, for which a single authoritative nameserver might be responsible. In turn, parallelization at this point would result in an implicit denial of service attack on the authoritative nameserver (see the paragraph on ethics). By parallelizing our approach through Algorithm 2 and Algorithm 3 parallel queries are made for different IPv6 networks, thus, most likely to different authoritative servers.

## 8.5 Evaluation

We evaluate our methodology on a single machine running Scientific Linux 6.7 with the following hardware specification: four Intel Xeon E7-4870 CPUs (2.4GHz each) for a total of 80 logical cores, 512GB of main memory, and 2TB of hard-disk capacity. We installed a local recursive DNS resolver (Unbound 1.5.1) against which we perform all DNS queries. Connection-tracking has been disabled for all DNS related packets on this machine, as well

| | | ip6.arpa. | GRT_SEED$_{80}$ | GRT_SEED$_{400}$ | $\sum$ |
|---|---|---|---|---|---|
| Runtime (minutes) | /32 | 120 | 7 | 7 | |
| | /48 | 130 | 232 | 144 | |
| | /64 | 429 | 1,040 | 404 | |
| | Full | 3,244 | 2,956 | 775 | |
| | $\sum$ | 3,932 | 4,235 | 1,330 | |
| Records Found | Seed | / | 72k | 72k | 73k |
| | /32 | 3.5k | 73k | 73k | 75k |
| | /48 | 52.5k | 856k | 834k | 895k |
| | /64 | 1M | 582k | 1.4M | 2.2M |
| Addresses Found | | 1.6M | 5.3M | 2.2M | 5.8M |
| | Unique | 335k | 2.8M | 33k | |
| Queries | /32 | 46.5k | 21.5k | 22k | |
| | /48 | 380k | 2.1M | 2M | |
| | /64 | 3M | 27.2M | 27.4M | |
| | Full | 59M | 192M | 161.2M | |
| | $\sum$ | 62M | 221.3M | 190.7M | |
| Dynamic Zones | /32 | 615 | 1.5k | 1.5k | 1.5k |
| | /48 | 15k | 716k | 690k | 732k |
| | /64 | 223k | 80.5k | 796k | 1M |
| Blacklisted Zones | /32 | 0 | 713 | 715 | 715 |
| | /48 | 1.5k | 63 | 65 | 1.6k |

Table 8.1: Overview of the results of our evaluation.

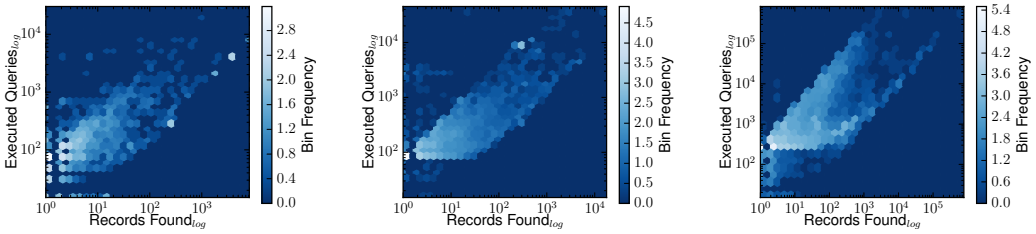
as other upstream-routers for DNS traffic from this machine. An overview of our results can be found in Table 8.1.

### 8.5.1 Enumerating ip6.arpa.

In our first evaluation scenario, we enumerate addresses using the PTR zone root node of .ip6.arpa. as the initial input only, which will serve as basic ground-truth. The respective dataset corresponds to the first column of Table 8.1: ip6.arpa. The enumeration was completed within 65.6 hours, of which most time was spent enumerating pre-identified /64s networks. As such, the impact of dynamic-generation is evident from this experiment: More than 615 /32 prefixes are ignored due to dynamically-generated PTR records, with an additional 15,000 /48 prefixes and more than 16,000 /64 networks subsequently. This ground-truth experiment yields a total of 1.6 million allocated IPv6 addresses.

### 8.5.2 GRT_SEED$_{80}$: Seeded Enumeration (80 Threads)

For our second experiment, we used the current IPv6 GRT as a seed and ran our algorithm with 80 threads in parallel. The respective dataset is identified as GRT_SEED$_{80}$ in Table 8.1. The GRT is compiled following our description in Section 8.4. In contrast to simply enumerating the ip6.arpa. zone, pre-aggregating to /32 prefixes takes significantly




(a) Enum. to /48 (b) Enum. to /64 (c) Enum. to /128

Figure 8.2: Executed DNS queries vs. obtained records for GRT_SEED$_{80}$.

less time. The reduced time is primarily due to the seeds in the GRT having a certain prefix length already, mostly /32 prefixes. The same can be observed when comparing the seed set among aggregated /32 prefixes. Interestingly, the dataset only increases by around 1,000 prefixes in that aggregation step, mostly due to longer prefixes being cropped. However, in the next step, we do find a significantly larger number of prefixes than those contained in the seed set. Unfortunately, the next aggregation step demonstrates that a significant amount of them are in fact dynamically-generated client allocations. Nonetheless, at more than 5.4 million unique allocated IPv6 address collected, leveraging the GRT seed to improve collection exceeds the initial dataset by far (1.6 milion to 5.4 million). It is important to note, however, that we discovered 335,670 records that are unique to the ip6.arpa. dataset. These originate from currently unannounced prefixes. The ip6.arpa. root-node should hence be included into every seed-set. However, depending on the purpose of the data collection, identified yet unrouted addresses should be marked in the collected data set.

### 8.5.3 GRT_SEED$_{400}$: Seeded Enumeration (400 Threads)

Unfortunately, a full run with 80 parallel threads takes nearly three full days to complete. Therefore, a higher time resolution is desirable. Due to low CPU load on the measurement machine we investigated the impact of running at a high parallelization degree, using 400 threads to exploit parallelization more while waiting for input/output. We refer to this dataset as GRT_SEED$_{400}$, which was collected in less than a day. In comparison to collecting with less parallel threads, we do not see a significant impact at the first aggregation level toward /32s prefixes (which we expected) due to the generally low number of them that must be enumerated here.

At the same time, we see a far higher number of obtained prefixes, primarily /64 prefixes. However, when examining the number of detected dynamically-generated and blacklisted prefixes closer, we do see that a number of dynamically-generated prefixes is not being detected correctly, which we discovered is due to packet loss. This is highlighted by the number of prefixes in GRT_SEED$_{400}$ for each aggregation level, which are considered dynamically-generated in a less specific aggregation level of GRT_SEED$_{80}$. Indeed, for

92.94% of dynamically-generated /64 in GRT_SEED$_{400}$, they have a /48 prefix already considered dynamically-generated in GRT_SEED$_{80}$.

Although the results between GRT_SEED$_{80}$ and GRT_SEED$_{400}$ differ significantly, CPU utilization for GRT_SEED$_{400}$ was not significantly higher. The core reason for this behavior is that our technique is not CPU bound. Instead, the number of maximum sockets and in-system latency during packet handling have a significantly higher impact on the result. Hence, instead of running the experiment on a single host, researchers should opt to parallelize our technique over multiple hosts.

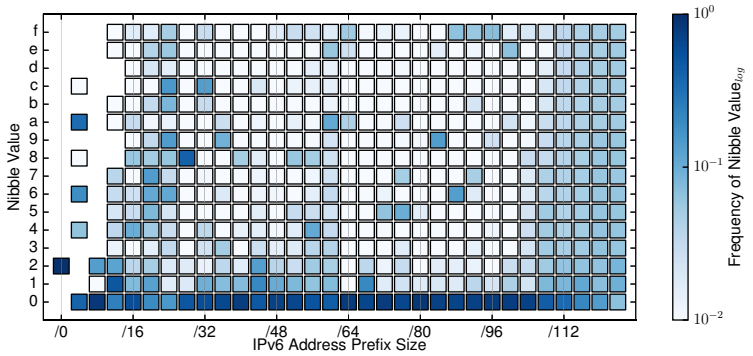
### 8.5.4 Queries per Zone and Records Found

The number of queries sent to each /32, /48 and /64 prefixes respectively versus the number of more specific ip6.arpa. records obtained per input prefix is contrasted in Figure 8.2(a)-8.2(c). An interesting insight of our evaluation is that most zones at each aggregation level contain only a limited set of records. Furthermore, we discover that the number of records found versus the number of executed queries is most densely populated in the area of less than 10 records per zone. Additionally, we see a clear lower-bound for the number of required queries. Specifically, the lower bound consists of the 16 queries needed to establish if a zone is dynamically-generated, plus the minimum number of queries necessary to find a single record. Correspondingly, for the de-aggregation to /64, an additional 64 queries are required. To go from an aggregation level of /64 to a single terminal record, at least 256 queries are necessary.

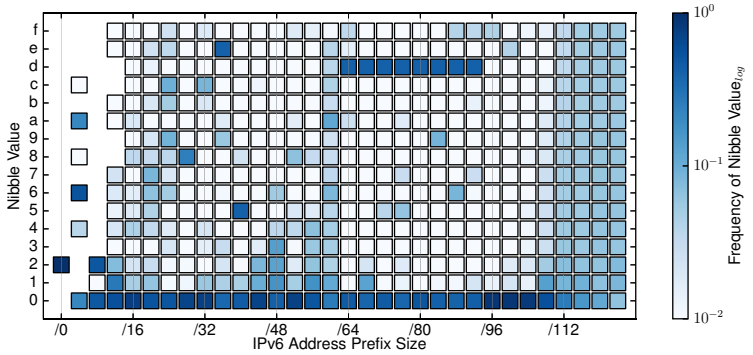
Clear upper and lower bounds for the quotient of executed queries and obtained records are also visible. In fact, these bound become increasingly clear while the aggregation level becomes more specific and follows an exponential pattern, hinting at an overall underlying heavy-tailed distribution. Furthermore, the two extremes appear to accumulate data-points, which is evident from Figure 8.2(c). The upper bound thereby corresponds to zones with very distributed entries, i.e., zones that require a lot of different paths in the PTR tree to be explored, e.g., zones auto-populating via configuration management that adds records for hosts with stateless address auto-configuration (SLAAC). On the other hand, the lower bound relates to well-structured zones, i.e., for which the operators assign addresses in an easily enumerable way, e.g., sequentially starting at *PREFIX::1*.

### 8.5.5 Address Allocation

We utilized the visualization technique introduced by Foremski et al. (Foremski et al., 2016) to analyze our dataset. To do so, we created the set of all unique IPv6 address records we obtained over all measurements. The respective results are depicted in Figure 8.3: the least significant bits are relatively evenly distributed, which aligns with our observation that zones are either very random or in some form sequential.




(a) Combined Result Set




(b) Biased Data Acquisition

Figure 8.3: Probability mass function for each nibble's value in the obtained datasets following Foremski et al. (Foremski et al., 2016). Figure 8.3(a) visualizes our combined dataset, with 5,766,133 unique IPv6 addresses. Figure 8.3(b) depicts an artifact from a measurement error in an earlier study.

Fortunately, the technique by Foremski et al. (Foremski et al., 2016) also allows us to validate our dataset. Specifically, Figure 8.3(b) has been created over an earlier dataset that we collected where an unexpected summation of the value *d* in IPv6 addresses between the $64_{th}$ and $96_{th}$ bit appears. A closer investigation revealed that this artifact was caused by a US-based educational institution that use their *PREFIX:dddd:dddd::/96* allocation for their DHCPv6 Wi-Fi access networks. As aforementioned, this dynamically-generated network was not detected due to the step-sizes in Algorithm 2, which is why we excluded it manually, see Section 8.4.4. Further work should consider adopting, e.g., 4 nibble wide steps.

## 8.6 Summary

In this chapter we introduce a new methodology for collecting a large IPv6 dataset from exclusively public data sources. Our initial evaluation of the methodology demonstrates its practical applicability. Requiring no access to a specific network vantage point, we are able to collect more than 5.8 million allocated IPv6 addresses, of which 5.4 million addresses were found in just three days by issuing 221 million DNS queries. Specifically, our technique discovered one allocated IPv6 address per only 41 DNS queries on average. Additionally, we detail the algorithms that can be used to reproduce and re-create our collection tools. To promote further research, we provide the toolchain used in this chapter to the research community at: `https://gitlab.inet.tu-berlin.de/ptr6scan/toolchain`

Leveraging our dataset we obtained for our study, we are able to provide an in-depth look into the data-centers of a large cloud provider. By comparing these results with the corresponding IPv4 reverse entries, we successfully demonstrate that our technique can discover systems which would have been missed by previous proposals for collecting IPv6 addresses for security research (Czyz, Luckie, et al., 2016). Correspondingly, it underlines that our technique is an important tool in investigating security misconfigurations in IPv6 with security scans.

Furthermore, our technique can also be applied to E.164 records (Telephone Numbers in DNS). An evaluation of the respective opportunities for phone numbers remains future work.

Our technique, in conjunction with a sufficiently scaled time resolution, can enable security research in the same way as after zMap became widely available (ShadowServer Foundation, 2014). Following the findings of Czyz et al. (Czyz, Luckie, et al., 2016), such projects are direly needed considering that firewalls for IPv6 are usually configured more lean than for IPv4. To increase the coverage of such scans, additional seeds and other address collection techniques should be integrated.

In addition, our case studies demonstrate the security impact of our technique. Apart from our observational study of a large SaaS provider we are able to provide insights into restricted networks. Specifically, our technique allows us to identify the topology of a network that can not be scanned using active probing. Furthermore, we showcase how security scans using our technique can lead to the discovery of so far unknown misconfigurations. Indeed, our third case study demonstrates a security misconfiguration that is exclusive to IPv6.

However, we also see that our technique is quiet easily mitigateable. Operators, fearful of having their numbering and topology mappings exposed, may choose to hide their PTR zones by never returning NXDOMAIN for these zones. While this is a mild violation of RFC8020 (Bortzmeyer and Huque, 2016), it is probable.

# 9

# On the Reliability of Reverse DNS as a Data Source

Our approach for collecting IPv6 datasets presented in Chapter 8 is highly based upon rDNS. Moreover, the community is polarized on the topic of using rDNS for Internet measurements. On one hand, plenty of research actively uses rDNS datasets; on the other hand, some work claims that rDNS zones below in-addr.arpa (IPv4) and ip6.arpa (IPv6) are insufficiently maintained by operators, and are not a reliable data source.

In this chapter, we investigate claims regarding the lacking reliability of rDNS datasets, clarify inaccurate common beliefs, and explore the merits of rDNS. We first conduct an in-depth analysis on passive DNS data, and, contrary to claims by prior work, we find that only 12.06% of queries for public IPv4 addresses' reverse records and 2.83% for IPv6 respectively receive no authoritative answer. Additionally, we show that the common belief that forward confirmation for mail servers dominates rDNS is inaccurate. Instead, rDNS traffic is primarily used to enrich logs for improved readability.

Subsequently, we present a validation of current techniques for gathering active rDNS datasets, and we observe that they do significantly cover the IPv4 and IPv6 address space. In fact, ip6.arpa datasets collected through rDNS expose aspects of IPv6 not covered by other datasets (such as from CDN), which have been used to investigate IPv6 adoption previously. Our analysis shows that rDNS datasets are a reliable and complimentary information source for researchers. We underline the unique research perspective of rDNS data with case-studies that shed new light on challenges of deploying IPv6.

## 9.1 Motivation

The Domain Name System (DNS) has become an integral part of the Internet. Forward DNS, which resolves names like `www.google.com` to an IP address makes the Internet usable for end-users. Its counterpart is reverse DNS (rDNS), which allows resolving the name behind an IPv4 or IPv6 address. To resolve an IP address to a name, IANA designated two second level zones below .arpa, in-addr.arpa and ip6.arpa. Below them, operators can have zones corresponding to their IP network prefixes delegated to their authoritative nameservers. There, they can serve pointer (PTR) resource records to point to the fully

qualified domain name (FQDN) for an IP address. Some of the use cases of rDNS are the forward confirmation of mail servers to fight spam (Cormack, 2007) and enriching logs for improved readability and debugging (Nicholas and Huntington, 2003).

rDNS as a tool provides valuable information on networks to operators and researchers. In fact, it is often used by researchers to gather valuable information on network topologies (R. V. Oliveira et al., 2008; M. Zhang et al., 2006). However, recent work questions the reliability of rDNS as a data source for Internet measurements. Specifically, Gao et al. claim that operators do not sufficiently care toward rDNS, as they observe that 25.1% of all rDNS queries can not be authoritatively answered (Gao et al., 2013). Furthermore, Phokeer et al. report an increasing number of broken rDNS delegations for the APNIC region (Phokeer et al., 2016). In turn, it raises the question whether techniques relying on rDNS to perform Internet measurements, e.g., the one proposed in Chapter 8 to measure IPv6 adoption, are actually viable and meaningful.

Therefore, in this chapter, we investigate these prior results on the use of rDNS. Specifically, we aim to re-investigate and validate results of prior research on the use of rDNS by operators. We also revisit the technique presented in Chapter 8, and evaluate how much of the IPv6 reverse DNS it covers. Using these datasets, we further investigate in how far rDNS data can provide new insights on the adoption of IPv6, and in how far the use of reverse DNS differs between IPv4 and IPv6.

## 9.2 Related Work

Prior work on rDNS was commonly part of more general approaches to forward DNS. Hao et al. conducted an Internet-wide observation of DNS lookup patterns, however, they have not focused or reported on rDNS in their work (Hao et al., 2010). Gao et al. follow up on Hao's work and provide an updated look on DNS three years later (Gao et al., 2013). However, neither performs an in-depth dive into rDNS as we do. Considering all PTR requests in their dataset rDNS related queries, they determine that 25.1% of all rDNS queries do not receive an authoritative answer. In turn, it leads them to the conclusion that rDNS zones are often poorly maintained. Even earlier, Jung et al. investigated DNS caching effectiveness and encounter similar patterns for rDNS (Jung, Sit, et al., 2002) and coming to the same conclusions, again, as we will later show, given incorrect assumptions. Finally, Zhang et al. find that inaccuracies in rDNS may reduce its usefulness for topology discovery (M. Zhang et al., 2006).

Still, researchers heavily rely on DNS measurements. For example, Oliveira et al. supplement their discovery of the Internet's topology with rDNS data (R. V. Oliveira et al., 2008). Kalafut et al. investigate the effects of misconfiguration in zone provisioning, however, only for forward DNS (Kalafut et al., 2008). At the moment, the major use-case of active rDNS measurements is an investigation of the deployment of IPv6. Specifically, we also contribute a technique for enumerating allocated IPv6 addresses in Chapter 8. However, as their work focuses on describing the technique, they lack an in-depth validation of their

results. Regardless, their work is promising, as it may provide a counter-dataset to the CDN datasets commonly used for IPv6 measurements. In fact, the CDN dataset has been used by Plonka and Berger (Plonka and Berger, 2015), as well as Foremski et al. (Foremski et al., 2016), to investigate IPv6 deployment. Other contemporary work that utilizes rDNS includes the observations on IPv6 adoption by Czyz et al. (Czyz et al., 2015). Similarly, Czyz et al. leverage rDNS to identify dual stack hosts, for which they then evaluate their security posture (Czyz, Luckie, et al., 2016).

Overall, some prior work claims that rDNS zones are insufficiently maintained, and, therefore, are not a plausible research tool. At the same time, other research relies on rDNS and actively uses it as a data source. In this chapter, we revisit prior claims on the reliability and quality of rDNS measurements using the same datasets that have been used by prior studies. Furthermore, we validate the technique presented in Chapter 8, by contrasting the findings we obtain with it to the results of our analysis of passive rDNS traces. We also cross-validate the technique with IPv4, by comparing it to an active evaluation of IPv4 rDNS. Finally, we illustrate the potential of rDNS-based studies by documenting new insights into the deployment of IPv6.

## 9.3 Passive Trace Set: Is rDNS Well Maintained?

Following, we re-evaluate prior findings on how rDNS is used, to determine if rDNS is a reliable and meaningful source for data used in research. Specifically, we test the claim of Gao et al. (Gao et al., 2013) that rDNS is commonly poorly maintained, and the common assumption that the forward confirmation of mail server is the dominant use case of rDNS.

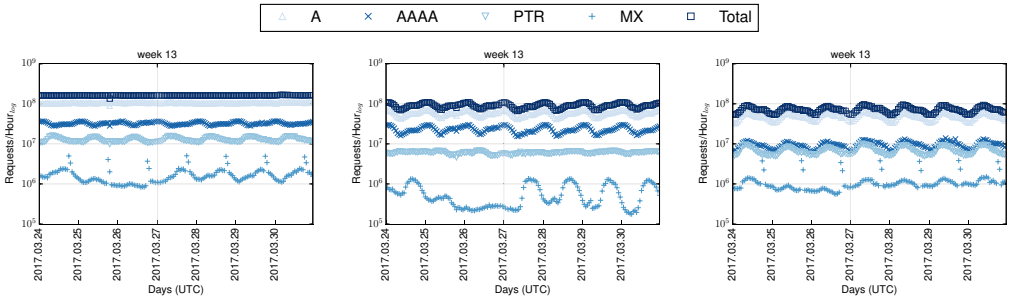
### 9.3.1 Dataset Collection

We leverage the Farsight DNS dataset to get data on the practical use of rDNS by clients for our study. The Farsight dataset is a collection of traces from recursive DNS resolvers of Internet service providers from all around the globe, made available by the Farsight Inc. We use this dataset, as previous studies for which we revisit the findings, especially Gao et al. (Gao et al., 2013) have used the same source for data for their analysis.

We receive a full stream of DNS messages encoded in the Network Message (NMSG) format provided by Farsight. For the purpose of our study we use DNS traffic observed between March 23$^{rd}$, 2017 and April 17$^{th}$, 2017 from midnight to midnight (UTC). As the focus of our study is on rDNS, and to address privacy constraints we pre-aggregate this data. Specifically, we limited our dataset to the requested reverse pointer, the time of the request, the returned FQDN, the recorded SOA, the replying nameserver, information on whether a CNAME was observed, and, if present, the error that occurred. Table 9.1 shows the full list of the fields we retained.

| Record Type | Data Field | Data Type |
|---|---|---|
| Farsight Record | AUTHORITY | DNS AUTHORITY |
| | ANSWER | DNS ANSWER |
| | QUESTION | DNS Query |
| | rcode | DNS rcode |
| | Query IP | IP address |
| | Requested PTR | String |
| | Returned Record | String |

Table 9.1: Overview of datatypes in the agregated farsight dataset.




(a) Full Farsight dataset. (b) Only biased operator. (c) Dataset w/o biased operator.

Figure 9.1: An overview of the first week of the passive trace for the three dataset splits. We only visualize the first week as this greatly enhances readability of the figures, while we did not observe any unexpected patterns in the latter two weeks.

### 9.3.2 Dataset Split

An initial exploratory analysis of the dataset shows that requests from a single ISP's recursive DNS servers heavily bias the dataset (see Figure 9.1). Specifically, we observe the following effects in the initial dataset (see Figure 9.1(a)):

1. Absence of any daily pattern in the total number of requests and number of A requests.
2. Disjoint day patterns between the number of AAAA and PTR queries.
3. Constant queries for PTR records in ip6.int., the discontinued rDNS zone for IPv6 (Huston, 2005), for addresses outside of the currently used IPv6 range (7000::/8).
4. Regular DNS Service Discovery (Cheshire and Krochmal, 2013) PTR requests for domains such as icloud.com, mac.com and dell.com. All in the same order of magnitude of requests per minute.

Therefore, we split the dataset in two sub-sets: (i) containing only the ISP with the strange request patterns, and (ii) containing all other operators' data.

Investigating the single operator, we discover that it contributes nearly half of all DNS queries in the original dataset (see Figure 9.1(b)). Furthermore, the operator has a significantly larger number of AAAA requests compared to the rest of the dataset. Furthermore, the flat-line we observe for PTR requests consists almost exclusively of requests to ip6.int.

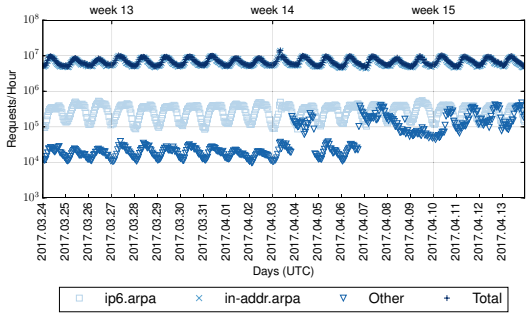



Figure 9.2: Requests to ip6.arpa, in-addr.arpa and other second-level domains.

and service discovery requests. Given these abnormal patterns, we attribute these effects to misconfigured Customer Premise Equipment in the ISP's network.

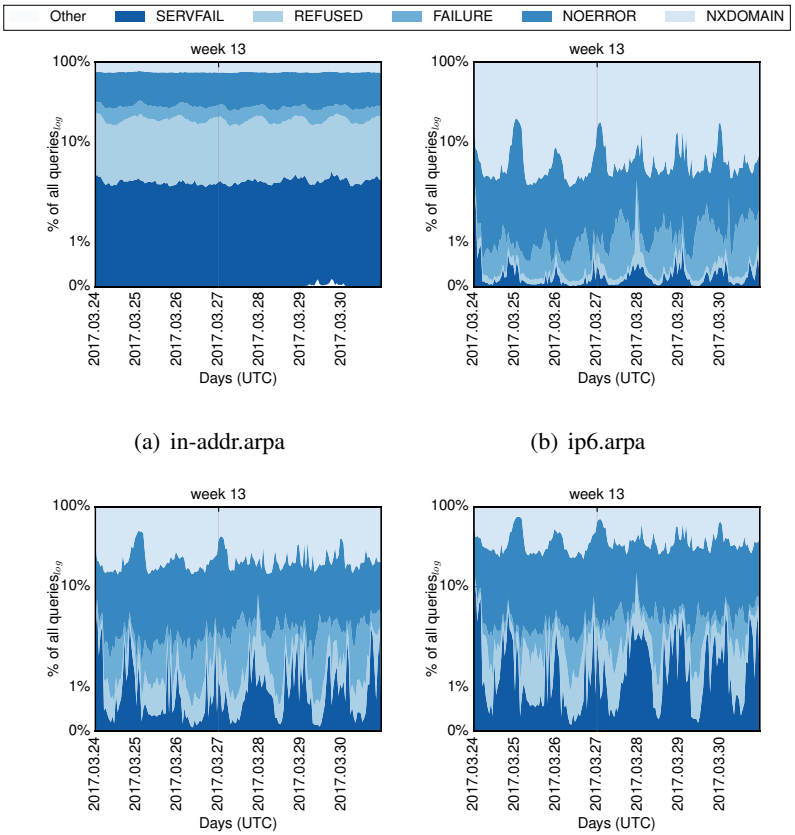
The dataset excluding this operator does not contain these unexpected effects (see Figure 9.1(c)). The remaining daily outliers for MX requests can be attributed to a Russian ISP running a daily mass-mail campaign. We identified this ISP by conducting a heavy-hitter analysis on the dataset. Furthermore, the overall share of PTR requests is in the same order of magnitude as AAAA requests (see Figure 9.1(c)), and they are still in the same order of magnitude as Gao et al. (Gao et al., 2013) reported.

Since Gao et al. did not publish their dataset, we can not verify whether their sample from the same source of data contained this operator, and if so, whether the operator was already inducing the same bias that we observed. Given the fact that their overview over their dataset correlates with our clean dataset (depicted in Figure 9.1(c)), we assume that the biasing operator was not yet present in the dataset or did not induce the bias yet. Hence, we will use the dataset without the offending operator for our subsequent analysis.

### 9.3.3 Dataset Overview

Next, we go beyond the scope of Gao et al. and look at the second level domains for which the PTR queries are issued. It allows us to distinguish between rDNS queries for IPv4 and IPv6 addresses, as well as other use cases of PTR queries. Requests to in-addr.arpa, the reverse zone for IPv4, accounts for the major part of all PTR requests, followed by ip6.arpa, the reverse zone for IPv6 (see Figure 9.2). Still, requests for ip6.arpa are two orders of magnitude less frequent than requests for in-addr.arpa.

This low number of IPv6 reverse queries hints that the adoption of IPv6 in the "heavy tail" of the Internet and it is significantly lower than the numbers for IPv6 adoption observed at large content providers (Foremski et al., 2016; Google, 2017; Plonka and Berger, 2015). However, it corresponds to earlier observations by Pujol et al. (Pujol et al., 2017). In addition, it is also supported by data from a group with access to the IXP dataset (Chatzis et al., 2013), whom we contacted and who confirmed our findings. In their dataset, they find an average of 2.96%/2.47% (packets/bytes) IPv6 traffic via the IXP for a week in early 2017,




(a) in-addr.arpa (b) ip6.arpa

(c) ip6.arpa w/o Teredo (d) ip6.arpa w/o Resv.

Figure 9.3: Share of response codes.

again in strong contrast to the, e.g., 17.98% adoption reported in the Google IPv6 statistics for April 29th, 2017 (Google, 2017).

In addition to the IPv4 and IPv6 rDNS zones, we encounter a number of requests to other top and second level domains. Interestingly, these are commonly related to DNS based service discovery (DNS-SD) (Cheshire and Krochmal, 2013) by clients, which account for 77.04% of observed queries outside of .arpa. The outliers in the "Other" category (see Figure 9.2) starting April 4th, 2017 correspond to DNS-SD queries within the domain of a major news network, which leaked into the Farsight dataset via a single operator. However, they are not frequent enough to notably influence the total numbers and, thus, would not cause artifacts (in Figure 9.1). Instead, they are most likely related to a newly deployed piece Customer-Premises Equipment (CPE), such as a set-top box, or an application for such a device receiving an update on or shortly before April 4th, 2017.

| rcode | in-addr.-arpa | ip6.arpa | ip6.arpa w/o Teredo | ip6.arpa w/o Resv. |
|---|---|---|---|---|
| NOERROR | 47.21% | 4.00% | 18.77% | 32.11% |
| NXDOMAIN | 25.36% | 94.87% | 78.33% | 63.93% |
| REFUSED | 15.47% | 0.14% | 1.42% | 1.13% |
| FAILURE | 8.77% | 0.81% | 0.83% | 1.41% |
| SERVFAIL | 3.17% | 0.18% | 0.66% | 1.42% |
| FORMERR | 0.01% | ≤0.01% | ≤0.01% | ≤0.01% |
| NOTAUTH | ≤0.01% | - | - | - |
| NOTIMP | ≤0.01% | - | - | - |

Table 9.2: Distribution of response codes (rcodes) for ip6.arpa and in-addr.arpa for the week depicted in Figure 9.3.

### 9.3.4 Share of Answer Response Codes

The conclusion by Gao et al. that rDNS zones are often poorly maintained is based on their observation that 25.1% of all PTR queries do not receive an authoritative answer. However, they do not differentiate between the different PTR queries, i.e., they also count queries that do not correspond to rDNS, e.g., DNS-SD. We investigate the response codes (rcodes) that we observe for PTR queries to the rDNS zones, i.e., queries that can be clearly and safely attributed to rDNS (see Table 9.2 for an overview).

**in-addr.arpa** For in-addr.arpa, 47.21% of all queries are successful, while 25.36% return NXDOMAIN, and 15.47% return REFUSED, possibly because the operators want to hide internal information which would become public from the hostnames returned for these RRs (see Figure 9.3(a)). The brief increase of "Other" replies on March 29$^{th}$, 2017 is attributed to DNS servers of a Singaporean ISP returning FORMERR for all requests. Due to the short duration of the event, we assume a temporary misconfiguration. Furthermore, we find a solid socket of 3.17% of queries returning SERVFAIL, indicating that some zone delegation for the in-addr.arpa zone is broken, or that the authoritative DNS server does not respond correctly. It leaves 8.77% of all queries returning FAILURE and less than 0.02% returning FORMERR, NOTAUTH, and NOTIMP. Overall, it means that only 12.06% of PTR requests to in-addr.arpa cannot be authoritatively answered, which stands in significant contrast to the 25.1% reported by Gao et al. (Gao et al., 2013). More importantly, only 3.17% of queries cannot be authoritatively answered due to poor or missing maintenance, i.e., broken delegations.

**Heavy Hitters in ip6.arpa** Contrary to in-addr.arpa, for ip6.arpa only 0.99% of all requests cannot be authoritatively answered. However, we also find that just 4.00% of queries result in a NOERROR response. Instead, ip6.arpa is dominated by NXDOMAIN replies, which account for 94.87% of all responses (see Figure 9.3(b)). Interestingly, this large share of NXDOMAIN responses is caused by only a small number of heavy hitters. We identify the networks to which these records belong as Teredo (Huitema, 2006) prefixes.

Excluding these hosts yields a more coherent picture, which we refer to as "ip6.arpa w/o Teredo" (see Table 9.2 and Figure 9.3(c)). Indeed, the overall response rate after filtering out Teredo hosts increases, with NXDOMAIN now accounting for 78.33% of all responses, and NOERROR now reaching 18.77%. The number of FAILURE and SERVFAIL responses does not significantly change, while REFUSED responses make up 1.42% of observed ones for ip6.arpa. Most interesting is that SERVFAIL and REFUSED remain, compared to in-addr.arpa, relatively low. We conjecture, that the SERVFAIL is less pronounced for ip6.arpa than it is for in-addr.arpa because in-addr.arpa has been in use much longer. As such, it provides more time for things to go wrong, i.e., delegations and systems to break. The lower REFUSED rate may be attributed to lower security measures being in place for IPv6 systems and infrastructure yet, also reported by Czyz et al. (Czyz, Luckie, et al., 2016).

**Local and Reserved Networks in ip6.arpa** Looking at the still large number of NXDOMAIN for ip6.arpa, most queries to records in ip6.arpa actually correspond to private and reserved addresses. Specifically, 41.77% of all queries to ip6.arpa correspond to addresses in reserved or private ranges (Teredo is considered as "Reserved" here), while the same is the case for only 1.21% of all queries to in-addr.arpa[1]. This effect has been observed by Czyz et al. already, who found significant leakage of traffic for link-local addresses in their IPv6 background radiation study (Czyz, Lady, et al., 2013). These queries are naturally answered with NXDOMAIN responses by the authoritative servers for ip6.arpa, as the corresponding zones are not actually delegated (Bortzmeyer and Huque, 2016). We attribute the difference between in-addr.arpa and ip6.arpa to insufficiently deployed filtering on operators recursive servers. While it is common practice to stop queries to private parts of in-addr.arpa before they are forwarded upstream, it is not (yet) being done for ip6.arpa. An alternative explanation might be that some tools may not be able to detect IPv6 link-local unicast addresses (fe80::/10) (Hinden and Deering, 2006), or other reserved networks, as such, and, hence, issue queries for addresses in those ranges.

Unsurprisingly, we find that filtering on our dataset emphasizes the observed daily patterns (see ip6.arpa w/o Resv in Table 9.2 and Figure 9.3(d)). Furthermore, with NXDOMAIN accounting for now 63.93%, we observe a NOERROR rate of 32.11%. Overall, we see a significantly higher NOERROR rate for ip6.arpa than the initially observed 4.00%, but the rate still lacks behind the 47.21% successful responses for in-addr.arpa.

**Summary** Our results paint a different picture than that drawn by Gao et al. (Gao et al., 2013). They report that 25.1% of PTR queries cannot be answered authoritatively, and use it to support their conclusion that rDNS is insufficiently maintained by operators. However, we find that this view is biased due to not sufficiently differentiating between different types of PTR queries.

[1] We do not provide a graph for the filtered in-addr.arpa dataset in Figure 9.3, as the filtering does not have any visual impact on the depiction of the returned rcodes.

| Port | Protool | Port | Protocol |
|---|---|---|---|
| 25 | SMTP | 587 | SMTP Submission |
| 110 | pop3 | 993 | IMAPs |
| 143 | IMAPv4 | 995 | pop3s |
| 465 | SMTPs | - | ICMP |

Table 9.3: Scanned ports and protocols.

For in-addr.arpa, we find that only 12.06% of all rDNS queries cannot be authoritatively answered. Therein, only 3.17% of all queries fail with SERVFAIL, indicating broken delegations or misconfigured authoritative nameservers. For IPv6 even less queries fail, with only 2.83% of queries receiving no authoritative answer, of which only 1.42% fail due to a SERVFAIL. These numbers stand in direct opposition to the findings of Gao et al. and, following their reasoning, demonstrate the that rDNS is actually actively and well maintained. Furthermore, we find that rDNS for ip6.arpa is more consistently maintained. We attribute this to the fact, that IPv4 has been around significantly longer, and there was simply more time for things to go wrong and get fixed.

Revisiting Gao et al., we conclude that the situation has either significantly improved, since their study in 2013, or that insufficiently differentiating PTR queries' target domains lead them to build a conclusion on inaccurate premises. In any case, we can conclude that rDNS is currently a reliable and viable source for data used by research.

### 9.3.5 rDNS for SMTP Forward Confirmation

Next, we investigate the claim that rDNS is mainly used for forward confirmation of mail servers. We recall that forward confirmation is a common tool in mitigating email spam (Cormack, 2007). In fact, it is claimed to be *the* dominant use case of rDNS. If this is indeed true in practice, then most addresses for which we see rDNS queries should be mail servers (i.e., mail servers communicating among each other). Hence, we should be able to encounter services listening for mail sending and receiving on these hosts.

Evaluating systems for mail services requires active measurements, and these scans have to be conducted sufficiently soon after observing a request, to not be affected by churn. Therefore, we selected April 19$^{th}$, 2017 for our in-depth evaluation. We scan all hosts as soon as they appear in the dataset and we ensure that every host is only scanned once. For each host we then check if it runs at least one mail related open TCP port (see Table 9.3), and whether they reply to ICMP echo requests.

We find that 19.98% of all addresses for which we see in-addr.arpa requests respond to ICMP echo requests, while 15.28% of all hosts for which we see ip6.arpa requests reply to ICMP. Hosts running mail services as detailed in Table 9.3 contribute 10.05% of active hosts in in-addr.arpa, amounting to 2.01% of all hosts for which we saw rDNS queries. However, for ip6.arpa, 31.53% of reachable hosts, or 4.82% of all hosts, have open mail related ports.

These numbers align with our earlier observation, namely that the share of "serious" traffic for ip6.arpa is higher than for in-addr.arpa.

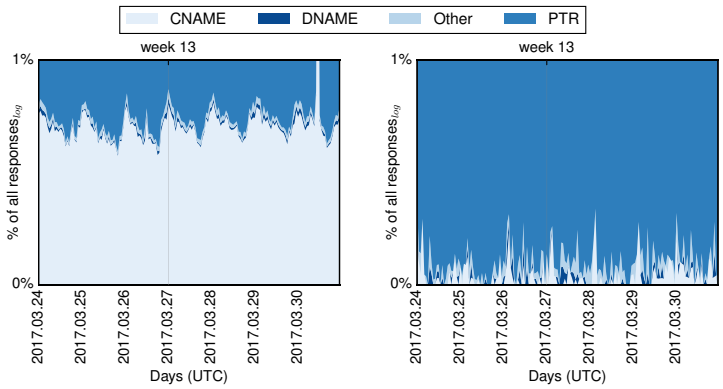
Considering that forward confirmation is commonly not performed for MUA (Mail User Agent) connections that try to relay an email, for which the user trying to send an email is required to authenticate herself, in practice, forward confirmation should be performed mostly for: (i) spam senders, and, (ii) other mail servers. However, with the increased use of blacklists (Levine, 2010; Ramachandran et al., 2006), and adoption of Sender Policy Framework (SPF) and DomainKeys Identified Mail (DKIM) over the past years, spam distribution moved to using (compromised) mail servers, and sending spam emails via compromised mail accounts of legitimate users (Alazab and Broadhurst, 2017; Hu et al., 2016). Although our results are a lower bound, they strongly indicate that forward confirmation of MX servers is *not* the dominant use-case of rDNS (anymore), contrary to common belief.

### 9.3.6 Churn in Queried Reverse Names

Since forward confirmation is not the dominant use case of rDNS anymore, we evaluate the churn of requested names for rDNS to test the assumption that many rDNS requests are generated by IDSs and log systems upon receipt of client connections. Unfortunately, a simple heavy-hitter analysis is not possible because it would require information on the systems that request rDNS information. Due to privacy concerns, this information is not available in the Farsight dataset. If our assumption is correct, we should see a relatively low foundation of stable addresses, accompanied by a large amount of reoccurring and newly queried names.

Particularly, we investigate the churn of ip6.arpa (/128 and /64, Figure 9.5(c) and 9.5(b)) and in-addr.arpa hosts (see Figure 9.5(a)), both filtered for private and reserved addresses. This allows us to reason about how many reverse queries are issued for server systems (i.e., systems that commonly reoccur), and how many are issued for clients with changing addresses. For IPv6 clients, privacy addresses (Narten et al., 2007) are a likely cause for changing addresses, while Carrier Grade NAT (Perreault et al., 2013) is a common cause for IPv4 addresses, among others. In comparison, in-addr.arpa and per-/64 aggregated requests to ip6.arpa exhibit around 50% of reoccurring records after three weeks (49.74% for in-addr.arpa and 53.91% for ip6.arpa). However, for full IPv6 addresses, only 40.07% of records reoccur. Looking at the share of queried names seen on subsequent days, this number changes: In fact, it pertains to 24.29% of all records in in-addr.arpa, while, on average, 35.34% of all records have been seen on the previous day for ip6.arpa aggregated to /64s as well. Interestingly, with 30.12%, this is not significantly lower for full IPv6 addresses.

These results support our assumption that a major portion of rDNS requests is actually related to client addresses being resolved. Furthermore, we find that the low number of reoccurring hosts for full IPv6 addresses aligns with findings of prior work regarding the dynamic use of /64s for privacy extensions (Foremski et al., 2016; Plonka and Berger, 2015).




(a) in-addr.arpa (b) ip6.arpa

Figure 9.4: Share of response types.

### 9.3.7 RRtypes in Successful Answers

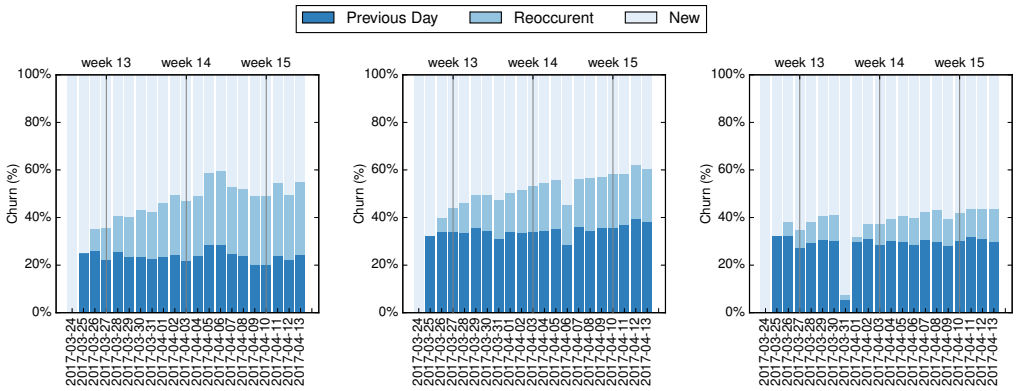
Next, we make specific observations on the use of rDNS, which will serve as landmark observations used for comparison later on, to validate our active rDNS traces.

Naturally, the RRtypes of responses to rDNS queries are dominated by PTR RRs. Given that in-addr.arpa is split at the octet-boundary, while IPv4 networks are not anymore (see Section 8.2), we expect a notable number of CNAME responses for in-addr.arpa, but not for ip6.arpa. Specifically, the share of CNAMEs should be higher for in-addr.arpa as they are used to delegate rDNS authority for networks that are smaller than a /24 network (Eidnes et al., 1998). Indeed, we find that CNAMEs account for 0.71% of all query responses in in-addr.arpa. While this share is comparatively low, it still constitutes a solid socket (see Figure 9.4, note the logarithmic scale from 0% to 1%), particularly compared to ip6.arpa. Similarly, we find a small layer of DNAMEs[2] for in-addr.arpa, but not for ip6.arpa. Any other record types (A, SOA, etc., labeled "Other" in the graph) relate to additional information sent by authoritative nameservers, e.g., sending along the A record for the returned FQDN in a PTR request.

### 9.3.8 Summary

In this section, we revisited prior measurement lore on the use of rDNS. We find that prior estimations on the use of rDNS are incorrect. Either because the use of rDNS has significantly improved for the better, by an order of magnitude since 2013, or because prior work arrived at inaccurate conclusions because they did not perform an in-depth analysis of the underlying dataset that their conclusion is based on. Specifically, contrary to Gao et al., the Farsight passive trace dataset does not support the conclusion that operators largely maintain

[2]DNAMEs work like CNAMEs, but instead of referring a single record, they apply to a full zone.




(a) in-addr.arpa (b) ip6.arpa (/64) (c) ip6.arpa (/128)

Figure 9.5: Churn for requested names in in-addr.arpa and ip6.arpa.

their reverse zones insufficiently (anymore). Subsequently, we also determine that forward confirmation by mail servers is only a minor part of rDNS use nowadays, instead of being the dominating one, as networking lore still proclaims. In fact, the resolution of connecting clients, e.g., for improved readability of logs by humans, is the major use-case of rDNS today. In addition, delegating reverse IPv6 addresses at the nibble-boundary, instead of an octet-boundary as for IPv4, has successfully reduced the need for CNAME based delegation techniques.

## 9.4 Active rDNS measurements: What do we really see?

In the previous section, we challenged existing measurement lore, and we demonstrated that rDNS is indeed well maintained and commonly used, contrary to lore. However, our data still shows that a major number of rDNS queries for names that do not exist (NXDOMAIN). One might think that it is evidence for poorly maintained zones, i.e., operators not populating zones, even though they are correctly delegated. To evaluate this assumption, we leverage active datasets of the in-addr.arpa and ip6.arpa rDNS zones. We use these datasets to describe which portion of systems connected to the Internet we can observe with rDNS, and what portion of the rDNS space we can enumerate with the technique presented in Chapter 8.

### 9.4.1 Collection Methodology for Active Datasets

To continue our investigation of rDNS, we actively collected in-addr.arpa and ip6.arpa datasets.

| Record Type | Data Field | Data Type |
|---|---|---|
| arpa. Record | timestamp | Unix Timestamp |
| | IPversion | (IPv4‖IPv6) |
| | ARPA | Evaluated .arpa record |
| | AUTHORITY | DNS AUTHORITY |
| | ANSWER | DNS ANSWER |
| | rcode | DNS rcode |
| | BLACKLIST | Bool |
| | AUTOGEN | Bool |
| | PREFIXLENGTH | IPv6 prefixlength |

Table 9.4: Overview of datatypes in the active datasets.

**Data Collection Infrastructure** All actively collected datasets have been obtained on the same measurement platform: a cluster of 16 machines, each with an Intel Xeon X3450 CPU, 8GB of main memory, 300GB of hard disk storage as data collection platform. Each system runs a local recursive DNS resolver (Unbound 1.4.22) against which we perform all DNS queries to benefit from caching effects. The cluster is orchestrated by an additional workstation which distributes jobs using GNU parallel. We do not perform connection tracking for any traffic from any machine, and we disabled it on all upstream routers that are under our control, up to the default-free zone (DFZ).

**IPv6 Reverse DNS Dataset** We use the technique presented in Chapter 8, between March 26$^{th}$, 2017 01:04 UTC and March 30$^{th}$, 2017 10:49 UTC, to collect a dataset with more than 10.2 million reverse records. We also store intermediate information for non-terminal records, to understand how IPv6 reverse zones are delegated and to compare that to the IPv4 datasets. Furthermore, we also collect CNAME records, in addition to PTR records.

**Complete IPv4 Dataset** In contrast to ip6.arpa, in-addr.arpa can be enumerated through brute-force within a reasonable amount of time. Hence, we gather a complete IPv4 rDNS dataset which will serve as a baseline for our evaluation. For this dataset, we query all possible name records under in-addr.arpa (see Table 9.4 for the dataset's format). To speed up the process, we exclude rDNS zones for private, reserved, or unused IPv4 addresses. If no PTR record exists for an IPv4 address, then we check for a possible delegation via a CNAME record. Using this technique, we collect 1.18 billion reverse records between April 6$^{th}$, 2017 20:23 UTC and April 15$^{th}$, 2017 11:20 UTC.

**NXDOMAIN IPv4 Dataset** We also ported the NXDOMAIN technique of Fiebig et al. to the IPv4 rDNS zone. This, in contrast to the brute-force approach, allows us to investigate delegation in IPv4 rDNS. Specifically, we use the following steps to collect the dataset:

1. We collect a view on the GRT from RIPE RIS (Ripe NCC, 2016b) and Route-views (University of Oregon, 2016) and add in-addr.arpa to the seed set.

2. We use NXDOMAIN enumeration to perform a breadth-first search in the tree (instead of 16, every node now has 256 possible children).
3. When the algorithm finds a terminal node, we terminate for that branch.

Using this approach, we collect an in-addr.arpa NXDOMAIN dataset between March 31st, 2017 16:28 UTC and April 6th, 2017 05:46 UTC. It is comprised of 1.21 billion terminal records and CNAMEs.

**Brute-Force vs. NXDOMAIN** Comparing the results of the brute-force and NXDOMAIN enumeration approaches for in-addr.arpa, we can evaluate the coverage of the NXDOMAIN approach. We find 1.21 billion PTR records in the NXDOMAIN dataset, and 1.18 billion PTR records in the brute-force dataset. That the NXDOMAIN dataset contains more records can be attributed to the technique finding records outside the seed set, as mentioned in Chapter 8. More specifically, the NXDOMAIN dataset for in-addr.arpa overlap in 1.1 billion records, with 106 million records (8.76%) being unique to the NXDOMAIN dataset and 79 million PTR records (6.69%) being unique to the brute-force dataset. Moreover, the NXDOMAIN approach was faster (5.5 days runtime vs. 8.6 days), while collecting more information on the intermediate DNS levels (e.g. 168.192.in-addr.arpa). The difference in runtime is caused by zones which are not well delegated and time out. While these are excluded early in the NXDOMAIN approach, the brute-force approach still tries to request all leaf records in these zones, subsequently running into timeouts. In conclusion, NXDOMAIN-based collection of in-addr.arpa rDNS provides nearly the same results, while it is significantly faster and collects auxiliary information on intermediate zones. Hence, in the remainder of this section, we use the NXDOMAIN-based dataset.

**Ethical Considerations** Our work follows the same ethical guidelines as already detailed in Chapter 8. As discussed there, in our scans we ensured that outbound traffic to any given single system did not exceed 2MBit/s. We also set appropriate rDNS entries for our scanning systems (see Section 8.2).

**Dataset Availability/Reproducibility** To facilitate the independent reproducibility of our results, we provide our full toolchain for collecting and parsing all data that we used in our study. The toolchain is available at: `https://gitlab.inet.tu-berlin.de/ptr6scan/toolchain`. While we will not make the datasets publicly available, we will provide it to researchers upon request. We refrain from doing so for ethical reasons: the collected datasets hold a large amount of server-side IPv6 addresses that are not covered by prior research. Considering that previous research has demonstrated that IPv6 security measures are commonly lacking behind (Czyz, Luckie, et al., 2016), we only publish our data collection toolkits, and not a readily available dataset, to raise the bar for attackers.

### 9.4.2 Visible IPv4 Space in in-addr.arpa

Comparing the in-addr.arpa dataset with the global IPv4 space, we gain an understanding of how well rDNS are maintained and populated by network operators. In an ideal world, we would see rDNS names, i.e., either CNAMEs or PTRs, for all allocated IPv4 address. With 1.21 billion PTR records in the in-addr.arpa dataset we see rDNS names for 28.17% of the total IPv4 address space. Technically, it is a considerably low coverage, but the IPv4 space is not fully used on the public Internet (Richter, Allman, et al., 2015). Hence, we compare these numbers to the 1.2 billion active IPv4 addresses reported by Richter et al. (Richter, Smaragdakis, et al., 2016). We note that we reach a considerably close overlap. This indicates that rDNS zones are not only well delegated in general, as we concluded beforehand (see Section 9.3), but also that network operators do indeed actively populate and maintain their rDNS zones. Based on our prior observation that ip6.arpa zones are less frequently involved in broken delegations or have unresponsive servers than in-addr.arpa zones, we expect to find a similar overlap of active IPv6 addresses and the ip6.arpa zone.

### 9.4.3 Covered IPv6 Space in ip6.arpa

Fiebig et al. demonstrate that their method is applicable for gathering IPv6 rDNS datasets at scale. However, they do not evaluate the accuracy and completeness of their method. Hence, we briefly compare and validate our actively gathered datasets. Specifically, we determine an upper bound of what the NXDOMAIN approach by Fiebig et al. sees, and how it relates to other datasets commonly used for IPv6 adoption studies.

**ip6.arpa vs. CDN Dataset** The CDN dataset was used by several prior publications, which form the current state of the art base-line for investigating IPv6 adoption (Foremski et al., 2016; Plonka and Berger, 2015). Researchers with access to the dataset agreed to provide us with comparative aggregated data on our dataset. They reported that the plain overlap between the ip6.arpa dataset and their CDN dataset is 81K hosts, of which they identify 70K as stable, i.e., reoccurring on three subsequent days. Hence, we conclude that the ip6.arpa dataset covers part of the IPv6 address space which has not been the subject of prior studies. In fact, it aligns with our observations on the share of actual IPv6 usage in the Internet's heavy-tail (see Section 9.3).

Moreover, we note that /32s and /48s are the most commonly dynamically generated zones in the ip6.arpa dataset. We attribute this to the delegation concepts in ISPs: They either hand out /64s or /48s networks to their customers (Foremski et al., 2016; Plonka and Berger, 2015). Hence, they dynamically generate zones starting at the covering standard prefix size, i.e., /32s or /48s. Note, that the most likely reason for the low overlap with the CDN dataset is that that the CDN dataset is client-centric. Most of the addresses found in the CDN dataset should reside in network prefixes that we detect as dynamically generated. While it would have been beneficial to evaluate which portion of the CDN dataset is covered by the

prefixes that we determine to be automatically generated, due to privacy concerns, it has not yet been possible.

**RFC8020 Compliance** We have established that NXDOMAIN-based enumeration of ip6.arpa finds a significant number of records that are not part of other IPv6 datasets, we also want to establish an upper bound of the IPv6 address space's portion that can be observed using this technique. The technique heavily depends on authoritative servers correctly implementing RFC8020 (Bortzmeyer and Huque, 2016). Thus, we investigate how frequently rDNS servers adhere to it. From the Farsight dataset, we obtained all queries for entries in ip6.arpa, which were replied to successfully, a total of 361K unique names. For each record, we now determine all zone delegations up to the root (ip6.arpa) via which the leaf record can be reached. We then query for the NS records of all intermediate zones.

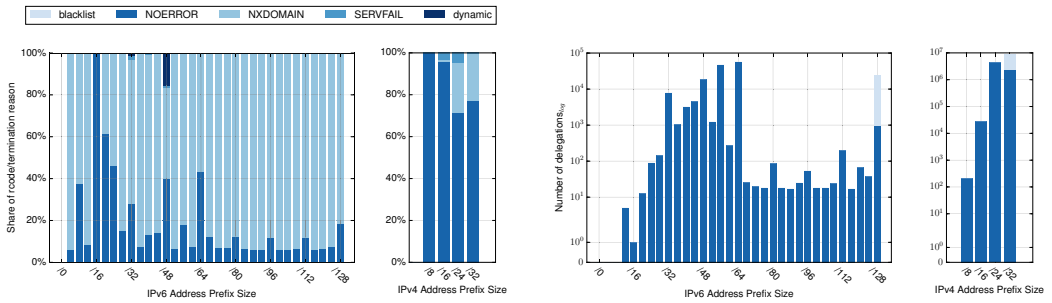
Utilizing the initial leaf records, we can now test each of the authoritative name servers for all identified domains, if they: (i) follow RFC8020; (ii) always return NXDOMAIN, even though an element in the tree below them exists; (iii) always return NOERROR, even though nothing exists below the queried records; (iv) do return an error (SERVFAIL, REFUSED, timeouts); and, (v) if there are any differences for this between the different authoritative servers of a domain.

We discover that 39.58% of all rDNS zones in the dataset only use authoritative servers in compliance with RFC8020, while 46.42% always return NXDOMAIN, and 11.61% always return NOERROR. In turn, we will detect 46.42% of zones as having no entries at all, while 11.61% of zones will be flagged as dynamically generated. The remaining 2.38% are split among 0.59% of zones that return errors, and 1.79% of zones exhibiting a mix of the above conditions. We note that for the latter, at least one nameserver is compliant with RFC8020 and can be used for enumeration, while the others always return NXDOMAIN or REFUSED.

Therefore, the visibility into the IPv6 address space using the NXDOMAIN technique ranges around per level 40%, given that operators carefully maintain their reverse DNS zones. Further considering our prior results (see Section 9.3), well-maintained zones are more common for IPv6, for which less than 3% of all rDNS requests are not being answered authoritatively. In addition, it also indicates that dedicatedly querying all authoritative servers of a zone during enumeration is not strictly necessary. While it increases the result set for some zones, the additional overhead stands in no relation to the 1.79% of zones which could be enumerated more.

### 9.4.4 CNAMEs and Delegations

Previously, we observed that CNAMEs are mostly used in in-addr.arpa (see Section 9.3). They are used to delegate rDNS authority for networks smaller than the minimum rDNS zone size, i.e., smaller than a /24. Furthermore, we noted that requests to in-addr.arpa show a higher rate of SERVFAILs than requests to ip6.arpa. If our active rDNS datasets are




(a) Observed termination reasons/rcodes.

(b) Zone Delegations (CNAMEs in light blue).

Figure 9.6: rcodes and delegation steps in ip6.arpa and in-addr.arpa.

indeed viable datasets, then we should find evidence of these artifacts in our active traces as well. Hence, we look into how delegations occur for rDNS, and, how many delegations are broken or excluded from our study.

**rDNS Zone Delegation** To investigate delegation in rDNS, we build a trie[3] from the gathered reverse zones. We first sort the zones by corresponding prefix size, and then add them to the trie. Sorting them before adding them to the trie ensure that we do not add a longer prefix before we add the covering shorter prefix. For each input zone, we check if a less specific prefix exists in the trie. If it exists, we check if the authority section for the associated domain is the same. If the zone in the authority section changed, we found a delegation for the current prefix length. For terminal records, we also check if the zone reported in the authority section is a well-formed PTR zone, either under ip6.arpa or in-addr.arpa (depending on the zone we evaluate). If it is not the case, then it is not a delegation, but a CNAME for a terminal record.

For in-addr.arpa delegations happen consistently (see Figure 9.6(b)). The /8s prefixes are delegated to RIRs (and some Internet adopters who received large prefixes (Richter, Allman, et al., 2015)). Each /8 prefix is then split by the RIRs and delegated to LIRs in smaller blocks, which are further delegated to end-users and small network operators. This regular pattern extends down to the terminal records, where we find a surprisingly high number of delegation attempts, as well as 6.2 million CNAME records. Indeed, this corresponds to 0.51% of all 1.21 billion in-addr.arpa records being CNAMEs, close to the expected 0.71% of CNAME responses (see Section 9.3). Moreover, looking at the zones to which CNAMEs point, most of the target-zones (92.85%) have more than one CNAME pointing to them. Confirming and conforming to the purpose of CNAMEs in in-addr.arpa: Delegating rDNS for networks smaller than a /24, as suggested by RFC2317 (Eidnes et al., 1998).

In ip6.arpa, delegations mostly occur for the most common prefix lengths, which are /32s, /48s, /56s, and /64s. As to be expected, it relates closely to the more structured addressing

[3]Tries are efficient data-structures for prefix-based lookups, i.e., determining if a covering prefix to an element already exists in the data-structure (Gog and Venturini, 2016).

policies that became possible with the larger address space of IPv6. While a large operator might have to use several smaller prefixes collected from various organizations for IPv4 (Krenc and Feldmann, 2016), now a single IPv6 prefix is sufficient per region[4]. In turn, it causes ip6.arpa to be delegated primarily for larger prefixes.

Following IPv6 addressing best practices, we expected most delegations for /48s and /56s prefixes, as /64s are the suggested maximum prefix length for a subnet and the prefix-length that should be assigned to an interface (IAB and IESG, 2001; Velde et al., 2008). We would not expect /64s to be individually delegated, as a customer with multiple subnets would receive a /48 or /56 instead. However, we find that the total number of delegations actually increases from /48s to /64s, where it peaks. Furthermore, we even encounter delegations for prefixes more specific than /64s, peaking at the corresponding 4-nibble-block boundaries. Surprisingly, a high number of CNAMEs for terminal records exist, which we did not expect due to the better delegation option in ip6.arpa, due to the per-nibble zones.

Of these CNAMEs, 87.81% belong to the DHCPv6 range of a single operator. This operator uses CNAMEs to point PTR records from a full /96 representation in the ip6.arpa zone to another zone of the form ip6.arpa-suffix.ip6.dhcp6.operator.tld. Fiebig et al. already briefly mentioned such setups (Fiebig, Borgolte, et al., 2017). Of the remaining 12.19% most (80.77%) point to names in in-addr.arpa, to ensure a coherent addressing in dual-stack scenarios. Consequently, this is an indication of an "IPv4 first" policy employed by operators: Operators first deploy IPv4, and then roll out IPv6 on top, leveraging CNAMEs to ensure consistency through-out the network. Yet, IPv4 remains the leading technology, even though the setup is dual-stack.

Relating these numbers back to Section 9.3, we find that CNAMEs are slightly more common than expected, constitution 0.22% of the dataset. However, if we consider the single operator as an artifact, and exclude her, we arrive at the expected low CNAME density of 0.02%

**SERVFAIL in the Active Traces** SERVFAILs are much more frequent for in-addr.arpa than for ip6.arpa (see Section 9.3). Indeed, we find corroborating evidence for this in the active datasets. For in-addr.arpa, 3.40% of zones at the /16 level, and 4.87% of zones at the /24 level result in SERVFAIL (see Figure 9.6(a)). In contrast, for ip6.arpa, we only find small amounts of SERVFAIL for /32s and /48s, totaling 2.14% of all /32s, and 1.02% of all /48s. Again, we attribute this to the fact that ip6.arpa has not been in use for as long as in-addr.arpa, and, in turn, had far less time to accumulate broken delegations. It may also be the effect described by Phokeer et al. (Phokeer et al., 2016), who found more broken IPv4 reverse delegations in developing regions.

[4]Technically, one prefix would suffice globally, however, it is consider good practice to allocate a prefix from the RIR for the region where it is being used.

### 9.4.5 Summary

In this section, we validated and revisited two techniques for actively collecting rDNS datasets. We find that they provide valid datasets which can be used by researchers. Specifically, investigating the in-addr.arpa dataset, we conclude that operators do not only maintain well, but also actively populate their rDNS zones. Moreover, comparing our dataset with the popular CDN dataset, we show that the ip6.arpa dataset contains mostly systems not covered by the CDN dataset, thus rendering the two highly complimentary. In turn, it is an interesting tool to investigate parts of the Internet not already covered by the CDN dataset. Finally, we verified and confirmed observations from passively collected rDNS traces (see Section 9.3), which further underlines the consistency of the evaluated techniques.

## 9.5 Summary

In this chapter, we revisited existing measurement wisdom on the use of rDNS. Contrary to prior work (Cormack, 2007; Gao et al., 2013), we find that rDNS zones are commonly well maintained. Furthermore, our results show that forward confirmation by mail servers is not the predominant use case of rDNS anymore. Instead, rDNS usage pattern indicate that it is most commonly used to enrich logs to improve readability.

To underline our findings, we conduct a close investigation of IPv6 rDNS usage, and compare to the use of rDNS for IPv4. In addition to confirming prior assumptions, we find strong evidence for an "IPv4-first" approach across the Internet, i.e., operators still plan and build IPv4 infrastructures first, and then deploy IPv6 later on.

We also validated previously presented techniques to obtain active rDNS datasets. In particular, we have shown that they indeed provide a meaningful and viable dataset for research, and we have established an upper bound for the rDNS space that they cover. Given our insights on how reverse DNS is being used, we confirm that the technique presented in Chapter 8 is a viable research tool to investigate IPv6 deployment practices.

# 10

# Observations on Security and Misconfigurations via rDNS Datasets

After introducing our technique for collecting IPv6 datasets in Chapter 8 and investigating key-issues on the validity of rDNS as data source in Chapter 9, we now present various observations that become possible by using our rDNS scanning technique.

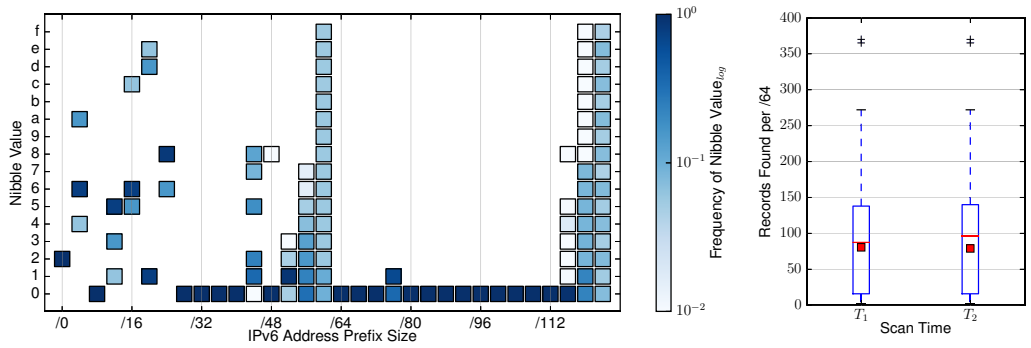
## 10.1 Numbering Policies in a Global SaaS Platform

To demonstrate the measurement opportunities that arise from our technique we investigate the IPv6 numbering policies and deployment practices of a large SaaS platform. We selected the prefixes of this operator based on its IPv6 announcements collected via bgp.he.net. To obtain further ground-truth, we also collected the PTR records of all IPv4 prefixes announced by the operator's autonomous system (AS) from bgp.he.net. We took two measurements, $T_1$ and $T_2$, two weeks apart in September 2016.

Figure 10.1 shows an overview of the allocation policy of the operator. Specifically, the operator uses three /32 IPv6 prefixes, with one being used per region she operates a data center in (see Figure 10.1(a)). In each region, the operator splits her prefix via the $40_{th}$ to $44_{th}$ bit of addresses. IPv6 networks used by network-edge equipment for interconnectivity links between different regions are distinguished by an *8* at the $48_{th}$ to $51_{st}$ bit, instead of *0*, which is used by all other prefixes.

Another interesting part of the addressing policy are the /48 networks the SaaS provider allocates. Here, we can see that networks are linearly assigned, starting with *PREFIX:0000::/48*, thus creating pools of /64s for various purposes. Furthermore, with /48s being linearly assigned, we discover that prefixes with higher indexes have not yet been assigned. Similarly, the same assignment policy holds for hosts in /64s networks, which is further supported by the distribution over the last twelve bits of used addresses.

A third aspect of the operators assignment policy is documented in Figure 10.1(b). Specifically, the boxplots show the number of hosts per /64 prefix in the operators networks. For both measurements, we only observe two /64 prefixes with significantly more than 250




(a) Density in SaaS provider at $T_2$ (b) Addr. per /64

Figure 10.1: Overview of address allocation in the SaaS cloud provider's network.

hosts. A closer investigation of these networks reveals that they are related to internal backbone and firewalling services spanning multiple Points-of-Presence, as indicated by the PTR naming schemes of the obtained records. Apart from this change, we do see a slight increase in the number of hosts per network in the median, but not the mean. An interesting side-note is that the IPv6 PTR records appear to be manually allocated by the operator's network staff. We do arrive at this conclusion because we encountered various records with typographical mistakes in them. This indicates that key operations in the SaaS provider's network are handled manually.

When comparing of the datasets with the corresponding IPv4 PTR sets, we discover that the diversity of records is far higher in the IPv4 set. There, various second-level domains can be found mixed together, which we did not encounter for the IPv6 set. Various naming schemes for infrastructure hosts are also present. For example, we discover that the customer-facing domain of the operator is being used for infrastructure services. However, it has apparently been disbanded with the growth of the organization, as we also discover infrastructure specific second-level domains. For the IPv6 set we only discover one infrastructure domain. The same observations hold for the overall naming schemes, with IPv6 being significantly more consistent. Our conjecture is that the operator made an effort in keeping a consistent state when finally rolling out IPv6, while IPv4 is suffering from legacy setups introduced during the company's growth.

Finally, the last striking observation is that the PTR records returned for IPv4 and IPv6 reverse pointers do not resolve to valid A and AAAA records themselves. A direct consequence is that, for this network operator, the technique proposed by Czyz et al. (Czyz, Luckie, et al., 2016) is not applicable. We conjecture that the operator chose this setup because she does not require forward lookups, yet wants traceroutes and other reverse-lookup related tools, especially distributed logging, to show the FQDNs.

In summary, the key observations from this case study are, that: (i) Even large operators use manual processes to allocate IPv6 addresses, (ii) Due to missing legacy requirements, and

many smaller network segments in IPv4, IPv6 addressing can be more structured, but, (iii) The IPv6 topology still caters towards classical IPv4 needs, e.g., concerning the number of hosts per network segment.

## 10.2 Encoding IPv4 in IPv6

In the previous section, we observed an SaaS providers' numbering schemes. We previously discovered that operators follow an "IPv4 first" approach, which aligns with our observations from Section 10.1. Indeed, we find that 80% of hosts in the ip6.arpa dataset are in /64 networks with three or less hosts, 90% in networks with four or less hosts and 99% of all addresses are in networks with less than 40 hosts.In general, IPv6 rDNS datasets allow interesting observations on addressing. Leveraging our two datasets, we evaluate how IPv6 addressing is derived from IPv4 addressing in dual-stack cases.

Specifically, we leverage that CNAMEs in ip6.arpa are commonly used to preserve consistency with in-addr.arpa. There, for simplicity reasons operators usually encode the IPv4 address in the IPv6 address. Indeed, we discover several common conversion schemas: The most common one is adding the last octet of the IPv4 address as the suffix to the IPv6 address in decimal (63.86%). It is followed by the same approach, but encoding the last octet as a hexadecimal digit (34.88%), then followed by encoding the whole IPv4 address in hex, and inserting it into the IPv6 suffix.

A numbering technique that we were surprised to not see is encoding the full IPv4 address in the IPv6 suffix in decimal instead of hexadecimal. For each octet of the IPv4 address, one 4-nibble block of the corresponding IPv6 address is filled. Indeed, specifically looking for it in our raw dataset, we identify various operators that leverage this technique for several thousands of hosts. However, these operators do not use CNAMEs to point from the in-addr.arpa record to the ip6.arpa record or vice versa. Still, both records resolve to the same FQDN. Hence, to perform a global analysis on this matter, we would have to perform A and AAAA lookups for all FQDNs returned for ip6.arpa and in-addr.arpa PTR requests. Even though such a full-scale analysis of numbering in dual-stack scenarios is certainly possible with our dataset, it is out of scope for this thesis.

## 10.3 Preservation of rDNS for NAT64

During our initial evaluation of the data gathering technique we identified a case where rDNS for clients is adjusted for an IPv4 to IPv6 migration. Specifically, we found a /96 prefix of a Japanese operator that mapped the entire IPv4 rDNS space into the last 32 bits of a /96 network using CNAMEs.

The observed prefix is a NAT64 prefix of that operator. In NAT64 (Bagnulo, Matthews, et al., 2011), an operator uses one of her IPv6 prefixes to map the IPv4 space into it. IPv6

clients can then connect to IPv4-only hosts by contacting the corresponding address, which is translated in a stateful manner on a middle box. Clients automatically receive the right IPv6 address from the NAT64 range for an IPv4-only host by using DNS64 (Bagnulo, Sullivan, et al., 2011). A providers recursive server implementing DNS64 automatically adds a AAAA record for DNS queries which do only return an A record answer.

However, neither of these RFCs, nor RFC7269 (G. Chen et al., 2014), which details "NAT64 Deployment Options and Experience", provide any conclusive information on how to preserve reverse lookups in case of NAT64/DNS64. Therefore, if an operator uses NAT64, hosts relying on this transitioning technique lose the ability to translate addresses of remote IPv4-only hosts to names. Recall that rDNS is mostly used by systems and middle boxes for information on participants in ongoing connections (see Section 9.3).

To tackle this problem, the operator mapped the entire in-addr.arpa space using CNAMEs to the deterministic range of her NAT64 prefix. Note, that contrary to DNS64, this portion of the transitioning technology does not interfere with DNSSEC, as it only requires CNAMEs to be set in a zone controlled by the operator (the ip6.arpa zone for the NAT64 prefix), instead of the insertion of non-existing AAAA records in a foreign zone. In conjunction with our prior findings (see Section 9.3), it signifies that there is real-world demand for a contingency feature in this transition technique. Therefore, we suggest that the IETF reviews the idea of recommending the adoption of the technique observed with the Japanese ISP in future iterations of the NAT64/DNS64 RFC.

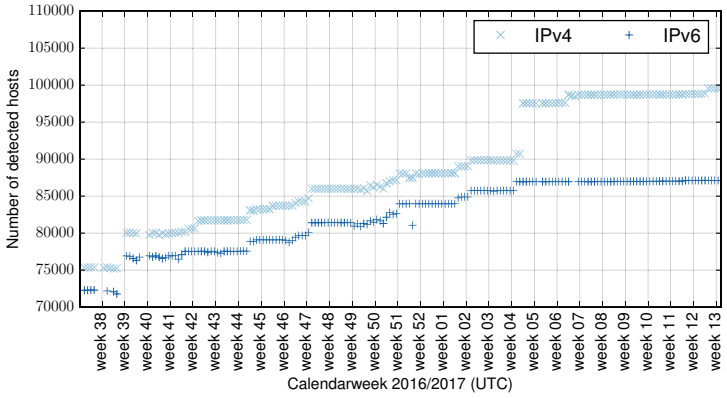



Figure 10.2: Number of IPv4 and IPv6 hosts in the SaaS provider's networks.

## 10.4 SaaS Provider: From IPv4 to IPv6?

Coming back to the SaaS provider from Section 10.1, we compare the provider's IPv4 and IPv6 hosts (see Figure 10.2). The number of hosts in the networks increases significantly by 5,000 hosts on the weekend before calendar week 39 in 2016. It has been caused by the deployment of a large number of infrastructure hosts, and it is also coincides with the announcement of a record sales quarter and a 25% revenue plus for that quarter.

Interestingly, the number of IPv6 hosts closely follows the line of IPv4 hosts, on average with around 3,000 hosts less. Indeed, this difference stems from the client facing systems of the operator being not yet IPv6-enabled. Through ip6.arpa PTR records, we find various infrastructure related second level domains: “net” (networking), “ops” (operations), “ib” (related to the Equinix Business Exchange Program (Equinix, 2017)), “oob” (out of band management), and “eng” (engineering) (see Figure 10.3). The sa.s.net entry is the same single record with a typographical error that Fiebig et al. already mentioned (Fiebig, Borgolte, et al., 2017). In addition to these infrastructure domains, we discover various customer-related third level domains in the in-addr.arpa dataset. These include “mta” (Mail Transfer Agent), “my” (a landing page for users of the product) and “login” (systems hosting the login pages of the platform).

Similar to the earlier increase in deployed systems in 2016, we encountered an even larger increase in calendar week 4 of 2017, with around 7,000 new hosts being deployed within just several days. However, this time, we do not observe a corresponding increase in the number of deployed IPv6 systems. We can attribute this even to initially 6,842, and, later, after calendar week 12 of 2017, 7,564 systems being added under the subdomain gslb.sitesaas.com. The abbreviation GSLB commonly stands for Global Server Load Balancing. It describes a technique were requests are globally distributed over the network of a provider, to equalize load on the provider’s systems (Hsu et al., 2008). In fact, judging from their FQDNs, these systems are co-located in four data centers across the globe in equal shares, with a point of presence in Chicago (chi), London (lon), Tokyo (tyo) and Washington State (was). Deploying GSLB for IPv4 may suggests two possible cases: either, client-facing IPv6 is not a top priority for the SaaS provider, or they plan to use the GSLB setup to aid the customer facing deployment of IPv6, as suggested by RFC6589 (Livingood, 2012).

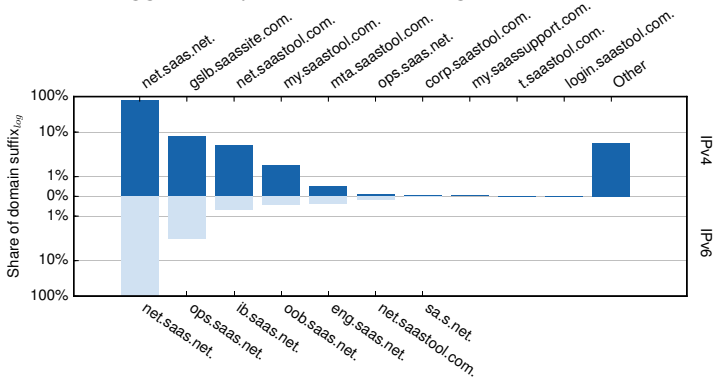



Figure 10.3: Share of tuples of third, second and top level domains in the SaaS provider’s network for IPv4 and IPv6 at the beginning of week 13, 2017.

## 10.5 Military Network Reconnaissance

Next, we evaluate what (possibly security sensitive) information can be obtained about networks that can not be probed actively. For this purpose we investigate the internal network

of a US military organization. To preserve the confidentiality of the concerned organization, we anonymize the presented data.

We specifically selected a US military organizational unit as these commonly employ strict perimeter firewalling. Furthermore, individual prefixes are commonly not routed, but only a larger covering prefix is announced to the GRT. Hence, for our evaluation of this not individually routed network a less targeted approach as for the previous case study is necessary.

The data which we analyze in this section has been obtained during the $\text{GRT\_SEED}_{80}$ experiment described in Section 8.5. From the obtained dataset we select all entries related to a single /48 within the range of the US military. Using the returned FQDNs we cross checked that: (i) No other networks contain hosts using the third and second level domain of that organization, and (ii) all hosts in the selected networks do use the same third and second level domain.

Using that dataset, we construct an overview of the logical topology in an automatic fashion, i.e., by representing the logical topology as an undirected graph. We create the graph in a bottom-up approach using peculiarities of IPv6 and the way the investigated organization assigns PTR records:

1. We create nodes for each host that we were able to enumerate and add them to the graph. We do this based on the assigned FQDN. This leverages that the operator assigns the same PTR record to multiple addresses assigned to the same host (multi homing).
2. Next, we add nodes for each /64, for which we found at least one host in the previous step. We assume that the operator adheres to the suggested minimum segment allocation of /64 in IPv6. If the operator diverts from this approach, an additional manual clearing process is necessary.
3. For each identified host we add edges connecting it to each /64 in which the host has an address.
4. We identify all hosts that have more than one adjacent edge. We assume that these are routers and mark them appropriately.

The result of this process for the selected military network can be found in Figure 10.4. We note that our heuristic for identifying routers works well. Indeed, the hosts tagged as routers commonly hold FQDNs commonly attributed to gateways or routers. It seems like the organization prefixes FQDNs for gateways with `gw`. However, some systems we identify as routers are prefixed with `srx`. This may indicate that these devices are Juniper SRX firewalls, a common gateway/firewall product of this US based vendor. We also note that one system is designated `gwborder.`, which resembles the term “border gateway”. We assume that this is the single point of outbound connectivity. Indeed, on a more general note, we find the topology of a network that adheres to industry standard best practices with a multi-layer security approach, separating internal systems via multiple gateways and firewalls from the network’s border.

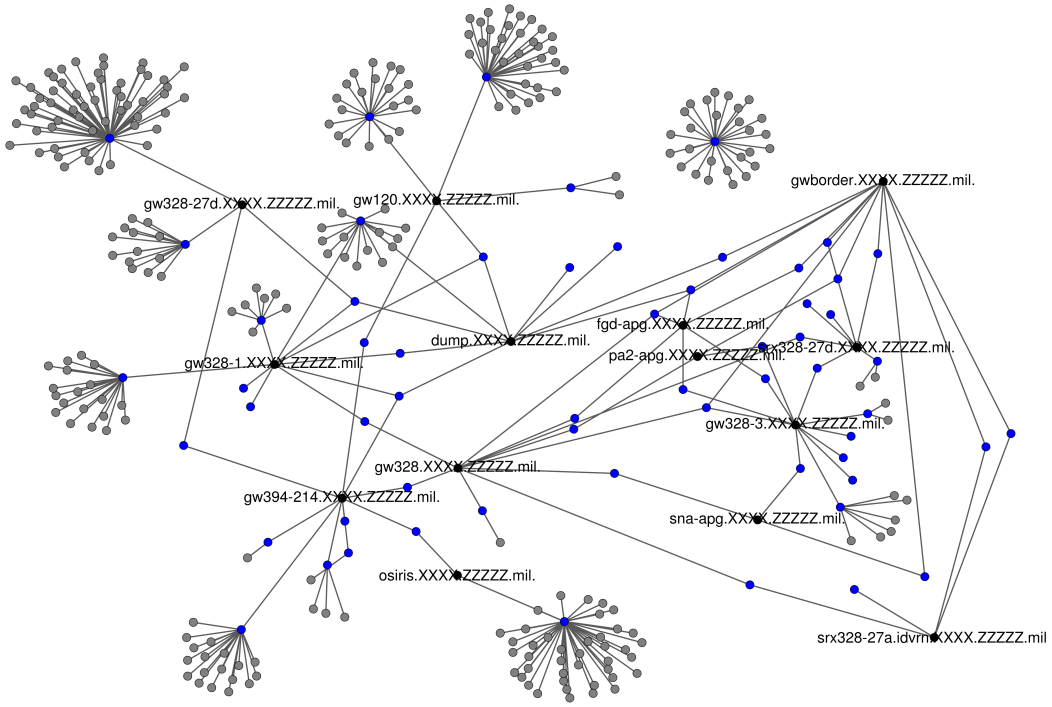



Figure 10.4: Overview of the military network's topology. Hosts are gray, /64 networks blue, and routers black. Labels have been included for routers only.

Regarding the IPv6 numbering policy, we find that the operator seems to use /64 networks for physical transfer segments. Single logical connections between gateway systems commonly share a dedicated /64. However, in some cases, we do find transfer networks that are reachable via more than one router. In addition to transfer networks we also find networks that are used for connecting a couple of internal systems. Judging from the FQDNs of these systems, networks are usually a mix of workstations and service providing systems. Furthermore, we find that various gateways are also connected to /64s that seemingly do not contain any hosts. Due to their topological location, adjacent to a single gateway, we assume that these are end-user networks hosting client systems that receive dynamically allocated addresses. Please note the host `srx328-27a.idvrn.` in the lower right of the graph, which is only connected to transfer networks (two of them forming a redundant connection to `gwborder.`) and a single network with no other hosts. Due to these characteristics, we assume that this host is a Virtual Private Network (VPN) gateway for the organization.

The topological information presented in this section could already be utilized by an attacker to plan and execute a targeted intrusion attempt. However, in addition, the selected FQDNs in the operators network allow for further observations on the operator's selection of software. For example, see Figure 10.5, we find various hosts named `ingest[0-9].` and `bigbrother[0-9].` in the network connected to the router `osiris.` (bottom middle

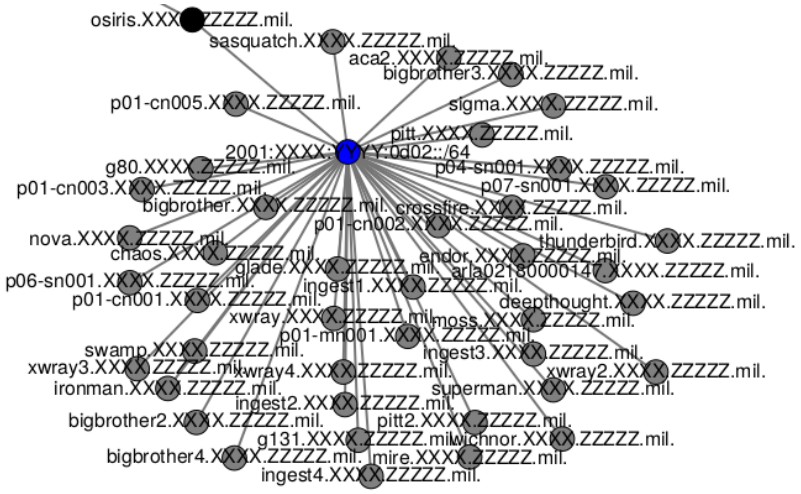



Figure 10.5: Excerpt from Figure 10.4: Used software being visible in reverse DNS entries.

in Figure 10.4). These hostnames are most likely related to the monitoring system "Big-Brother". This system used to be relatively popular in the late 1990's and early 2000's. Since then, it has been mostly replaced by more modern system (Massie et al., 2004).

Similarly, we find various hosts with verbose names in the larger network connected to `gw328-27d.` (top left in Figure 10.4). Judging from the FQDNs, this network is mostly occupied by service providing systems. Indeed, we find the following FQDNs: (i) `puppet-prod.`, indicating that the concerned system is a puppet master. Puppet is a remote orchestration and configuration management tool. From an attackers point of view, this host is a valuable target as it can usually execute code on all orchestrated systems. (ii) `krb5.`, `kdc-[a,b]`, and `ldap.`, are most likely the components of an authentication and authorization setup using LDAP and Kerberos (in any of the available software stack combinations). These hosts commonly regulate all authentication and authorization for a network, hence are also of interest to an attacker. (iii) `repo.`, a hostname indicating that the associated host provides repositories or software mirror services for the organization. Either way, an attacker with access to this system can spread malicious code easily throughout the organization. (iv) `smtp.`, is in itself not an important piece of information. However, this means that the aforementioned systems are located in the same network segment as a host possibly processing information from outside the network. From an attackers perspective, this is important information for planning attacks.

In summary, our observations demonstrate that sensitive information that can be obtained using our technique, even from well firewalled networks that can not be scanned with ICMP/TCP SYN scans or traceroutes.

## 10.6 Security: Exposed Router Backplane APIs

We also conducted an active scan of the enumerated addresses to showcase how our technique can be used to detect novel misconfigurations. For this purpose we ICMP probed a subset of hosts from the GRT_SEED$_{80}$ evaluation. Hosts replying to these requests were surveyed with a SYN only scan for selected ports known to be related to misconfigurations, among others, see Chapter 4. During these scans we adhered to best current practices concerning minimally invasive scans, see (Durumeric et al., 2013). Among others, we included port TCP/6666 in this list, as IRC servers are commonly run on this port. Initially, this was done to detect IRC based command and control servers with weak access protection.

However, our evaluation found various hosts that exposed this port together with TCP/22 (Secure SHell remote management), TCP/23 (Telnet remote management), and TCP/179 (Border Gateway Protocol (BGP)). In addition, the FQDNs returned by the reverse DNS pointers indicate Internet backbone links. Furthermore, the second level domains indicated that these systems were operated by top-tier networks. Based on the SSH version run, and an FQDN containing a product name, we were able to attribute these devices to a major US base network device vendor. In fact, these device are large Internet backbone routers, used for multi 100gE network routing.

```
root@*******% netstat -aln | grep 6666
tcp6       0      0  *.6666                 *.*            LISTEN
tcp4       0      0  128.0.0.1.6666         *.*            LISTEN
udp6       0      0  *.6666                 *.*
udp4       0      0  128.0.0.1.6666         *.*
```

Listing 10.1: Netstat entries on an affected vendor's router.

We were able to obtain authorized lab access to an affected device via a related research network. On the device we were able to determine that TCP/6666 was listening on an internal IPv4 address used for backplane intercommunication, see Listing 10.1. Indeed, we were able to verify that the service is not an IRC server, but an undocumented, internal services for intra backplane communication of the affected systems.

We contacted the vendor regarding this issue. The vendor reported that the issue is generally known to them. They reassured us, that at this point no attacks against the confidentiality, integrity, or availability of the associated service is known. Hence, they do not consider the issue of exposing this port as critical. However, we consider it highly unlikely that an internal backplane communication API being exposed to the Internet does not pose security risks. Yet, due to the scope of this work we refrained from a full fledged security evaluation of the affected software.

This case also provides interesting indications on why and how this internal service was exposed. The IPv4 range used by the vendor used to be reserved. This was changed in 2002 with RFC3330 (IANA, 2002). Up until recently, this vendor even used to filter this netblock by default (Aben and Romijn, 2011). With the introduction of IPv6 and the removal of

reservations for 128.0.0.0/16, an engineer at the vendor may have accidentally moved to make the service globally listen for IPv6, as it is listening on a public address for IPv4 as well. However, this remains speculation and could only be conclusively answered by the affected vendor.

In summary, the key observations from this case study are that: (i) Data collected with our technique can be used for subsequent security scans, (ii) We can identify new, so far unknown, security misconfigurations that are *exclusive* to IPv6 hosts, and (iii) Changes to and prior practices for IPv4 addressing policies may be related to security misconfigurations occurring in IPv6 networks.

## 10.7 Summary

In this chapter, we underline the benefits of our technique with different case studies that highlight previously overlooked industry needs and practices when deploying IPv6, which only become apparent utilizing our datasets. For instance, we document a case of—possibly—the start of a large-scale IPv6 deployment for end-user facing services by a major SaaS platform. The current situation indicates that they are using a GSLB approach to aid the transition, as suggested in RFC6589 (Livingood, 2012). In another case-study, an operator's need for rDNS in IPv6 transitioning techniques led to workarounds that preserve reverse lookups for NAT64/DNS64 setups. Therefore, we suggest that the relevant standard bodies, i.e., the IETF, approach this problem promptly to ease IPv6 adoption.

Furthermore, we are able to observe a misconfiguration which is novel in two ways: (i) It has been introduced by a *vendor*, and can not be *mitigated* by operators of the affected device due to limited *access* to the configuration, and, (ii) it is exclusive to IPv6.

# Part IV

# Discussion and Conclusion

# 11 Discussion

In this chapter we first provide a summary of this thesis. Next, we review and discuss our findings on the relationship between complex and misconfiguration based attacks. This leads towards a discussion of future challenges, that have to be tackled to mitigate security misconfiguration on the Internet. The chapter concludes with a discussion of obstacles and limitations encountered during the research projects presented in this thesis.

## 11.1 Thesis Summary

To answer our research questions, we use a multi-part approach. We first obtain initial understanding of misconfigurations and introduce the necessary terminology. This also entails an observational approach to misconfiguration incidents and their core aspects in Chapter 2. Subsequently, to contrast misconfigurations to sophisticated attacks, we present attacks on highly secure systems, i.e., multi compartment smartphones. Furthermore, due to the nature of these systems, and their specific usage scenarios, we attempt to gather credentials commonly used in multi-factor authentication (Bhargav-Spantzel et al., 2007). While our later use of this kind of technique demonstrate that it is a technically feasible threat, it also demonstrates its limited relevance for end-users.

In Chapter 4, we investigate one of the possible root-causes that leads to Internet services being misconfiguration prone: The underlying application level protocols. We performed an analysis of past protocol design and development practice over time. In our analysis we focus on protocols of various layers as defined by the Internet Engineering Task Force (IETF) in various Requests for Comments (RFC) published in the last decades. We opted to focus on protocols by the IETF instead of those by the International Telecommunication Union (ITU) or International Organization for Standardization (ISO). We did this because RFCs cover the most practically used protocols, including the Internet Protocol (IP) in version 4 and 6 (IPv4 and IPv6) and are freely and publicly available. Our analysis indicated various pitfalls that do indeed cater towards later service misconfiguration.

We demonstrate how conceptual weaknesses in the design of network protocols leads to implementations that are prone to being attacked due to misconfigurations. We then proceed by re-iterating good design practices as lessons learned that would mitigate these issues. As this is only practical for protocols in development, we also demonstrate how existing

exposed and vulnerable systems can be identified on the IPv4 and IPv6 Internet for subsequent mitigation. However, we would like to stress that our findings are not limited to network protocols by themselves. Instead, not only the underlying issues can be found in related technologies, but also our lessons learned do apply to other areas of network architectures.

Still, our insights on preventing misconfiguration prone services by designing protocols more carefully, are limited to future protocols, and do not provide an indication of how already deployed misconfigured services should be approached. Hence, in the second part of this thesis, we investigate how security misconfigurations can be detected on the Internet. Furthermore, more generally, we also take a look at the conceptual underpinnings of network scanning in general.

Following this literature driven approach, we investigated the situation of misconfigurations and misconfiguration mitigation in real world deployments. Hereby, we opted to utilize in-memory key-value stores as a case study. For this case study we observe the number of exposed systems on the Internet, i.e., systems on the Internet that one can connect to and retrieve process statistics from without being prompted for authentication data. For a more in-depth discussion of the specific method, utilized data sources and ethical considerations please see Section 6.3 in Chapter 6. Particularly, we choose Redis and memcached as object in the case study for the following reasons:

1. The problem of exposed instances for these two key-value stores has been made public to the operations community quiet recently and nearly at the same time (John Matherly, 2015a).
2. Even though the problems have been uncovered recently, they are known long enough for qualitative datasets to be available (ShadowServer Foundation, 2014).
3. Redis recently exhibited an additional security issue (Sanfilippo, 2015) that allows observations on the impact of such an incident on the number of exposed instances.
4. They are instances of novel database systems, which we identified as being most misconfiguration prone in Chapter 2 and Chapter 4.

We find that the results of this case study signify the importance of exhaustive scans, e.g., with zMap (Durumeric et al., 2013) for mitigating misconfiguration related issues on the Internet.

Despite the importance of exhaustive scans for the mitigation of security issues on the Internet, they will not remain available in the foreseeable future. Exhaustive security scans are not applicable on the vast address space of IPv6. This poses a significant threat to efforts aiming to make the Internet more secure. Hence, we do need new techniques to retain our capabilities to perform quantitative security research. Indeed, we have to find techniques that allow a pre-selection of possibly active IPv6 addresses for subsequent security scanning.

Prior work led to the realization that techniques for enumerating IPv6 addresses should be readily available. Otherwise, independent scientific reproduction of potentially obtained

results is nearly impossible. In our work presented in the third part of this thesis, we accomplish this, by overcoming various challenges, including the presence of automatically generated IPv6 DNS reverse zones and non RFC8020 (Bortzmeyer and Huque, 2016) compliant authoritative servers. Our performance driven evaluation of this technique indicates its general applicability for researchers.

However, our technique raises the question, if reverse DNS can actually be a viable datasource for collecting IPv6 address datasets. As the literature indicates that it is not, we revisit correctness and completeness of rDNS in general and in the context of our approach. We find that rDNS is indeed a viable datasource for collecting IPv6 datasets. Furthermore, we demonstrate its potential to obtain security sensitive information in various case-studies case studies.

## 11.2 Complex Attacks vs. Misconfigurations

In general, it seems misconfigurations are hardly considered a serious security threat and matter for research. While misconfigurations are quiet likely to be exposed and can have devastating impact when exploited, see Chapter 2, they commonly do not receive the same uproar as complex attacks. This is mostly related to misconfiguration related attacks being relatively simple: People should know how to prevent them. In fact, they do not.

This is different for complex attacks. While our attacks presented in Chapter 3 are feasible, we also find that it is highly unlikely that they are executed in real-world scenarios en large. Our evaluation of an example for a complex attack build to taint the security of high-profile security solutions. Our attack is feasible and provides reasonable results. However, it does not scale. The infection vector of abusing end-users' low awareness of permissions requires additional spear phishing. Furthermore, our attack has to be ported to each individual target device family. This does require access to skilled programmers, that can adjust the malware. Furthermore, for initially creating a weaponized malware, an attackers' organization would have to obtain further operation knowledge on image recognition, or attempt to exfiltrate hundreds of images from a mobile device.

Independent of which extraction technique the attacker chooses, either the available bandwidth and data transmission budget or the battery runtime of the target will suffer. However, both these parameters have been identified as major metrics for a possible device compromise by end-users (Kraus et al., 2015). Hence, an attacker would require additional resources to improve on these parameters for a successful use of this attack. This makes the large-scale application of this attack vector highly unlikely.

Complimentary to complex attacks we also find complex security mechanisms. In general, complex mitigations provide a good guarantee that the three security properties confidentiality, integrity, and availability are not tainted. Proposals such as MemGuard (Cowan et al., 1998), control flow integrity (Abadi et al., 2005) and position independent executables (PIEs) (Payer, 2012) aim at preventing the impact of simple buffer overflows all together.

We also found similar techniques in our analysis of protocol design in Chapter 4. There we group such protocols in class three, complex security. However, we find that protocols of this class are commonly not adopted. Misconfigurations occur when older protocols that provide similar *functionality* with weaker *security* and present misconfiguration traps are used instead of the secure but complex protocol. This already foreseeable in Chapter 3, where our complex attack can be mitigated by a simple strip of duct-tape. Furthermore, our target platform is rarely used by high-profile targets due to its usability restrictions (Dreßler et al., 2014). This trend continues in Chapter 4, where we see that, e.g., SNMPv3 is less frequently used than SNMPv2, leading to numerous instances with weak credentials on the Internet. A similar trend can be observed for plaintext telnet. Technically, this protocol has been long science abandoned due to missing encryption. Yet, modern IoT systems commonly use it, leading to countless unprotected systems on the Internet (Pa et al., 2015).

Already in Chapter 2, we detail that large scale data compromises due to simple misconfigurations are a common event on the Internet. Our systematic investigation of protocol design revealed that misconfiguration traps can oft already be found in the protocol standards that then lead to misconfiguration prone implementations. This makes the mitigation of misconfigurations extremely difficult, as the underlying standards have to be modified, before implementations can reflect the necessary changes. Even then, as documented in Chapter 6, the mitigation of deployed misconfigured services is hard, as it requires the cooperation of various parties.

In summary, we find that complex attacks often have simple mitigations, while an attacker needs significant skill and resources to exploit them. At the same time, seemingly simple issues, like misconfigurations that can be exploited by unskilled attackers, are hard to mitigate, and often prompt for complex mitigations. In turn, these complex mitigations are hardly adopted, promoting further traps for misconfiguration and simple issues in implementation, deployment and operation of Internet connected services.

In fact, we note that at the time of writing Google's chief of Android security just commented on this issue at an industry conference (Ludwig, 2017). He reports that Google's internal data demonstrates that complex attacks are a non-issue for smartphone security. He illustrates this point with the stagefright vulnerability (Drake, 2015), an issue in modern Android devices that allows for remote compromises. Since the vulnerability was published, not a single attack exploiting this vulnerability was observed. Indeed, a similarly critical vulnerability, MasterKey, e.g., see (Lindorfer et al., 2014), peaked at merely 8 infections per 1 million devices when the vulnerability was published. Instead, Ludwig finds that malicious applications exploit devices' users: By abusing their trust, or, ingenuous and naive use of apps, these malware groups generate revenue. These assertions support our observations in this thesis.

## 11.3 Future Challenges

The previous section demonstrates that misconfiguration based security incidents are among the hardest to mitigate. In the past there have been various approaches towards the problem of human error in software deployments. A common approach to this issue are new methods for deploying and maintaining software. These usually try to remove misconfiguration traps by introducing simple, modular, and automated software installation, configuration and updating.

The recent past has seen various technologies that changed the way we handle software. The most iconic change here are "App stores" on mobile and PC platforms, which provide coordinated access to programs and security updates. At the same time configuration options are streamlined, leading to a more stringent user experience. Technically, this follows the concept of packet managers, which have been available on Linux distributions for decades. However, these new concepts are less simplistic than established packet managers. Contrary to these established concepts, applications in mobile app stores usually contain shared libraries as well as the main application, while traditional packet managers carry independent packets for shared libraries. These concepts have been partly transferred to server side applications, e.g., with tools like Docker. These tools promote a new way of deploying software. Yet, while these concepts technically reduce the surface for misconfigurations, they also introduce new misconfiguration opportunities. With, e.g., Docker, a single misconfiguration can now "spread" to all instances that utilize a Docker container template. Furthermore, while Docker technically reduces the effort needed to keep software up-to-date, the shared libraries found in containers have to be as well maintained as the main software for which they have been installed. This problem space can also be found in various cloud scenarios, with the widely available and shared templates for virtual machines offering specific services.

Another revolution in the field of system and network engineering is the ever increasing trend towards more automation in software deployments. This trend has been described with various terms, including Site Reliability Engineering (SRE) and DevOps. It is characterized by an inherent drive towards automating classical operations tasks. In a 2016 publication, several Google employees claim that this concept was designed during the early days of Google. The by then fast growing Google Inc.'s operational needs were addressed as programming challenges by a newly founded team consisting of operations people and programmers, mostly holding PhDs, alike (Petoff et al., 2016).

SRE, or DevOps, shows great promise to reduce the occurrence of misconfigurations in installations. Larger organizations, which are more common to employ configuration and deployment automation, seem to be less likely affected by misconfigurations. This effect has also been observed by Lichtblau et al. in their 2016 work on network operators originating unwanted traffic. They find a correlation between an operator's traffic volume at an IXP and the fraction of that traffic that is indeed unwanted due to insufficiently configured filters on routers (Lichtblau et al., 2016).

Despite the promise automatisation approaches currently show, they also allow for the amplification of human error. The data presented in Chapter 6 suggests that several operators accidentally expose key-value stores on the Internet due to issues with their automated configuration management systems. This observations is supported by the high density of affected systems in the relevant network segments presented in that chapter.

Automatisation and configuration management are, despite their shortcomings, promising research objectives for reducing misconfiguration traps in server software. However, the IoT has lead to a landslide development: By now, an uncountable number of network connected devices is being deployed to homes, businesses and factories around the globe. Following our analysis from Chapter 4, these devices commonly utilize functionality oriented protocols, e.g., telnet. All the while, these devices come with weak default credentials and outdated software, seldom providing sufficient software update mechanics (Pa et al., 2015). This has already moved these devices in the focus of attackers, giving rise to so far unprecedentedly large botnets (Mansfield-Devine, 2016).

The aforementioned developments bring a new set of questions for misconfigurations, accidentally exposed systems, and insufficiently updated systems. It is common for self proclaimed experts to call for guarantees from vendors that their systems will receive updates. However, it can not always be expected that the entity who bought a systems does *consent* to system updates.

In general, the concept of IoT transmutes the issue of security misconfigurations to a whole new level. IoT introduces two new groups to system administration tasks that were never meant to operate systems. First, programmers: In the age of IoT, incidents due to misconfigurations by the vendor, which can not be mitigated by users will become more frequent. We provide a first case-study of such an incident in Chapter 10. The second group is end-users: With the introduction of IoT, computerized systems providing Internet services are pushed into every household. End-users are forced to take up the associated system operations tasks. And while vendors try to do their best to provide usable interfaces for these tasks, the underlying concepts can hardly be broken down enough.

In summary, even with this thesis, the full extent of the problem space remains insufficiently defined. Hence, future research should be first directed towards identifying the most pressing issues in the field, before developing new solutions that possibly do not address the *right* problems.

## 11.4 Limitations

As every scientific work, this thesis has limitations. Some were the result of conscious choices, other were only seen in hindsight. Hence, here, we briefly detail limitations of our work, why we opted to accept them, and why they do not threaten the validity of our approach.

### Complex Attacks vs. Misconfigurations

In Chapter 3 we introduce two distinct attacks using high resolution smart-phone cameras. These attacks are both designed to facilitate a better understanding of high resolution smartphone cameras' impact on mobile device security. We do acknowledge that we did not weaponize our attacks, i.e., our attacks were only conducted in a laboratory environment. However, Xu et al. already demonstrated a reflection keylogger using telescopes with automated offline processing in 2013 (Y. Xu et al., 2013). We demonstrate that these attacks are also feasible using facial reflections abusing the front-camera of a victim's smartphone. The engineering challenge of porting their (non published) code to a smartphone would have brought no additional insights. For the same reasons we did not evaluate all performance metrics of the keylogger, e.g., accuracy, precision over different subjects etc. Fingerprint extraction was also not weaponized, again because the focus of our study was on demonstrating attack feasibility using smartphone cameras.

### Protocol Definitions as Misconfiguration Facilitating Factors

Our work on protocol design based misconfiguration traps in itself includes a conclusive literature study with a specifically simple systematization of protocol design paradigms. Still, it contains various limitations, mostly due to a deliberate restriction of the studies scope of the study.

The most obvious limitation is the limited set of protocols. However, with thousands of network protocols being currently in use, an exhaustive analysis is out of scope. We do acknowledge that a more exhaustive, automatic, analysis of protocol issues would be beneficial, and currently treat this as a matter of our ongoing research interests. Furthermore, our categorization is deliberately simple and straight forward. We do acknowledge that other ways of systematizing misconfigurations and their related protocols exist, e.g., by first systematizing the misconfigurations. Furthermore, the work would have benefited from more empirical observations on the identified misconfiguration prone protocols.

Despite the outlined limitations our systematization highlights common traits in protocol design that lead to implementations prone to misconfiguration. Furthermore, our work permits us to draw conclusions for future protocol design.

### A Case Study on Exposed Key-Value Stores

The limitations in our study on exposed in-memory key-value stores are mostly introduced due to legal and ethical implications in the research environment our host institution provides. Furthermore, additional in-depth scans for past events are not possible. The study utilizes a historic dataset on exposed Redis and memcached servers supplied by a third party. The third party opted to aggregate the dataset on a per AS (Autonomous System) level. While we are able to obtain valuable insights from this dataset, our results could have

been strengthened by access to per IP address datasets for each day. An example for such insights can be found in our observations on snow-shoe spamming in Section 6.5. This could have been accomplished by performing the scans ourselves. However, our research institution recommends not to conduct such research on a large scale. The maximum extent which was accomplishable were the limited scans we conducted to closer investigate single operators.

### Global Collection of IPv6 Scan-Targets From DNS

Our work on collecting global IPv6 datasets from IPv6 PTR records was focused on exploring the feasibility of extending a known technique to a global scale. The publication that is the basis of Chapter 8 has been written in an early stage of the project, as it became clear that the technique shows great promise. Our main objective was making the technique itself available to the research community at large. While we continued our work on this topic, we found various limitations and improvements. These have been, if feasible, addressed in our publications of the research tools available at: `https://gitlab.inet.tu-berlin.de/ptr6scan/toolchain` The full-scale evaluation was then conducted in an additional project, which we document in Chapter 9.

The study presented in Chapter 9 found limitations in the tools and experimental setup used for the study presented in Chapter 8. We found out that the utilized version of GNU parallel, a tool for parallelizing program calls, had a bug that would lead to high disk utilization and unreasonably high disk space usage. As we did not include sufficient accounting for possibly lost data, we can only conjecture on the actual impact of this. Our later reproduction of the study using similar tools however indicates that this limitation did not have a significant impact. Furthermore, our initial methodology did not sufficiently account for empty terminals, i.e., CNAMEs. This problem has been addressed in the updated version of our toolchain. Furthermore, we opted to enumerate the IPv6 PTR tree in the following step sizes: 32, 48, 64, 128. We later migrated our methodology to 4 nibble steps. This increases the accuracy of our technique in the light of dynamically generated zones smaller than /64, e.g., as shown in Figure 8.3(b). The last limitation of our work has been brought to our attention during the anonymous peer review process of the associated publication. One of the reviewers suggested that we should have considered if different authoritative servers responsible for the same zone handle the zones differently, see Sub-Section 8.4.1 in Chapter 8.

In addition to this, we see limitations in our experimental setup. Specifically, we could have used more hosts and IP addresses to conduct our scans. In contrast to a single host, this would have significantly increased throughput, reducing the overall time needed to do a full measurement. This is most interesting for further work. Due to the relatively long time it takes to perform a full measurement, we do not provide a historic dataset. Furthermore, our work should have considered a deeper analysis of network topologies and the security state of detected hosts. However, we considered this out of scope for the work at hand, as it is not directly necessary for answering our research questions. Along these

lines it would also have been beneficial to compare our obtained dataset, especially our findings on dynamically generated networks, to the dataset used by Foremski, Plonka and Berger (Foremski et al., 2016; Plonka and Berger, 2015). However, we did not have access to this dataset. As with the limitations in the toolchain, these have been addressed for the study presented in Chapter 9.

In summary, there are either good reasons for limitations in our work, or, we can show that these limitations do not have a significant impact.

# 12
# Conclusion

This thesis provides initial answers to what misconfigurations are, an overview of how they are currently measured and mitigated, and how we can measure them on the IPv6 Internet. In our observations on misconfiguration mitigation, we note that misconfiguration based security issues are among the hardest to mitigated (Part II). Furthermore, we find that misconfigurations are one of the security issues that do have the highest impact, especially on average users, and are the hardest to mitigate (Part II). With our work on collecting IPv6 datasets for subsequent security scans we allow the (partial) application of traditional scan and mitigation processes to IPv6 (Part III). Nevertheless, while now applicable to IPv6, the drawbacks of mitigating security misconfiguration related issues still hold. Hence, misconfigurations and their mitigation are an important research objective for the future. Specifically, we will have to investigate solutions on the systems and usability level, as well as on the process level.

We motivate our investigation of security misconfigurations with observations on the frequency and common features of security misconfiguration incidents in the last years. Subsequently, we contrast attacks based on security misconfigurations to more complex attacks, which are commonly perceived as a significant threat. While these do allow an attacker to target high-profile personnel and circumvent state of the art security technology, they require strong determination as well as significant technical capabilities and resources on the attackers side. This makes it unlikely, that large groups of end-users are affected by such attacks. This is a strong contrast to misconfiguration related issues which do not require access to special resources or expert knowledge in order to be exploited. In addition, our investigation of misconfigurations is further motivated by the observation that, even though our sophisticated attacks received international press coverage, e.g., (Darlene Storm, Computerworld.com, 2014), our subsequent user study found that end-users feel more threatened by operators' errors than by sophisticated attacks.

Our analysis of protocol design and misconfigurations finds that over the last decades various paradigms for designing protocols existed, even though they were never defined as such. While the early Internet catered towards a process that was founded upon mutual trust, the Internets subsequent commercialization lead to the introduction of fencing based security. Protocols were designed to function in secured, separated, networks. While these protocols were in deployment, issues with the previous trust based approach became apparent. When malicious actors started to exploit the open trust based design of earlier protocols, e.g., that

of SMTP by sending unsolicited bulk mail, protocol designers attempted to designing inherently secure protocols. However, while creating secure protocols, they also introduced complexity which ultimately proofed to be an adoption barrier. In contrast to this, current protocol design returned to less secure concepts, prioritizing functionality over security. Security is usually implemented by tunneling the associated protocol over a secured lower network layer, that is perceived as secure, e.g., tunneling a API without authentication or authorization via TLS encrypted, authenticated, HTTP.

Each of these protocol generations promotes implementations prone to misconfiguration in its own way. Earlier protocols usually lack sufficient mutual authentication, which has been commonly added to them in later revisions. The most common misconfiguration here is simply not activating these "new" features. With protocols relying on fencing for security, connecting them directly to the Internet without firewalls, or having configuration issues in the firewall rules are common issues. Complex security solutions are often not adapted in favor of their less secure predecessors, or have a configuration so complex, that simple errors lead to an insecure setup. Last, the new functionality driven approach fails—similar to earlier fencing based approaches—as soon as the additional security layer is omitted. While these are certainly not the only root-causes of wide-spread service misconfigurations we observe on the Internet, we demonstrate that it led to the current issues observed with, e.g., IPP and SNMP. From our observations we draft guidelines that, if adopted, will lead to more secure protocols: By focusing on simplicity in security functions and removing possible legacy features in existing protocols, misconfiguration opportunities and obstacles to operating services securely can be removed.

However, misconfiguration prone implementations still exist and misconfigured services are still being deployed on the Internet. Utilizing exposed in-memory key-value stores as a case study we investigate how we can identify hosts running misconfigured services on the Internet. We find that current mitigation techniques rely on exhaustive, brute-force security scans of the IPv4 Internet. Trusted entities, e.g., CSIRTs, subsequently notifying affected operators about misconfigured systems in their network segments. We note, that this technique is, in general, not overly successful in coercing operators to fix misconfigured services. Nevertheless, we also demonstrate that such scans allow the identification of systematic, e.g., due to common configuration management, misconfigurations in larger operators. These operators can then be directly approached, increasing the chance of successful mitigation. We also note that this method of approaching service misconfiguration is not feasible for IPv6 enabled networks, due to the larger address space of this protocol. Subsequently, we present a new method that allows security researchers to obtain and later scan a dataset of assigned IPv6 addresses. Our method works by enumerating reverse DNS zones. In this thesis we focus on an approach that utilizes the semantics of denial of existence records as defined in RFC8020 (Bortzmeyer and Huque, 2016). While our method does not fully solve the problem of security scanning in IPv6, it is a step forward. We illustrate the impact of our technique using three distinct case-studies. First, we focus on the opportunities of our technique for measuring networks. We showcase how our technique can be used to obtain insights on numbering and (physical) topology information of datacenters, using a global SaaS provider. Subsequently, we investigate how an attacker can use our technique to obtain

information, including used software, from networks that are protected against active probing. Last, we also investigate how our technique can be combined with security scanning. In fact, we find a so far unknown security misconfiguration, which is exclusively visible on IPv6 enabled hosts.

## Future Work

Our results demonstrate how misconfiguration are currently mitigated, and how setting traps can be avoided in further protocol specifications. However, the problem of human error remains. The seemingly obvious answer to this question would be a new approach to how software is deployed, configured and maintained, that removes opportunities for misconfigurations, allows for simple and modular software updates, and ideally applies to client as well as server software.

Indeed, the recent past has seen various technologies that changed the way we handle software. The most iconic change here are "App stores" on mobile and PC platforms, which provide coordinated access to programs and security updates. At the same time configuration options are streamlined, leading to a more stringent user experience. These concepts have been partly transferred to server side applications, e.g., with tools like Docker. These tools promote a new way of deploying software. Yet, while these concepts technically reduce the possibile pitfalls where misconfigurations can occur, they also introduce new misconfiguration opportunities. Hence, future research should be directed towards reducing the human factor in creating, maintaining and deploying software.

However, as discussed in Chapter 11 the IoT also brings new challenges for software deployment and especially maintenance. This is even more critical, considering that especially IoT devices commonly suffer from protocols we classify as functionality driven in Chapter 4, and have moved into the center of attackers' attention (Mansfield-Devine, 2016). We furthermore find, see Chapter 2, that even though misconfigurations already have a devastating impact today, the opportunity for far more critical incidents in the future exists. In summary, while this thesis provides first insights on the problem space and methodology to explore it, the field of preventing and mitigating misconfigurations still provides various research challenges. These will have to be addressed, in order to reach a more secure Internet.

# List of Figures

# List of Tables

# Bibliography


Abadi, M., Budiu, M., Erlingsson, Ú., and Ligatti, J. (2005). „Control-flow Integrity“. In: *Proc. ACM Conference on Computer and Communications Security (CCS)*, pp. 340–353 (cited on p. 143).

Aben, E. and Romijn, E. (2011). *The Curious Case of 128.0/16*. URL: https://labs.ripe.net/Members/emileaben/the-curious-case-of-128.0-16 (cited on p. 137).

Aboba, B., Tseng, J., Walker, J., Rangan, V., and Travostino, F. (2004). *Securing Block Storage Protocols over IP*. RFC 3723 (Proposed Standard). Updated by RFC 7146. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc3723.txt (cited on pp. 49, 52).

Adrian, D., Durumeric, Z., Singh, G., and Halderman, J. A. (2014). „Zippier ZMap: Internet-Wide Scanning at 10 Gbps.“ In: *Proc. USENIX Workshop on Offensive Technologies (WOOT)* (cited on p. 74).

Alarcón, R. and Wilde, E. (2010). „RESTler: Crawling RESTful services“. In: *Proc. World Wide Web Conference*, pp. 1051–1052 (cited on p. 60).

Alazab, M. and Broadhurst, R. (2017). „An Analysis of the Nature of Spam as Cybercrime“. In: *Cyber-Physical Security*. Springer, pp. 251–266 (cited on p. 118).

Alllman, M. and Paxson, V. (2007). „Issues and etiquette concerning use of shared measurement data“. In: *Proc. ACM Internet Measurement Conference*, pp. 135–140 (cited on p. 91).

Allman, M. and Ostermann, S. (1999). *FTP Security Considerations*. RFC 2577 (Informational). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc2577.txt (cited on p. 42).

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al. (2010). „A view of cloud computing“. *Communications of the ACM*, 53(4), pp. 50–58 (cited on p. 56).

Assolini, F. (2012). *The Tale of One Thousand and One DSL Modems*. Kaspersky Lab (cited on p. 58).

Atikoglu, B., Xu, Y., Frachtenberg, E., Jiang, S., and Paleczny, M. (2012). „Workload analysis of a large-scale key-value store“. In: *ACM SIGMETRICS Performance Evaluation Review*. Vol. 40. 1, pp. 53–64 (cited on pp. 58, 77, 78).

Atkins, D. and Austein, R. (2004). *Threat Analysis of the Domain Name System (DNS)*. RFC 3833 (Informational). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc3833.txt (cited on p. 100).

Atkinson, R. (1995). *Security Architecture for the Internet Protocol*. RFC 1825 (Proposed Standard). Obsoleted by RFC 2401. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc1825.txt (cited on p. 51).

Aviv, A. J., Sapp, B., Blaze, M., and Smith, J. M. (2012). „Practicality of accelerometer side channels on smartphones“. In: *Proc. ACM Annual Computer Security Applications Conference (ACSAC)*, pp. 41–50 (cited on p. 20).

Aweke, Z. B., Yitbarek, S. F., Qiao, R., Das, R., Hicks, M., Oren, Y., and Austin, T. (2016). „ANVIL: Software-based protection against next-generation rowhammer attacks“. In: *Proc. ACM Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 743–755 (cited on p. 2).

Backes, M., Durmuth, M., and Unruh, D. (2008). „Compromising reflections - or - how to read LCD monitors around the corner“. In: *Proc. IEEE Security & Privacy* (cited on p. 20).

Bagnulo, M., Matthews, P., and Beijnum, I. van (2011). *Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers*. RFC 6146 (Proposed Standard). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc6146.txt (cited on p. 131).

Bagnulo, M., Sullivan, A., Matthews, P., and Beijnum, I. van (2011). *DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers*. RFC 6147 (Proposed Standard). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc6147.txt (cited on p. 132).

Balfanz, D., Durfee, G., Grinter, R. E., and Smetters, D. K. (2004). „In search of usable security: Five lessons from the field“. *Proc. IEEE Security & Privacy*, (5), pp. 19–24 (cited on p. 65).

Baraniuk, C. (2016). *Millions of Mexican voter records 'were accessible online'*. Accessed: 18.02.2017. URL: http://www.bbc.com/news/technology-36128745 (cited on p. 14).

Barnes, R. (2011). *Use Cases and Requirements for DNS-Based Authentication of Named Entities (DANE)*. RFC 6394 (Informational). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc6394.txt (cited on p. 64).

Barr, D. (1996). *Common DNS Operational and Configuration Errors*. RFC 1912 (Informational). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc1912.txt (cited on p. 96).

Barrett, D. J., Silverman, R. E., and Byrnes, R. G. (2005). *SSH, The Secure Shell: The Definitive Guide: The Definitive Guide*. O'Reilly Media, Inc. (cited on p. 55).

Barrett, R., Kandogan, E., Maglio, P. P., Haber, E. M., Takayama, L. A., and Prabaker, M. (2004). „Field studies of computer system administrators: analysis of system management tools and practices“. In: *Proc. ACM Conference on Computer Supported Cooperative Work*, pp. 388–395 (cited on p. 56).

Bellovin, S. M. and Cheswick, W. R. (1994). „Network firewalls“. *IEEE Communication Magazine*, 32(9), pp. 50–57 (cited on p. 45).

Bencsáth, B., Pék, G., Buttyán, L., and Felegyhazi, M. (2012). „The cousins of stuxnet: Duqu, flame, and gauss“. *Future Internet*, 4(4), pp. 971–1003 (cited on p. 2).

Bernstein, D. J. (n.d.). *How the AXFR protocol works*. URL: `http://cr.yp.to/djbdns/axfr-notes.html` (cited on p. 44).

Bernstein, D. J., Lange, T., and Schwabe, P. (2012). „The security impact of a new cryptographic library“. In: *Progress in Cryptology–LATINCRYPT 2012*, pp. 159–176 (cited on p. 65).

Bhargav-Spantzel, A., Squicciarini, A. C., Modi, S., Young, M., Bertino, E., and Elliott, S. J. (2007). „Privacy preserving multi-factor authentication with biometrics“. *Journal of Computer Security*, 15(5), pp. 529–560 (cited on p. 141).

Bhushan, A. (1971). *File Transfer Protocol*. RFC 114. Updated by RFCs 133, 141, 171, 172. Internet Engineering Task Force. URL: `http://www.ietf.org/rfc/rfc114.txt` (cited on p. 42).

Biermann, K. (2015). *Mit der Kamera Merkels Fingerabdruck hacken*. URL: `http://www.zeit.de/digital/datenschutz/2014-12/fingerabdruck-merkel-leyen-hack-ccc-31c3` (cited on p. 23).

Bikos, A. N. and Sklavos, N. (2013). „LTE/SAE security issues on 4G wireless networks“. *Proc. IEEE Security & Privacy*, 11(2), pp. 55–62 (cited on p. 52).

Binary Edge (2015a). *Data, Technologies and Security - Part 1*. URL: `http://blog.binaryedge.io/2015/08/10/data-technologies-and-security-part-1/` (cited on pp. 1, 3, 35, 59, 61, 63, 76, 77).

– (2015b). *VNC, image analysis and data science*. URL: `http://blog.binaryedge.io/2015/09/30/vnc-image-analysis-and-data-science/` (cited on pp. 60, 63).

Birrell, A. D. and Nelson, B. J. (1984). „Implementing remote procedure calls“. *ACM Trans. Computer Systems*, 2(1), pp. 39–59 (cited on p. 60).

Bishop, M. (2003). „What is computer security?“ *IEEE Security & Privacy Magazin*, 1(1), pp. 67–69 (cited on pp. 2, 11).

Black, D. and Koning, P. (2014). *Securing Block Storage Protocols over IP: RFC 3723 Requirements Update for IPsec v3*. RFC 7146 (Proposed Standard). Internet Engineering Task Force. URL: `http://www.ietf.org/rfc/rfc7146.txt` (cited on p. 49).

Blumenthal, U., Maino, F., and McCloghrie, K. (2004). *The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model*. RFC 3826 (Proposed Standard). Internet Engineering Task Force. URL: `http://www.ietf.org/rfc/rfc3826.txt` (cited on p. 55).

Blumenthal, U. and Wijnen, B. (1998). *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)*. RFC 2264 (Proposed Standard). Obsoleted by RFC 2274. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc2264.txt (cited on p. 54).

– (2002). *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)*. RFC 3414 (INTERNET STANDARD). Updated by RFC 5590. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc3414.txt (cited on p. 55).

Bodenheim, R., Butts, J., Dunlap, S., and Mullins, B. (2014). „Evaluation of the ability of the Shodan search engine to identify Internet-facing industrial control devices“. *Elsevier Journal of Critical Infrastructure Protection*, 7(2), pp. 114–123 (cited on p. 41).

Boe, M. and Altman, J. (2002). *TLS-based Telnet Security*. INTERNET-DRAFT draft-ietf-tn3270e-telnet-tls-06. Internet Engineering Task Force. URL: https://tools.ietf.org/html/draft-ietf-tn3270e-telnet-tls-06 (cited on p. 57).

Bollinger, G. (2015). „Securely Managing Your Networks With SNMPv3“. *CiscoLIVE! BRKNMS-2658*. URL: https://clnv.s3.amazonaws.com/2015/usa/pdf/BRKNMS-2658.pdf (cited on pp. 54, 55).

Borman, D. (1993). *Telnet Authentication Option*. RFC 1409 (Experimental). Obsoleted by RFC 1416. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc1409.txt (cited on p. 57).

Bortzmeyer, S. and Huque, S. (2016). *NXDOMAIN: There Really Is Nothing Underneath*. RFC 8020 (Proposed Standard). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc8020.txt (cited on pp. 98, 99, 108, 116, 124, 143, 152).

Botta, D., Werlinger, R., Gagné, A., Beznosov, K., Iverson, L., Fels, S., and Fisher, B. (2007). „Towards understanding IT security professionals and their tools“. In: *Proc. ACM Symposium on Usable Privacy and Security*, pp. 100–111 (cited on p. 56).

Bouillon, E. (2009). „Taming the beast: Assess Kerberos-protected networks“. *Black Hat EU* (cited on p. 54).

Boyen, X., Dodis, Y., Katz, J., Ostrovsky, R., and Smith, A. (2005). „Secure remote authentication using biometric data“. In: *Proc. Advances in Cryptology–EUROCRYPT*. Springer, pp. 147–163 (cited on p. 32).

Braun, H. and Rekhter, Y. (1991). *Advancing the NSFNET routing architecture*. RFC 1222 (Informational). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc1222.txt (cited on p. 45).

Breen, S. (2015). *What Do WebLogic, WebSphere, JBoss, Jenkins, OpenNMS, and Your Application Have in Common? This Vulnerability.* URL: http://foxglovesecurity.com/2015/11/06/what-do-weblogic (cited on p. 61).

Brocker, M. and Checkoway, S. (2014). „iSeeYou: Disabling the MacBook Webcam Indicator LED.“ In: *Proc. Usenix Security Symp.* Pp. 337–352 (cited on p. 19).

Bundesamt für Sicherheit in der Informationstechnik (2014). *Die Lage der IT-Sicherheit in Deutschland 2014*. Tech. rep. (cited on p. 15).

Callaghan, B., Pawlowski, B., and Staubach, P. (1995). *NFS Version 3 Protocol Specification*. RFC 1813 (Informational). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc1813.txt (cited on p. 49).

Cantelon, M., Harter, M., Holowaychuk, T., and Rajlich, N. (2014). *Node.js in Action*. Manning (cited on p. 60).

Carna Botnet (2013). *Internet census 2012: Port scanning/0 using insecure embedded devices* (cited on pp. 58, 63).

Carr, C. (1969). *Network subsystem for time sharing hosts*. RFC 15. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc15.txt (cited on p. 57).

Casado, M., Garfinkel, T., Akella, A., Freedman, M. J., Boneh, D., McKeown, N., and Shenker, S. (2006). „SANE: A Protection Architecture for Enterprise Networks." In: *Proc. Usenix Security Symp.* (Cited on p. 51).

Case, J., McCloghrie, K., Rose, M., and Waldbusser, S. (1996). *Introduction to Community-based SNMPv2*. RFC 1901 (Historic). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc1901.txt (cited on p. 48).

Case, J., Mundy, R., Partain, D., and Stewart, B. (2002). *Introduction and Applicability Statements for Internet-Standard Management Framework*. RFC 3410 (Informational). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc3410.txt (cited on pp. 48, 54).

Case, J., Fedor, M., Schoffstall, M., and Davin, J. (1988). *Simple Network Management Protocol*. RFC 1067. Obsoleted by RFC 1098. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc1067.txt (cited on p. 47).

Caudill, A. (2013). *Security Done Wrong: Leaky FTP Server*. URL: http://adamcaudill.com/2013/04/04/security-done-wrong-leaky-ftp-server/ (cited on pp. 3, 36, 42).

CCC (2013). *Chaos Computer Club breaks Apple TouchID*. Accessed: 06.05.2014. URL: http://www.ccc.de/en/updates/2013/ccc-breaks-apple-touchid (cited on pp. 19, 32).

Chapman, D. B. (1992). „Network (In) Security Through IP Packet Filtering." In: *Proc. Usenix* (cited on p. 45).

Charland, A. and Leroux, B. (2011). „Mobile application development: Web vs. native". *Communications of the ACM*, 54(5), pp. 49–53 (cited on p. 60).

Chatzis, N., Smaragdakis, G., Böttger, J., Krenc, T., and Feldmann, A. (2013). „On the benefits of using a large IXP as an Internet vantage point". In: *Proc. ACM Internet Measurement Conference*, pp. 333–346 (cited on pp. 92, 96, 113).

Chen, D. and Zhao, H. (2012). „Data security and privacy protection issues in cloud computing“. In: *Proc. IEEE Computer Science and Electronics Engineering (ICCSEE)*, pp. 647–651 (cited on p. 66).

Chen, G., Cao, Z., Xie, C., and Binet, D. (2014). *NAT64 Deployment Options and Experience*. RFC 7269 (Informational). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc7269.txt (cited on p. 132).

Chen, M., Hildebrand, D., Kuenning, G., Shankaranarayana, S., Singh, B., and Zadok, E. (2015). „Newer Is Sometimes Better: An Evaluation of NFSv4.“ *Proc. ACM SIGMETRICS* (cited on p. 54).

Chen, Y., Paxson, V., and Katz, R. H. (2010). „What's new about cloud computing security“. *University of California, Berkeley Report No. UCB/EECS-2010-5 January*, 20(2010), pp. 2010–5 (cited on pp. 62, 66).

Cheshire, S. and Krochmal, M. (2013). *DNS-Based Service Discovery*. RFC 6763 (Proposed Standard). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc6763.txt (cited on pp. 112, 114).

Claise, B. (2004). *Cisco Systems NetFlow Services Export Version 9*. RFC 3954 (Informational). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc3954.txt (cited on p. 46).

– (2008). *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information*. RFC 5101 (Proposed Standard). Obsoleted by RFC 7011. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc5101.txt (cited on p. 55).

Clark, D. (1988). „The design philosophy of the DARPA Internet protocols“. *ACM Computer Communication Review*, 18(4), pp. 106–114 (cited on p. 41).

Conta, A., Deering, S., and Gupta, M. (2006). *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*. RFC 4443 (Draft Standard). Updated by RFC 4884. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc4443.txt (cited on p. 72).

Cormack, G. V. (2007). „Email spam filtering: A systematic review“. *Foundations and Trends in Information Retrieval*, 1(4), pp. 335–455 (cited on pp. 44, 97, 110, 117, 127).

Corrente, A. and Tura, L. (2004). „Security performance analysis of SNMPv3 with respect to SNMPv2c“. In: *Proc. IFIP/IEEE Network Operations and Management Symposium (NOMS)*. Vol. 1, pp. 729–742 (cited on p. 54).

Costin, A., Zaddach, J., Francillon, A., Balzarotti, D., and Antipolis, S. (2014). „A large-scale analysis of the security of embedded firmwares“. In: *Proc. Usenix Security Symp.* (Cited on p. 57).

Cowan, C. et al. (1998). „StackGuard: Automatic Adaptive Detection and Prevention of Buffer-overflow Attacks“. In: *Proc. Usenix Security Symp.* Pp. 5–5 (cited on p. 143).

Crocker, D. (1982). *STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES*. RFC 822 (INTERNET STANDARD). Obsoleted by RFC 2822, updated by RFCs 1123, 2156, 1327, 1138, 1148. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc822.txt (cited on p. 44).

Cuppens, F., Cuppens-Boulahia, N., and Garcia-Alfaro, J. (2005). „Detection and removal of firewall misconfiguration“. In: *Proc. IASTED Conference on Communication, Network and Information Security*. Vol. 1, pp. 154–162 (cited on pp. 36, 50).

Czyz, J., Allman, M., Zhang, J., Iekel-Johnson, S., Osterweil, E., and Bailey, M. (2014). „Measuring IPv6 Adoption“. *Proc. ACM SIGCOMM*, 44(4), pp. 87–98 (cited on pp. 96, 99).

– (2015). „Measuring IPv6 adoption“. *ACM Computer Communication Review (CCR)*, 44(4), pp. 87–98 (cited on p. 111).

Czyz, J., Lady, K., Miller, S. G., Bailey, M., Kallitsis, M., and Karir, M. (2013). „Understanding IPv6 Internet background radiation“. In: *Proc. ACM Internet Measurement Conference*, pp. 105–118 (cited on p. 116).

Czyz, J., Luckie, M., Allman, M., and Bailey, M. (2016). „Don't Forget to Lock the Back Door! A Characterization of IPv6 Network Security Policy“. In: *Proc. Symposium on Network and Distributed System Security (NDSS)*. Vol. 389 (cited on pp. 93, 99, 108, 111, 116, 122, 130).

Damopoulos, D., Kambourakis, G., and Gritzalis, S. (2012). „From keyloggers to touchloggers: Take the rough with the smooth“. *Computers & Security* (cited on p. 20).

Darlene Storm, Computerworld.com (2014). *New attacks secretly use smartphone cameras, speakers and microphones*. URL: http://www.computerworld.com/article/2598704/mobile-security/new-attacks-secretly-use-smartphone-cameras--speakers-and-microphones.html (cited on p. 151).

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W. (2007). „Dynamo: Amazon's highly available key-value store“. In: *ACM SIGOPS Operating System Review*. Vol. 41. 6, pp. 205–220 (cited on pp. 58, 61, 75, 76, 78).

Deering, S. and Hinden, R. (1995). *Internet Protocol, Version 6 (IPv6) Specification*. RFC 1883 (Proposed Standard). Obsoleted by RFC 2460. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc1883.txt (cited on p. 10).

– (1998). *Internet Protocol, Version 6 (IPv6) Specification*. RFC 2460 (Draft Standard). Updated by RFCs 5095, 5722, 5871, 6437, 6564, 6935, 6946, 7045, 7112. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc2460.txt (cited on pp. 10, 72).

Deutsch, P., Emtage, A., and Marine, A. (1994). *How to Use Anonymous FTP*. RFC 1635 (Informational). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc1635.txt (cited on p. 42).

Dissent (2016). *Modern Business Solutions' leaky bucket provided a field day for downloaders*. Accessed: 18.02.2017. URL: https://www.databreaches.net/modern-business-solutions-leaky-bucket-provided-a-field-day-for-downloaders/ (cited on pp. 14, 15).

Docker.com (2015). URL: http://docker.io (cited on p. 60).

Drake, J. (2015). „Stagefright: scary code in the heart of Android". *Black Hat Briefings* (cited on pp. 17, 144).

Dreßler, P., Fischer, D., and Markscheffel, B. (2014). „Software solutions for high-security voice and data communication for smartphones". In: *Proc. International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, pp. 327–332 (cited on p. 144).

Droms, R. (1993). *Dynamic Host Configuration Protocol*. RFC 1531 (Proposed Standard). Obsoleted by RFC 1541. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc1531.txt (cited on p. 47).

Droms, R. and Arbaugh, W. (2001). *Authentication for DHCP Messages*. RFC 3118 (Proposed Standard). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc3118.txt (cited on p. 47).

Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and Carney, M. (2003). *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*. RFC 3315 (Proposed Standard). Updated by RFCs 4361, 5494, 6221, 6422, 6644, 7083, 7227, 7283, 7550. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc3315.txt (cited on p. 47).

Durumeric, Z., Wustrow, E., and Halderman, J. A. (2013). „ZMap: Fast Internet-wide Scanning and Its Security Applications." In: *Proc. Usenix Security Symp.* Pp. 605–620 (cited on pp. 4, 5, 41, 66, 73, 74, 79, 97, 137, 142).

Dwivedi, H. (2005). „iSCSI Security". *Black Hat* (cited on p. 50).

Eeten, M. van, Bauer, J. M., Asghari, H., Tabatabaie, S., and Rand, D. (2010). „The role of internet service providers in botnet mitigation an empirical analysis based on spam data". In: *Proc. Research Conference on Communications, Information and Internet Policy formerly Telecommunications Policy Research Conference (TPRC)* (cited on p. 85).

Eggendorfer, T. (2007). „Reducing spam to 20% of its original value with a SMTP tar pit simulator". In: *Proc. MIT Spam Conference* (cited on p. 73).

Eidnes, H., Groot, G. de, and Vixie, P. (1998). *Classless IN-ADDR.ARPA delegation*. RFC 2317 (Best Current Practice). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc2317.txt (cited on pp. 97, 119, 125).

Eisler, M. (1999). *NFS Version 2 and Version 3 Security Issues and the NFS Protocol's Use of RPCSEC_GSS and Kerberos V5*. RFC 2623 (Proposed Standard). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc2623.txt (cited on p. 49).

Equinix (2017). *Equinix Brings Direct, Private Access to the World's No. 1 CRM*. Accessed: May 15, 2017. URL: http://www.equinix.com/newsroom/press-releases/pr/123525/equinix-brings-direct-private-access-to-the-world-s-1-crm/ (cited on p. 133).

Ethernet Storage Forum (2015). *An Updated Overview of NFSv4*. Tech. rep. SNIA — Storage Networking Industry Association (cited on p. 53).

Falliere, N., Murchu, L. O., and Chien, E. (2011). „W32. stuxnet dossier". *White paper, Symantec Corp., Security Response*, 5 (cited on p. 50).

Fallon, R. (2015). „Celebgate: Two Methodological Approaches to the 2014 Celebrity Photo Hacks". In: *Internet Science*, pp. 49–60 (cited on p. 61).

Felt, A. P., Chin, E., Hanna, S., Song, D., and Wagner, D. (2011). „Android permissions demystified". In: *Proc. ACM Conference on Computer and Communications Security (CCS)*, pp. 627–638 (cited on p. 18).

Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., and Wagner, D. (2012). „Android permissions: User attention, comprehension, and behavior". In: *Proc. ACM Symposium on Usable Privacy and Security (SOUPS)*, p. 3 (cited on p. 18).

Ferguson, N. and Schneier, B. (2000). „A cryptographic evaluation of IPsec" (cited on pp. 52, 63).

Fiebig, T., Borgolte, K., Hao, S., Kruegel, C., and Vigna, G. (2017). „Something From Nothing (There): Collecting Global IPv6 Datasets From DNS". In: *Proc. Passive and Active Measurement (PAM)* (cited on pp. 97, 126, 133).

Fiebig, T., Danisevskis, J., and Piekarska, M. (2014). „A metric for the evaluation and comparison of keylogger performance". In: *Proc. USENIX Security Workshop on Cyber Security Experimentation and Test (CSET)* (cited on p. 101).

Fiebig, T., Katz, W., and Beek, J. van (2013). *Grindr application security evaluation report*. URL: https://www.os3.nl/_media/reports/grindr.pdf (cited on pp. 60–62).

Findlay, A. (2011). „Best Practices in LDAP Security" (cited on p. 52).

Fitzpatrick, B. (2004). „Distributed caching with memcached". *Linux journal*, (124), p. 5 (cited on p. 77).

Flanagan, H. and Ginoza, S. (2014). *RFC Style Guide*. RFC 7322 (Informational). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc7322.txt (cited on p. 38).

Ford-Hutchinson, P. (2005). *Securing FTP with TLS*. RFC 4217 (Proposed Standard). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc4217.txt (cited on p. 42).

Foremski, P., Plonka, D., and Berger, A. (2016). „Entropy/IP: Uncovering Structure in IPv6 Addresses“. In: *Proc. ACM Internet Measurement Conference* (cited on pp. 92, 96, 99, 100, 102, 106, 107, 111, 113, 118, 123, 149).

Foster, I., Prudhomme, A., Koscher, K., and Savage, S. (2015). „Fast and Vulnerable: A Story of Telematic Failures“. In: *Proc. USENIX Workshop on Offensive Technologies (WOOT)* (cited on p. 57).

Fratantonio, Y., Qian, C., Chung, S. P., and Lee, W. (2017). „Cloak and Dagger: From Two Permissions to Complete Control of the UI Feedback Loop“. In: *Proc. IEEE Security & Privacy*, pp. 1041–1057 (cited on p. 17).

Froomkin, A. M. (2000). „The death of privacy?“ *Stanford Law Review*, pp. 1461–1543 (cited on p. 19).

Frye, R., Levi, D., Routhier, S., and Wijnen, B. (2000). *Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework*. RFC 2576 (Proposed Standard). Obsoleted by RFC 3584. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc2576.txt (cited on p. 54).

– (2003). *Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework*. RFC 3584 (Best Current Practice). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc3584.txt (cited on p. 54).

Fujieda, I. and Haga, H. (1997). „Fingerprint input based on scattered-light detection“. *Applied optics*, 36(35), pp. 9152–9156 (cited on p. 22).

Fuller, V. and Li, T. (2006). *Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan*. RFC 4632 (Best Current Practice). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc4632.txt (cited on p. 72).

Fuller, V., Li, T., Yu, J., and Varadhan, K. (1992). *Supernetting: an Address Assignment and Aggregation Strategy*. RFC 1338 (Informational). Obsoleted by RFC 1519. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc1338.txt (cited on p. 97).

– (1993). *Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy*. RFC 1519 (Proposed Standard). Obsoleted by RFC 4632. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc1519.txt (cited on p. 72).

Furnell, S. M., Clarke, N., Werlinger, R., Hawkey, K., and Beznosov, K. (2009). „An integrated view of human, organizational, and technological challenges of IT security man-

agement“. *Information Management & Computer Security*, 17(1), pp. 4–19 (cited on p. 56).

Gao, H., Yegneswaran, V., Chen, Y., Porras, P., Ghosh, S., Jiang, J., and Duan, H. (2013). „An empirical reexamination of global DNS behavior“. *Proc. ACM SIGCOMM*, 43(4), pp. 267–278 (cited on pp. 6, 110, 111, 113, 115, 116, 127).

Garcia-Alfaro, J., Cuppens, F., Cuppens-Boulahia, N., Martinez, S., and Cabot, J. (2013). „Management of stateful firewall misconfiguration“. *Elsevier Computers & Security*, 39, pp. 64–85 (cited on p. 50).

Gasser, O., Scheitle, Q., Gebhard, S., and Carle, G. (2016). „Scanning the IPv6 Internet: Towards a Comprehensive Hitlist“ (cited on pp. 92, 93, 96).

Geller, B., Almog, J., Margot, P., and Springer, E. (1999). „A chronological review of fingerprint forgery“. *Journal of forensic sciences*, 44, pp. 963–968 (cited on p. 28).

Genkin, D., Shamir, A., and Tromer, E. (2014). „RSA key extraction via low-bandwidth acoustic cryptanalysis“. In: *Proc. Advances in Cryptology (CRYPTO)*, pp. 444–461 (cited on p. 48).

Ghena, B., Beyer, W., Hillaker, A., Pevarnek, J., and Halderman, J. A. (2014). „Green lights forever: Analyzing the security of traffic infrastructure“. In: *Proc. USENIX Workshop on Offensive Technologies (WOOT)* (cited on pp. 1, 35).

Gog, S. and Venturini, R. (2016). „Succinct Data Structures in Information Retrieval: Theory and Practice“. In: *Proc. ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 1231–1233 (cited on p. 125).

Gont, F. and Chown, T. (2016). *Network Reconnaissance in IPv6 Networks*. RFC 7707 (Informational). Internet Engineering Task Force. URL: `http://www.ietf.org/rfc/rfc7707.txt` (cited on pp. 95, 96, 98, 99).

Google (2017). *Google IPv6 Statistics*. URL: `https://www.google.com/intl/en/ipv6/statistics.html` (cited on pp. 113, 114).

Greenwald, G., MacAskill, E., and Poitras, L. (2013). „Edward Snowden: the whistleblower behind the NSA surveillance revelations“. *The Guardian*, 9(6) (cited on p. 2).

Greenwald, S. J., Olthoff, K. G., Raskin, V., and Ruch, W. (2004). „The user non-acceptance paradigm: INFOSEC's dirty little secret“. In: *Proc. ACM Workshop on New Security Paradigms*, pp. 35–43 (cited on p. 56).

Gutmann, P. and Grigg, I. (2005). „Security usability“. *Proc. IEEE Security & Privacy*, 3(4), pp. 56–58 (cited on pp. 52, 65).

Haber, E. M. and Bailey, J. (2007). „Design guidelines for system administration tools developed through ethnographic field studies“. In: *Proc. ACM Symposium on Computer Human Interaction for the Management of Information Technology*, p. 1 (cited on p. 56).

Halpern, J. (1991). *OSI CLNS and LLC1 protocols on Network Systems HYPERchannel*. RFC 1223 (Informational). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc1223.txt (cited on p. 45).

Halteren, A. van and Pawar, P. (2006). „Mobile service platform: A middleware for nomadic mobile service provisioning“. In: *Proc. IEEE Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 292–299 (cited on p. 60).

Hammel, M. J. (2011). „Managing KVM deployments with Virt-Manager“. *Linux Journal*, 2011(201) (cited on p. 59).

Handley, M., Rescorla, E., and IAB (2006). *Internet Denial-of-Service Considerations*. RFC 4732 (Informational). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc4732.txt (cited on p. 43).

Hao, S., Feamster, N., and Pandrangi, R. (2010). „An Internet-Wide View into DNS Lookup Patterns“. *Technical Report, School of Computer Science, Georgia Tech* (cited on p. 110).

Harris, B. and Hunt, R. (1999). „TCP/IP security threats and attack methods“. *Elsevier Computer Communications*, 22(10), pp. 885–897 (cited on p. 63).

Harrison, R. (2006). *Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms*. RFC 4513 (Proposed Standard). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc4513.txt (cited on pp. 52, 53).

Hassler, V. (1999). „X.500 and LDAP security: A comparative overview“. *IEEE Network Magazine*, 13(6), pp. 54–64 (cited on p. 52).

Hayes, J. (2013). „Security Issues and Best Practices for Water/Wastewater Facilities“. *Proceedings of the Water Environment Federation*, 2013(8), pp. 6442–6461 (cited on p. 51).

Haynes, T. and Noveck, D. (2015). *Network File System (NFS) Version 4 Protocol*. RFC 7530 (Proposed Standard). Updated by RFC 7931. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc7530.txt (cited on p. 53).

Helmer, G., Wong, J., Slagell, M., Honavar, V., Miller, L., and Lutz, R. (2002). „A software fault tree approach to requirements analysis of an intrusion detection system“. *Springer Requirements Engineering*, 7(4), pp. 207–220 (cited on p. 42).

Herriot, R., Butler, S., Moore, P., and Turner, R. (1999). *Internet Printing Protocol/1.0: Encoding and Transport*. RFC 2565 (Experimental). Obsoleted by RFC 2910. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc2565.txt (cited on p. 53).

Hilton, S. (2016). *Dyn Analysis Summary Of Friday October 21 Attack*. Accessed: 18.02.2017. URL: http://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/ (cited on p. 14).

Hinden, R. and Deering, S. (2006). *IP Version 6 Addressing Architecture*. RFC 4291 (Draft Standard). Updated by RFCs 5952, 6052, 7136, 7346, 7371. Internet Engineering Task

Force. URL: http://www.ietf.org/rfc/rfc4291.txt (cited on pp. 98, 100, 116).

Hoffman, P. (1999). *SMTP Service Extension for Secure SMTP over TLS*. RFC 2487 (Proposed Standard). Obsoleted by RFC 3207. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc2487.txt (cited on p. 44).

Hoffman, P. and Schlyter, J. (2012). *The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA*. RFC 6698 (Proposed Standard). Updated by RFCs 7218, 7671. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc6698.txt (cited on p. 64).

Holm, H., Sommestad, T., Almroth, J., and Persson, M. (2011). „A quantitative evaluation of vulnerability scanning“. *Information Management & Computer Security*, 19(4), pp. 231–247 (cited on p. 63).

Hsu, I. P.-S., Cheung, D. C. Y., and Jalan, R. R. (2008). *Global server load balancing*. US Patent 7,454,500 (cited on p. 133).

Hu, X., Li, B., Zhang, Y., Zhou, C., and Ma, H. (2016). „Detecting Compromised Email Accounts from the Perspective of Graph Topology“. In: *Proc. ACM Conference on Future Internet Technologies*, pp. 76–82 (cited on p. 118).

Huitema, C. (2006). *Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)*. RFC 4380 (Proposed Standard). Updated by RFCs 5991, 6081. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc4380.txt (cited on p. 115).

Hunt, A. and Thomas, D. (2000). *The pragmatic programmer: From journeyman to master*. Addison-Wesley Professional (cited on p. 60).

Hurricane Electric (2016). *The Hurricane Electric BGP Toolkit*. URL: http://bgp.he.net/ (cited on p. 79).

Huston, G. (2005). *Deprecation of "ip6.int"*. RFC 4159 (Best Current Practice). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc4159.txt (cited on p. 112).

IAB and IESG (2001). *IAB/IESG Recommendations on IPv6 Address Allocations to Sites*. RFC 3177 (Informational). Obsoleted by RFC 6177. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc3177.txt (cited on p. 126).

IANA (2002). *Special-Use IPv4 Addresses*. RFC 3330 (Informational). Obsoleted by RFC 5735. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc3330.txt (cited on p. 137).

Institute, S. (2003). *Printer Insecurity: Is it Really an Issue?* URL: https://www.sans.org/reading-room/whitepapers/threats/printer-insecurity-issue-1149 (cited on p. 53).

*Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation* (2002). URL: %7BITU-T%20Recommendation%20X.680%20(07/02),%20ISO/IEC%208824-1:2002%7D (cited on p. 52).

Jain, A. K., Chen, Y., and Demirkus, M. (2007). „Pores and ridges: high-resolution fingerprint matching using level 3 features“. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1), pp. 15–27 (cited on p. 30).

Jiang, X. and Yau, W.-Y. (2000). „Fingerprint minutiae matching based on the local and global structures“. In: *Proc. Intl. Conference on Pattern Recognition*. Vol. 2, pp. 1038–1041 (cited on p. 31).

John Matherly (2015a). *It's the Data, Stupid!* URL: https://blog.shodan.io/its-the-data-stupid/ (cited on pp. 5, 76, 142).

– (2015b). *Memory As A Service*. URL: https://blog.shodan.io/memory-as-a-service/ (cited on p. 76).

Jung, J. and Sit, E. (2004). „An empirical study of spam traffic and the use of DNS black lists“. In: *Proc. ACM Internet Measurement Conference*. ACM, pp. 370–375 (cited on pp. 44, 62, 63).

Jung, J., Sit, E., Balakrishnan, H., and Morris, R. (2002). „DNS Performance and the Effectiveness of Caching“. *IEEE/ACM Trans. Networking (TON)*, 10(5), pp. 589–603 (cited on p. 110).

Kalafut, A. J., Shue, C. A., and Gupta, M. (2008). „Understanding implications of DNS zone provisioning“. In: *Proc. ACM Internet Measurement Conference*, pp. 211–216 (cited on pp. 44, 110).

Kamp, K. A., Timmerman, N., Lind, G., Graybill, J., and Natowsky, I. (1999). „Discovering childhood: Using fingerprints to find children in the archaeological record“. *American Antiquity*, pp. 309–315 (cited on p. 21).

Kent, S. and Seo, K. (2005). *Security Architecture for the Internet Protocol*. RFC 4301 (Proposed Standard). Updated by RFCs 6040, 7619. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc4301.txt (cited on p. 51).

Killcrece, G., Kossakowski, K.-P., Ruefle, R. M., and Zajicek, M. T. (2003). „State of the practice of computer security incident response teams (CSIRTs)“ (cited on p. 74).

Kim, T. and Zeldovich, N. (2013). „Practical and Effective Sandboxing for Non-root Users.“ In: *Proc. Usenix*, pp. 139–144 (cited on p. 61).

Kim, Y., Daly, R., Kim, J., Fallin, C., Lee, J. H., Lee, D., Wilkerson, C., Lai, K., and Mutlu, O. (2014). „Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors“. In: *ACM SIGARCH Computer Architecture News*. Vol. 42. 3, pp. 361–372 (cited on p. 2).

Klensin, J., Freed, N., Rose, M., Stefferud, E., and Crocker, D. (1995). *SMTP Service Extensions*. RFC 1869 (INTERNET STANDARD). Obsoleted by RFC 2821. Internet Engi-

neering Task Force. URL: http://www.ietf.org/rfc/rfc1869.txt (cited on p. 44).

Klensin, J. and Padlipsky, M. (2008). *Unicode Format for Network Interchange*. RFC 5198 (Proposed Standard). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc5198.txt (cited on p. 57).

Koivunen, E. (2010). „"Why Wasn't I Notified?": Information Security Incident Reporting Demystified". In: *Nordic Conference on Secure IT Systems*. Springer, pp. 55–70 (cited on p. 74).

Kolosowa, W. and Hoppenstedt, M. (2017). *Skirt Club is an online community "for girls who play with girls". But a VICE investigation found the organisation didn't take its users' security that seriously.* Accessed: 18.02.2017. URL: https://www.vice.com/en_uk/article/a-sex-club-for-bisexual-women-left-intimate-photos-of-its-members-freely-accessible-online (cited on p. 14).

Krämer, J. I. (2015). „Why cryptography should not rely on physical attack complexity". PhD thesis. Springer (cited on p. 38).

Kraus, L., Fiebig, T., Miruchna, V., Möller, S., and Shabtai, A. (2015). „Analyzing End-Users' Knowledge and Feelings Surrounding Smartphone Security and Privacy". *Proc. IEEE Security & Privacy Workshops - Mobile Security Technologies (MoST)* (cited on pp. 2, 65, 143).

Krebs, B. (2015). *13 Million MacKeeper Users Exposed*. Accessed: 18.02.2017. URL: http://krebsonsecurity.com/2015/12/13-million-mackeeper-users-exposed/ (cited on p. 14).

Krenc, T. and Feldmann, A. (2016). „BGP Prefix Delegations: A Deep Dive". In: *Proc. ACM Internet Measurement Conference*, pp. 469–475 (cited on p. 126).

Krenc, T., Hohlfeld, O., and Feldmann, A. (2014). „An Internet census taken by an illegal botnet: A qualitative assessment of published measurements". *ACM Computer Communication Review*, 44(3), pp. 103–111 (cited on pp. 14, 58, 63, 74).

Kücken, M. (2007). „Models for fingerprint pattern formation". *Forensic science international*, 171(2), pp. 85–96 (cited on p. 21).

Kücken, M. and Newell, A. C. (2005). „Fingerprint formation". *Journal of theoretical biology*, 235(1), pp. 71–83 (cited on p. 21).

Kuhn, M. and Kuhn, C. (2003). „Compromising emanations: eavesdropping risks of computer displays". In: *Technical report*. University of Cambridge (cited on p. 20).

Kührer, M., Hupperich, T., Rossow, C., and Holz, T. (2014). „Exit from Hell? Reducing the Impact of Amplification DDoS Attacks". In: *Proc. Usenix Security Symp.* (Cited on p. 62).

Kushner, D. (2013). „The real story of Stuxnet“. *IEEE Journal Selected Areas in Communications*, 3(50), pp. 48–53 (cited on p. 2).

Lavie, A. and Denkowski, M. J. (2009). „The METEOR metric for automatic evaluation of machine translation“. *Machine Translation*, 23(2-3), pp. 105–115 (cited on p. 21).

Lawrence, N. and Traynor, P. (2012). „Under New Management: Practical Attacks on SNMPv3“. In: *Proc. USENIX Workshop on Offensive Technologies (WOOT)* (cited on p. 55).

Levine, J. (2010). *DNS Blacklists and Whitelists*. RFC 5782 (Informational). Internet Engineering Task Force. URL: `http://www.ietf.org/rfc/rfc5782.txt` (cited on p. 118).

Li, J., Zheng, R., and Chen, H. (2006). „From fingerprint to writeprint“. *Communications of the ACM*, 49(4), pp. 76–82 (cited on pp. 21, 22).

Lichtblau, F., Streibelt, F., Richter, P., and Feldmann, A. (2016). „Illegitimate Source IP Addresses At Internet Exchange Points“ (cited on p. 145).

Lindorfer, M., Neugschwandtner, M., Weichselbaum, L., Fratantonio, Y., Van Der Veen, V., and Platzer, C. (2014). „Andrubis–1,000,000 apps later: A view on current Android malware behaviors“. In: *Proc. Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*. IEEE, pp. 3–17 (cited on pp. 17, 144).

Liu, Y., Sarabi, A., Zhang, J., NAGHIZADEH ARDABILI, P., Karir, M., Bailey, M., and Liu, M. (2015). „Cloudy with a Chance of Breach: Forecasting Cyber Security Incidents“. In: *Proc. Usenix Security Symp.* (Cited on p. 36).

Livingood, J. (2012). *Considerations for Transitioning Content to IPv6*. RFC 6589 (Informational). Internet Engineering Task Force. URL: `http://www.ietf.org/rfc/rfc6589.txt` (cited on pp. 133, 138).

Lorente, E. N., Meijer, C., and Verdult, R. (2015). „Scrutinizing WPA2 Password Generating Algorithms in Wireless Routers“. In: *Proc. USENIX Workshop on Offensive Technologies (WOOT)* (cited on p. 58).

Luckie, M., Hyun, Y., and Huffaker, B. (2008). „Traceroute probe method and forward IP path inference“. In: *Proc. ACM Internet Measurement Conference*, pp. 311–324 (cited on pp. 92, 93).

Ludwig, A. (2017). *Delivering Secure, Client-Side Technology to Billions of Users*. Accessed: 17.02.2017. URL: `https://www.rsaconference.com/videos/delivering-secure-client-side-technology-to-billions-of-users` (cited on pp. 17, 144).

Lundgren, L. (2017). *Lightweight Protocol! Serious Equipment! Critical Implications!* Accessed: 17.02.2017. URL: `https://www.rsaconference.com/events/us17/agenda/sessions/6671-Lightweight-Protocol!-Serious-Equipment!-Critical-Implications!` (cited on p. 15).

Lyon, G. (2007). *Nmap reference guide (man page)* (cited on p. 73).

Macedo, T. and Oliveira, F. (2011). *Redis cookbook*. "O'Reilly Media, Inc." (cited on p. 77).

Macfarlane, R. and Buchanan, W. J. (2015). „Evaluation of TFTP DDoS Amplification Attack". *Elsevier Computers & Security* (cited on pp. 43, 63, 66).

Mahadevan, B. (2000). „Business models for Internet-based e-commerce". *California management review*, 42(4), pp. 55–69 (cited on p. 51).

Mahajan, R., Wetherall, D., and Anderson, T. (2002). „Understanding BGP misconfiguration". In: *ACM Computer Communication Review*. Vol. 32. 4, pp. 3–16 (cited on p. 12).

Mandiant Intelligence Center (2013). „Apt1: Exposing one of chinas cyber espionage units" (cited on p. 2).

Mansfield-Devine, S. (2016). „DDoS goes mainstream: how headline-grabbing attacks could make this threat an organisation's biggest nightmare". *Network Security*, 2016(11), pp. 7–13 (cited on pp. 146, 153).

Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., and Ghalsasi, A. (2011). „Cloud computing—The business perspective". *Decision Support Systems*, 51(1), pp. 176–189 (cited on p. 76).

Masse, M. (2011). *REST API design rulebook*. O'Reilly Media, Inc. (cited on p. 60).

Massie, M. L., Chun, B. N., and Culler, D. E. (2004). „The ganglia distributed monitoring system: design, implementation, and experience". *Parallel Computing*, 30(7), pp. 817–840 (cited on p. 136).

Mathews, L. (2016). *Hackers Use DDoS Attack To Cut Heat To Apartments*. URL: `http://www.forbes.com/sites/leemathews/2016/11/07/ddos-attack-leaves-finnish-apartments-without-heat/` (cited on pp. 1, 3, 14, 35).

Matsumoto, T., Matsumoto, H., Yamada, K., and Hoshino, S. (2002). „Impact of artificial gummy fingers on fingerprint systems". In: *Electronic Imaging 2002*. International Society for Optics and Photonics, pp. 275–289 (cited on p. 28).

Mayer, A., Wool, A., and Ziskind, E. (2000). „Fang: A firewall analysis engine". In: *Proc. IEEE Security & Privacy*, pp. 177–187 (cited on p. 50).

McGraw, G. (2004). „Software security". *Proc. IEEE Security & Privacy*, 2(2), pp. 80–83 (cited on p. 36).

– (2006). *Software security: building security in*. Vol. 1. Addison-Wesley Professional (cited on p. 36).

McGregor, S. E., Charters, P., Holliday, T., and Roesner, F. (2015). „Investigating the computer security practices and needs of journalists". In: *Proc. Usenix Security Symp.* Pp. 399–414 (cited on p. 65).

Meixell, B. and Forner, E. (2013). „Out of Control: Demonstrating SCADA Exploitation". *Black Hat* (cited on pp. 3, 36).

Janita (2016). *DDoS attack halts heating in Finland amidst winter*. URL: http://metropolitan.fi/entry/ddos-attack-halts-heating-in-finland-amidst-winter (cited on pp. 15, 17).

Mockapetris, P. (1983a). *Domain names: Concepts and facilities*. RFC 882. Obsoleted by RFCs 1034, 1035, updated by RFC 973. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc882.txt (cited on p. 43).

– (1983b). *Domain names: Implementation specification*. RFC 883. Obsoleted by RFCs 1034, 1035, updated by RFC 973. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc883.txt (cited on p. 43).

– (1987a). *Domain names - concepts and facilities*. RFC 1034 (INTERNET STANDARD). Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936, 8020. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc1034.txt (cited on pp. 96–98).

– (1987b). *Domain names - implementation and specification*. RFC 1035 (INTERNET STANDARD). Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2673, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966, 6604, 7766. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc1035.txt (cited on p. 103).

MongoDB website (2015). URL: https://www.mongodb.com/blog/post/mongodb-security-best-practices (cited on pp. 1, 3, 35, 59).

Moonen, R. and BV, C. D. I. (2012). „Digitale achterdeuren in de Nederlandse internet infrastructuur“. *Itsx bv* (cited on p. 48).

Munin (2002). *An open source networked resource monitoring tool*. URL: munin-monitoring.org (cited on p. 48).

Myers, J. (1999). *SMTP Service Extension for Authentication*. RFC 2554 (Proposed Standard). Obsoleted by RFC 4954. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc2554.txt (cited on p. 44).

Narten, T., Draves, R., and Krishnan, S. (2007). *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*. RFC 4941 (Draft Standard). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc4941.txt (cited on p. 118).

Nelson, B. J. (1981). „Remote procedure call“ (cited on p. 60).

Neuman, B. C. and Ts' O, T. (1994). „Kerberos: An authentication service for computer networks“. *IEEE Computer Magazine*, 32(9), pp. 33–38 (cited on p. 53).

Newman, S. (2015). *Building Microservices*. O'Reilly Media, Inc. (cited on pp. 60, 61).

Nicholas, D. and Huntington, P. (2003). „Micro-mining and segmented log file analysis: A method for enriching the data yield from internet log files“. *SAGE Journal of Information Science*, 29(5), pp. 391–404 (cited on pp. 97, 110).

Ning, P. (2014). „Samsung KNOX and Enterprise Mobile Security“. In: *Proc. ACM Workshop on Security and Privacy in Smartphones & Mobile Devices (SPSM)* (cited on pp. 2, 17, 22).

Cichonski, P., Millar, T., Grance, T., and Scarfone, K. (2012). „Computer Security Incident Handling Guide“. *NIST Special Publication*, 800, p. 61 (cited on p. 12).

Nussbaum, L., Neyron, P., and Richard, O. (2009). „On robust covert channels inside DNS“. In: *Proc. International Information Security Conference (IFIP)*, pp. 51–62 (cited on p. 101).

Okman, L., Gal-Oz, N., Gonen, Y., Gudes, E., and Abramov, J. (2011). „Security issues in NoSQL databases“. In: *Proc. IEEE Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 541–547 (cited on pp. 59, 63).

Oliveira, R. V., Pei, D., Willinger, W., Zhang, B., and Zhang, L. (2008). „In search of the elusive ground truth: Yhe Internet's AS-Level Connectivity Structure“. In: *Proc. ACM SIGMETRICS*. Vol. 36. 1, pp. 217–228 (cited on pp. 97, 110).

Orman, H. (2003). „The Morris worm: A fifteen-year perspective“. *Proc. IEEE Security & Privacy*, (5), pp. 35–43 (cited on p. 41).

O'Sullivan, T. (1971). *Telnet Protocol - a proposed document*. RFC 137. Updated by RFC 139. Internet Engineering Task Force. URL: `http://www.ietf.org/rfc/rfc137.txt` (cited on p. 57).

Ousterhout, J., Agrawal, P., Erickson, D., Kozyrakis, C., Leverich, J., Mazières, D., Mitra, S., Narayanan, A., Parulkar, G., Rosenblum, M., et al. (2010). „The case for RAMClouds: scalable high-performance storage entirely in DRAM“. *ACM SIGOPS Operating System Review*, 43(4), pp. 92–105 (cited on p. 77).

Pa, Y. M. P., Suzuki, S., Yoshioka, K., Matsumoto, T., Kasama, T., and Rossow, C. (2015). „IoTPOT: Analysing the Rise of IoT Compromises“. In: *Proc. USENIX Workshop on Offensive Technologies (WOOT)* (cited on pp. 3, 57, 58, 144, 146).

Padmanabhan, R., Owen, P., Schulman, A., and Spring, N. (2015). „Timeouts: Beware surprisingly high delay“. In: *Proc. ACM Internet Measurement Conference*, pp. 303–316 (cited on p. 73).

Pallis, G. (2010). „Cloud computing: the new frontier of Internet computing“. *IEEE Internet Computing*, (5), pp. 70–73 (cited on p. 56).

Payer, M. (2012). *Too much PIE is bad for performance*. Accessed: 27-01-2017. URL: `http://e-collection.library.ethz.ch/eserv/eth:5699/eth-5699-01.pdf` (cited on p. 143).

Payment Card Industry (2014). „Security Standards Council“. *Payment Card Industry Data Security Standard: Requirements and Security Assessment Procedures* (cited on p. 49).

Perreault, S., Yamagata, I., Miyakawa, S., Nakagawa, A., and Ashida, H. (2013). *Common Requirements for Carrier-Grade NATs (CGNs)*. RFC 6888 (Best Current Practice). Inter-

net Engineering Task Force. URL: http://www.ietf.org/rfc/rfc6888.txt (cited on p. 118).

Pessl, P., Gruss, D., Maurice, C., Schwarz, M., and Mangard, S. (2016). „DRAMA: Exploiting DRAM addressing for cross-cpu attacks“. In: *Proc. Usenix Security Symp.* (Cited on p. 2).

Petoff, J., Murphy, N. R., Jones, C., and Beyer, B. (2016). *Site Reliability Engineering: How Google Runs Production Systems*. " O'Reilly Media, Inc." (cited on p. 145).

Pfleeger, C. P. and Pfleeger, S. L. (2002). *Security in computing*. Prentice Hall Professional Technical Reference (cited on p. 38).

Phokeer, A., Aina, A., and Johnson, D. (2016). „DNS Lame delegations: A case-study of public reverse DNS records in the African Region“ (cited on pp. 110, 126).

Plonka, D. and Berger, A. (2015). „Temporal and Spatial Classification of Active IPv6 Addresses“. In: *Proc. ACM Internet Measurement Conference*. ACM, pp. 509–522 (cited on pp. 91–93, 96, 102, 111, 113, 118, 123, 149).

Polakis, I., Argyros, G., Petsios, T., Sivakorn, S., and Keromytis, A. D. (2015). „Where's Wally?: Precise User Discovery Attacks in Location Proximity Services“. In: *Proc. ACM Conference on Computer and Communications Security (CCS)*, pp. 817–828 (cited on pp. 60–62).

Postel, J. (1980). *User Datagram Protocol*. RFC 768 (INTERNET STANDARD). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc768.txt (cited on pp. 71, 73).

– (1981a). *Internet Control Message Protocol*. RFC 792 (INTERNET STANDARD). Updated by RFCs 950, 4884, 6633, 6918. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc792.txt (cited on p. 72).

– (1981b). *Internet Protocol*. RFC 791 (INTERNET STANDARD). Updated by RFCs 1349, 2474, 6864. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc791.txt (cited on p. 10).

– (1981c). *Transmission Control Protocol*. RFC 793 (INTERNET STANDARD). Updated by RFCs 1122, 3168, 6093, 6528. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc793.txt (cited on pp. 71, 72).

– (1982). *Simple Mail Transfer Protocol*. RFC 821 (INTERNET STANDARD). Obsoleted by RFC 2821. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc821.txt (cited on p. 44).

– (1992). *Introduction to the STD Notes*. RFC 1311 (Informational). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc1311.txt (cited on p. 38).

– (1993). *Instructions to RFC Authors*. RFC 1543 (Informational). Obsoleted by RFC 2223. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc1543.txt (cited on p. 38).

Postel, J. and Reynolds, J. (1997). *Instructions to RFC Authors*. RFC 2223 (Informational). Obsoleted by RFC 7322, updated by RFCs 5741, 6949. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc2223.txt (cited on p. 38).

Postel, J. and Reynolds, J. (1983). *Telnet Protocol Specification*. RFC 854 (INTERNET STANDARD). Updated by RFC 5198. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc854.txt (cited on p. 57).

Pratistha, I. M. P., Nicoloudis, N., and Cuce, S. (2003). „A Micro-Services Framework on Mobile Devices." In: *ICWS*, pp. 320–325 (cited on pp. 60, 61, 63).

Provos, N. and Honeyman, P. (2001). „ScanSSH: Scanning the Internet for SSH Servers." In: *Proc. Large Installation System Administration Conference (LISA)*, pp. 25–30 (cited on pp. 73, 74).

Pujol, E., Richter, P., and Feldmann, A. (2017). „Understanding the Share of IPv6 Traffic in a Dual-Stack ISP". In: *Proc. Passive and Active Measurement (PAM)*, pp. 3–16 (cited on p. 113).

Qazi, Z. A., Tu, C.-C., Chiang, L., Miao, R., Sekar, V., and Yu, M. (2013). „SIMPLE-fying middlebox policy enforcement using SDN". In: *ACM Computer Communication Review*. Vol. 43. 4, pp. 27–38 (cited on p. 61).

Quittek, J., Zseby, T., Claise, B., and Zander, S. (2004). *Requirements for IP Flow Information Export (IPFIX)*. RFC 3917 (Informational). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc3917.txt (cited on pp. 46, 55).

Ramachandran, A., Dagon, D., and Feamster, N. (2006). „Can DNS-based blacklists keep up with bots?" In: *Proc. Conference on Email and Anti-Spam (CEAS)* (cited on p. 118).

Ratha, N. K., Bolle, R., et al. (2004). *Automatic fingerprint recognition systems*. Vol. 1. Springer (cited on p. 22).

Rechthien, K. and Hargrave, W. (2011). *28C3 NOC Review*. URL: ftp://media.ccc.de/congress/2011/mp4-h264-HQ/28c3-4927-en-noc_review_28c3_camp_h264.mp4 (cited on p. 47).

Rekhter, Y. (1995). *CIDR and Classful Routing*. RFC 1817 (Historic). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc1817.txt (cited on p. 97).

Rekhter, Y. and Li, T. (1993). *An Architecture for IP Address Allocation with CIDR*. RFC 1518 (Historic). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc1518.txt (cited on p. 97).

Ren, K., Wang, C., and Wang, Q. (2012). „Security challenges for the public cloud". *IEEE Internet Computing*, (1), pp. 69–73 (cited on p. 62).

Rescorla, E. and Korver, B. (2003). *Guidelines for Writing RFC Text on Security Considerations*. RFC 3552 (Best Current Practice). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc3552.txt (cited on p. 38).

Richardson, T. and Levine, J. (2011). *The Remote Framebuffer Protocol*. RFC 6143 (Informational). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc6143.txt (cited on pp. 59, 60).

Richter, P., Allman, M., Bush, R., and Paxson, V. (2015). „A Primer on IPv4 Scarcity“. *ACM Computer Communication Review (CCR)*, 45(2), pp. 21–31 (cited on pp. 123, 125).

Richter, P., Chatzis, N., Smaragdakis, G., Feldmann, A., and Willinger, W. (2015). „Distilling the Internet's Application Mix from Packet-Sampled Traffic“. In: *Proc. Passive and Active Measurement (PAM)*, pp. 179–192 (cited on pp. 52, 61, 73).

Richter, P., Smaragdakis, G., Plonka, D., and Berger, A. (2016). „Beyond Counting: New Perspectives on the Active IPv4 Address Space“. In: *Proc. ACM Internet Measurement Conference* (cited on pp. 4, 72, 101, 123).

Rijswijk-Deij, R. van, Sperotto, A., and Pras, A. (2014). „DNSSEC and Its Potential for DDoS Attacks: A Comprehensive Measurement Study“. In: *Proc. ACM Internet Measurement Conference* (cited on p. 43).

Ripe NCC (2016a). *RIPE atlas*. URL: http://atlas.ripe.net (cited on p. 96).

– (2016b). *RIPE Routing Information Service (RIS)*. URL: https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris (cited on pp. 99, 121).

Roberts, L. G. (1967). „Multiple computer networks and intercomputer communication“. In: *Proc. ACM Symposium on Operating System Principles*, pp. 3–1 (cited on p. 9).

Roberts, L. G. and Wessler, B. D. (1970). „Computer network development to achieve resource sharing“. In: *Proc. of the ACM Spring Joint Computer Conference (AFIPS)*, pp. 543–549 (cited on pp. 9, 10).

Romanow, A., Mogul, J., Talpey, T., and Bailey, S. (2005). *Remote Direct Memory Access (RDMA) over IP Problem Statement*. RFC 4297 (Informational). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc4297.txt (cited on p. 52).

Rossow, C. (2014). „Amplification hell: Revisiting network protocols for DDoS abuse“. In: *Symposium on Network and Distributed System Security (NDSS)* (cited on pp. 43, 48, 63, 66).

Roth, V., Straub, T., and Richter, K. (2005). „Security and usability engineering with particular attention to electronic mail“. *International Journal of Human-Computer Studies*, 63(1), pp. 51–73 (cited on pp. 64, 65).

Rulifson, J. (1969). *Decode Encode Language (DEL)*. RFC 5. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc5.txt (cited on p. 10).

Sagiroglu, S. and Canbek, G. (2009). „Keyloggers“. *Technology and Society Magazine*, 28(3), pp. 10–17 (cited on p. 20).

Sanfilippo, S. (2015). *A few things about Redis security*. URL: http://antirez.com/news/96 (cited on pp. 5, 76, 78, 83, 142).

Santanna, J. J., Rijswijk-Deij, R. van, Hofstede, R., Sperotto, A., Wierbosch, M., Granville, L. Z., and Pras, A. (2015). „Booters—An analysis of DDoS-as-a-service attacks“. In: *Proc. International Symposium on Integrated Network Management (IM)*. IEEE, pp. 243–251 (cited on p. 16).

Satran, J., Meth, K., Sapuntzakis, C., Chadalapaka, M., and Zeidner, E. (2004). *Internet Small Computer Systems Interface (iSCSI)*. RFC 3720 (Proposed Standard). Obsoleted by RFC 7143, updated by RFCs 3980, 4850, 5048, 7146. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc3720.txt (cited on p. 49).

Schicker, P. and Duenki, A. (1976). „Virtual terminal definition and protocol“. *ACM Computer Communication Review*, 6(4), pp. 1–18 (cited on p. 9).

Schiller, J. (2002). *Strong Security Requirements for Internet Engineering Task Force Standard Protocols*. RFC 3365 (Best Current Practice). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc3365.txt (cited on pp. 51, 55, 67).

Schirrmacher, D. (2016). *https://www.heise.de/security/meldung/Hacker-kopieren-Nutzer-Daten-von-Mitfahrgelegenheit-de-und-Mitfahrzentrale-de-3507285.html*. Accessed: 18.02.2017. URL: https://www.heise.de/security/meldung/Hacker-kopieren-Nutzer-Daten-von-Mitfahrgelegenheit-de-und-Mitfahrzentrale-de-3507285.html (cited on pp. 14, 16).

Schneier, B. (2008). „The psychology of security“. In: *Progress in Cryptology–AFRICACRYPT 2008*. Springer, pp. 50–79 (cited on p. 56).

Schneier, B. and Kelsey, J. (1999). „Secure audit logs to support computer forensics“. *ACM Trans. on Information and System Security (TISSEC)*, 2(2), pp. 159–176 (cited on p. 78).

Schultz, E. E., Proctor, R. W., Lien, M.-C., and Salvendy, G. (2001). „Usability and security an appraisal of usability issues in information security methods“. *Elsevier Computers & Security*, 20(7), pp. 620–634 (cited on p. 65).

Schuster, F., Costa, M., Fournet, C., Gkantsidis, C., Peinado, M., Mainar-Ruiz, G., and Russinovich, M. (2015). „VC3: Trustworthy Data Analytics in the Cloud using SGX“ (cited on p. 2).

Sciberras, A. (2006). *Lightweight Directory Access Protocol (LDAP): Schema for User Applications*. RFC 4519 (Proposed Standard). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc4519.txt (cited on p. 52).

Segmuller, W. and Leiba, B. (2008). *Sieve Email Filtering: Relational Extension*. RFC 5231 (Proposed Standard). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc5231.txt (cited on p. 44).

ShadowServer Foundation (2014). *The scannings will continue until the internet improves*. URL: http://blog.shadowserver.org/2014/03/28/the-scannings-will-continue-until-the-internet-improves/ (cited on pp. 5, 74, 76, 79, 108, 142).

Al-Shaer, E. S. and Hamed, H. H. (2003). „Firewall policy advisor for anomaly discovery and rule editing“. In: *Proc. IFIP/IEEE Symposium Integrated Network Management*, pp. 17–30 (cited on p. 50).

Shapiro, E. (1969). *Network timetable*. RFC 4. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc4.txt (cited on p. 10).

Shodan (2015). *The search engine for the Internet of Things*. URL: https://www.shodan.io/ (cited on p. 41).

Simon, L. and Anderson, R. (2013). „PIN Skimmer: Inferring PINs Through The Camera and Microphone“. In: *Proc. ACM Workshop on Security and Privacy in Smartphones & Mobile Devices (SPSM)* (cited on pp. 19, 20).

Simonite, T. (2014). *Pay with Your Fingerprint*. Accessed: 30.04.2014. URL: http://m.technologyreview.com/news/525996/pay-with-your-fingerprint/ (cited on pp. 19, 32).

Slay, J. and Miller, M. (2008). *Lessons learned from the maroochy water breach*. Springer (cited on p. 50).

Sollins, K. (1992). *The TFTP Protocol (Revision 2)*. RFC 1350 (INTERNET STANDARD). Updated by RFCs 1782, 1783, 1784, 1785, 2347, 2348, 2349. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc1350.txt (cited on p. 42).

Sollins, K. (1981). *TFTP Protocol (revision 2)*. RFC 783. Obsoleted by RFC 1350. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc783.txt (cited on p. 42).

Solon, O. (2014). *NHS patient data made publicly available online*. URL: http://www.wired.co.uk/news/archive/2014-03/03/care-data-leaks (cited on pp. 14, 36).

Spafford, E. H. (1989). „The Internet worm program: An analysis“. *ACM Computer Communication Review*, 19(1), pp. 17–57 (cited on pp. 41, 45).

Specht, S. M. and Lee, R. B. (2004). „Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures.“ In: *ISCA PDCS*, pp. 543–550 (cited on p. 63).

Spring, T. (2016). *Open Databases a Juicy Extortion Target*. Accessed: 18.02.2017. URL: https://threatpost.com/open-databases-a-juicy-extortion-target/123688/ (cited on pp. 14, 16).

Srinivas, S. and Nair, A. (2015). „Security maturity in NoSQL databases-are they secure enough to haul the modern IT applications?“ In: *Proc. IEEE Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 739–744 (cited on pp. 59, 76).

Stallings, W. (1998). „SNMPv3: A security enhancement for SNMP“. *IEEE Communications Surveys*, 1(1), pp. 2–17 (cited on p. 54).

Stevenson, K. (2014). *Open Season on VNC Servers Around the World*. URL: https://medium.com/@kylestev/open-season-on-vnc-servers-around-the-world-4b89a0f8d992 (cited on p. 60).

Streibelt, F., Böttger, J., Chatzis, N., Smaragdakis, G., and Feldmann, A. (2013). „Exploring EDNS-client-subnet adopters in your free time“. In: *Proc. ACM Internet Measurement Conference*, pp. 305–312 (cited on pp. 44, 63).

Sultan, O. (2016). *https://www.hackread.com/tor-traffic-data-leak-caused-by-misconfigured-apache-servers/*. Accessed: 18.02.2017. URL: https://www.hackread.com/tor-traffic-data-leak-caused-by-misconfigured-apache-servers/ (cited on p. 14).

Tanenbaum, A. S. and Van Steen, M. (2007). *Distributed systems*. Prentice Hall (cited on pp. 10, 49).

Thomson, S., Huitema, C., Ksinant, V., and Souissi, M. (2003). *DNS Extensions to Support IP Version 6*. RFC 3596 (Draft Standard). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc3596.txt (cited on pp. 97, 98).

Troppens, U. (2014). *Secure Data Access with Kerberized NFS*. URL: https://www.ibm.com/developerworks/community/blogs/storageneers/entry/secure_data_access_with_kerberized_nfs?lang=en (cited on p. 49).

Tsikos, C. (1982). *Capacitive fingerprint sensor*. US Patent 4,353,056 (cited on p. 22).

Turnbull, J. (2013). *The Logstash Book*. James Turnbull (cited on pp. 77, 78).

Unger, N., Dechand, S., Bonneau, J., Fahl, S., Perl, H., Goldberg, I., and Smith, M. (2015). „SoK: Secure Messaging“. In: *Proc. IEEE Security & Privacy*, pp. 232–249 (cited on p. 65).

University of Oregon (2016). *Route Views Project*. URL: http://bgplay.routeviews.org (cited on pp. 99, 121).

Upmanyu, S. (2015). *Redis writing to .ssh/authorized_keys*. URL: http://stackoverflow.com/questions/33692230/redis-writing-to-ssh-authorized-keys (cited on p. 83).

Uppuluri, P. and Sekar, R. (2001). „Experiences with specification-based intrusion detection“. In: *Proc. Recent Advances in Intrusion Detection*, pp. 172–189 (cited on p. 42).

Vaas, L. (2016). *154 million voter records exposed, including gun ownership, Facebook profiles and more*. Accessed: 18.02.2017. URL: https://nakedsecurity.sophos.com/2016/06/23/154-million-voter-records-exposed-including-gun-ownership-facebook-profiles-and-more/ (cited on p. 14).

Veen, V. van der, Fratantonio, Y., Lindorfer, M., Gruss, D., Maurice, C., Vigna, G., Bos, H., Razavi, K., and Giuffrida, C. (2016). „Drammer: Deterministic rowhammer attacks on mobile platforms“. In: *Proc. ACM Conference on Computer and Communications Security (CCS)*, pp. 1675–1689 (cited on p. 17).

Velde, G. V. de, Popoviciu, C., Chown, T., Bonness, O., and Hahn, C. (2008). *IPv6 Unicast Address Assignment Considerations*. RFC 5375 (Informational). Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc5375.txt (cited on p. 126).

We-Vibe (2016). *make sex exciting again with the new We-Vibe Sync*. URL: http://we-vibe.com/ (cited on p. 1).

Vixie, P. A. (2016). „It's Time for An Internet-wide Recommitment to Measurement: And Here's How We Should Do It". In: *Proc. International on Workshop on Traffic Measurements for Cybersecurity*, pp. 1–2 (cited on pp. 96, 100).

Voiskounsky, A. E. and Smyslova, O. V. (2003). „Flow-based model of computer hackers' motivation". *CyberPsychology & behavior*, 6(2), pp. 171–180 (cited on p. 2).

Wahl, M., Howes, T., and Kille, S. (1997). *Lightweight Directory Access Protocol (v3)*. RFC 2251 (Proposed Standard). Obsoleted by RFCs 4510, 4511, 4513, 4512, updated by RFCs 3377, 3771. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc2251.txt (cited on p. 52).

Wall, D. (2007). *Cybercrime: The transformation of crime in the information age*. Vol. 4. Polity (cited on p. 2).

Weinmann, R.-P. (2016). *Were 900k Deutsche Telekom routers compromised by Mirai?* Accessed: 18.02.2017. URL: https://comsecuris.com/blog/posts/were_900k_deutsche_telekom_routers_compromised_by_mirai/ (cited on pp. 14, 17).

West, R. (2008). „The psychology of security". *Communications of the ACM*, 51(4), pp. 34–40 (cited on p. 56).

West-Brown, M. J., Stikvoort, D., Kossakowski, K.-P., Killcrece, G., and Ruefle, R. (2003). *Handbook for computer security incident response teams (CSIRTs)*. Tech. rep. Carnegie-Mellon University Pittsburgh, PA, Software Engineering Institute (cited on p. 74).

Wijnen, B., Harrington, D., and Presuhn, R. (1999). *An Architecture for Describing SNMP Management Frameworks*. RFC 2571 (Draft Standard). Obsoleted by RFC 3411. Internet Engineering Task Force. URL: http://www.ietf.org/rfc/rfc2571.txt (cited on p. 48).

Wool, A. (2004). „A quantitative study of firewall configuration errors". *IEEE Computer*, 37(6), pp. 62–67 (cited on p. 46).

Xu, T., Jin, L., Fan, X., Zhou, Y., Pasupathy, S., and Talwadker, R. (2015). „Hey, you have given me too many knobs!: Understanding and dealing with over-designed configuration in system software". In: *Proc. ACM Meeting on Foundations of Software Engineering*, pp. 307–319 (cited on p. 56).

Xu, T., Zhang, J., Huang, P., Zheng, J., Sheng, T., Yuan, D., Zhou, Y., and Pasupathy, S. (2013). „Do not blame users for misconfigurations". In: *Proc. ACM Conference on Symposium on Operating Systems Principles (SOSP)*, pp. 244–259 (cited on pp. 36, 52, 56).

Xu, Y., Heinly, J., White, A. M., Monrose, F., and Frahm, J.-M. (2013). „Seeing Double: Reconstructing Obscured Typed Input from Repeated Compromising Reflections“. In: *Proc. ACM Conference on Computer and Communications Security (CCS)*, pp. 1063–1074 (cited on pp. 19–21, 23–25, 27, 31, 147).

Xu, Z., Bai, K., and Zhu, S. (2012). „Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors“. In: *Proc. ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pp. 113–124 (cited on p. 20).

Yang, X., Wetherall, D., and Anderson, T. (2005). „A DoS-limiting network architecture“. In: *ACM Computer Communication Review*. Vol. 35. 4, pp. 241–252 (cited on p. 51).

Ylonen, T. and Lonvick, C. (2006). *The Secure Shell (SSH) Protocol Architecture*. RFC 4251 (Proposed Standard). Internet Engineering Task Force. URL: `http://www.ietf.org/rfc/rfc4251.txt` (cited on p. 57).

Ylönen, T. (1996). „SSH: Secure Login Connections over the Internet“. In: *Proc. Usenix Security Symp.* (Cited on pp. 57, 64, 65).

Yuan, L., Chen, H., Mai, J., Chuah, C.-N., Su, Z., and Mohapatra, P. (2006). „Fireman: A toolkit for firewall modeling and analysis“. In: *Proc. IEEE Security & Privacy*, 15–pp (cited on p. 50).

Zahid, A., Masood, R., and Shibli, M. A. (2014). „Security of sharded NoSQL databases: A comparative analysis“. In: *Proc. IEEE Conference on Information Assurance and Cyber Security (CIACS)*, pp. 1–8 (cited on p. 78).

Zeilenga, K. (2006). *Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map*. RFC 4510 (Proposed Standard). Internet Engineering Task Force. URL: `http://www.ietf.org/rfc/rfc4510.txt` (cited on p. 52).

Zhang, B., Liu, R., Massey, D., and Zhang, L. (2005). „Collecting the Internet AS-level topology“. *ACM Computer Communication Review*, 35(1), pp. 53–61 (cited on p. 99).

Zhang, J., Renganarayana, L., Zhang, X., Ge, N., Bala, V., Xu, T., and Zhou, Y. (2014). „EnCore: Exploiting system environment and correlation information for misconfiguration detection“. *ACM SIGPLAN Notices*, 49(4), pp. 687–700 (cited on pp. 10, 12).

Zhang, J., Durumeric, Z., Bailey, M., Liu, M., and Karir, M. (2014). „On the mismanagement and maliciousness of networks“. In: *Proc. Symposium on Network and Distributed System Security (NDSS)* (cited on p. 36).

Zhang, M., Ruan, Y., Pai, V. S., and Rexford, J. (2006). „How DNS Misnaming Distorts Internet Topology Mapping“. In: *Usenix Annual Technical Conference (ATC)*, pp. 369–374 (cited on pp. 97, 110).

Zhang, Y., Juels, A., Reiter, M. K., and Ristenpart, T. (2012). „Cross-VM side channels and their use to extract private keys“. In: *Proc. ACM Conference on Computer and Communications Security (CCS)*, pp. 305–316 (cited on p. 48).

Zhou, Y. and Jiang, X. (2012). „Dissecting android malware: Characterization and evolution“. In: *Proc. IEEE Security & Privacy*, pp. 95–109 (cited on p. 18).

Zhu, L., Jaganathan, K., and Hartman, S. (2005). *The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2*. RFC 4121 (Proposed Standard). Updated by RFCs 6112, 6542, 6649. Internet Engineering Task Force. URL: `http://www.ietf.org/rfc/rfc4121.txt` (cited on p. 53).