UNIVERSITY OF CALIFORNIA, SAN DIEGO

Spam Value Chain: Defensive Intervention Analysis

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Andreas Pitsillidis

Committee in charge:

Professor Stefan Savage, Chair
Professor Geoffrey M. Voelker, Co-Chair
Professor Bill Lin
Professor Ramesh Rao
Professor Lawrence K. Saul

2013

The dissertation of Andreas Pitsillidis is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____
Co-Chair

_____
Chair

University of California, San Diego

2013

DEDICATION

To Dimitris, Despo and Sophia.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisors, Professors Stefan Savage and Geoffrey M. Voelker. They are the primary reason that I decided to start this amazing journey in graduate school, and I am thankful to them for giving me the opportunity to do so. Their career achievements speak for themselves in regards to their quality as researchers. After working with them for the past six years though, I am even more amazed by their personalities, and I consider myself extremely lucky that I had the chance to have this experience.

I would also like to thank Kirill Levchenko for his invaluable help and guidance over all these years. His feedback helped shape up a lot of my work and in addition to my advisors, he is definitely one of the reasons I have made it this far. Additionally, there is a long list of collaborators I would like to thank for all their assistance: Chris Kanich, Damon McCoy, Neha Chachra, Tristan Halvorson, He 'Lonnie' Liu, Vern Paxson, Christian Kreibich, Chris Grier, Nick Weaver, Mark Felegyhzi, and Brandon Enright.

I also have a long list of current students and UCSD alumni I would like to thank; here they are in alphabetical order: Alvin AuYoung, Bhanu Vattikonda, Christos Kozanitis, Danny Huang, David Wang, Diwaker Gupta, Feng Lu, Justin Ma, Marti Motoyama, Michael Vrable, Patrick Verkaik, Peng Huang, Petros Mol, Qing Zhang, Ryan Braud, Yang Liu.

I would also like to thank my thesis committee members Bill Lin, Ramesh Rao, and Lawrence K. Saul for their valuable feedback and support during the course of my dissertation.

Finally, I would like to thank my family.

Chapters 1, 2, 3, 4, in part, are a reprint of the material as it appears in Proceedings of the ACM Internet Measurement Conference 2012. Pitsillidis, Andreas; Kanich, Chris; Voelker, Geoffrey M.; Levchenko, Kirill; Savage, Stefan. The dissertation author was the primary investigator and author of this paper.

Chapters 1, 2, 5, in part, are a reprint of the material as it appears in Proceedings of the Network and Distributed System Security Symposium 2010.

Pitsillidis, Andreas; Levchenko, Kirill; Kreibich, Christian; Kanich, Chris; Voelker, Geoffrey M.; Paxson, Vern; Weaver, Nicholas; Savage, Stefan. The dissertation author was the primary investigator and author of this paper.

Chapters 1, 2, 3, 6, in part, are a reprint of the material as it appears in Proceedings of the IEEE Symposium on Security and Privacy 2011. Levchenko, Kirill; Pitsillidis, Andreas; Chachra, Neha; Enright, Brandon; Felegyhzi, Mark; Grier, Chris; Halvorson, Tristan; Kanich, Chris; Kreibich, Christian; Liu, He; McCoy, Damon; Weaver, Nicholas; Paxson, Vern; Voelker, Geoffrey M.; Savage, Stefan. The dissertation author was one of the primary investigators and authors of this paper.

VITA

2007	Bachelor of Science in Computer Science University of Cyprus, Nicosia, Cyprus
2010	Master of Science in Computer Science University of California, San Diego, California, USA
2013	Doctor of Philosophy in Computer Science University of California, San Diego, California, USA

PUBLICATIONS

Andreas Pitsillidis, Chris Kanich, Geoffrey M. Voelker, Kirill Levchenko, and Stefan Savage. "Taster's Choice: A Comparative Analysis of Spam Feeds." In *Proceedings of the ACM Internet Measurement Conference, Boston, Massachusetts, USA, November 2012.*

Andreas Pitsillidis, Kirill Levchenko, Christian Kreibich, Chris Kanich, Geoffrey M. Voelker, Vern Paxson, Nicholas Weaver, and Stefan Savage. "Botnet Judo: Fighting Spam with Itself." In *Proceedings of the Network and Distributed System Security Symposium, San Diego, California, USA, February 2010.*

Kirill Levchenko, Andreas Pitsillidis, Neha Chachra, Brandon Enright, Mark Felegyhzi, Chris Grier, Tristan Halvorson, Chris Kanich, Christian Kreibich, He Liu, Damon McCoy, Nicholas Weaver, Vern Paxson, Geoffrey M. Voelker, and Stefan Savage. "Click Trajectories: End-to-End Analysis of the Spam Value Chain." In *Proceedings of the IEEE Symposium on Security and Privacy, Oakland, California, USA, May 2011.*

Damon McCoy, Andreas Pitsillidis, Grant Jordan, Nicholas Weaver, Christian Kreibich, Brian Krebs, Geoffrey M. Voelker, Stefan Savage, and Kirill Levchenko. "PharmaLeaks: Understanding the Business of Online Pharmaceutical Affiliate Programs." In *Proceedings of the USENIX Security Symposium, Bellevue, Washington, USA, August 2012.*

Andreas Pitsillidis, Yinglian Xie, Fang Yu, Martin Abadi, Geoffrey M. Voelker, and Stefan Savage. "How to Tell an Airport from a Home: Techniques and Applications." In *Proceedings of the 9th ACM Workshop on Hot Topics in Networks, Monterey, California, USA, October 2010.*

Chris Grier, Lucas Ballard, Juan Caballero, Neha Chachra, Christian J. Dietrich, Kirill Levchenko, Panayiotis Mavrommatis, Damon McCoy, Antonio Nappa, Andreas Pitsillidis, Niels Provos, M. Zubair Rafique, Moheeb Abu Rajab, Christian

Rossow, Kurt Thomas, Vern Paxson, Stefan Savage, and Geoffrey M. Voelker. "Manufacturing Compromise: The Emergence of Exploit-as-a-Service." In *Proceedings of the ACM Conference on Computer and Communications Security, Raleigh, North Carolina, USA, October 2012.*

ABSTRACT OF THE DISSERTATION

Spam Value Chain: Defensive Intervention Analysis

by

Andreas Pitsillidis

Doctor of Philosophy in Computer Science

University of California, San Diego, 2013

Professor Stefan Savage, Chair
Professor Geoffrey M. Voelker, Co-Chair

Much of computer security research today engages a hypothetical adversary: one whose aims and methods are either arbitrary or driven by some pre-supposed model of behavior. However, in many cases, the scope, motivation and technical evolution of actual attacks can be quite different and leads to a model where our research frequently trails the "truth on the ground". At the same time, our present ability to gather, process and analyze data concerning Internet activity is unmatched and thus there are tremendous opportunities in advancing a regime of "data-driven security", wherein our understanding of the adversary, of vulnerable users and of the efficacy of our current defenses and interventions can be placed on a strong empirical footing.

The spam problem is a primary example where the abundance of available data, enables a data-focused approach for studying it. Also known as unsolicited bulk e-mail, spam is perhaps the only Internet security phenomenon that leaves no one untouched, and has been continuously growing since the first reported complaint in 1978. Spam is essentially an advertising business which is very complex, with a lot of moving parts, and very specialized, with multiple parties involved. In this dissertation, I focus on the infrastructure and parties of the spam ecosystem responsible for monetization, which I define as the spam value chain. I focus on both advertising and click support, the two primary components of the spam value chain, analyze them, and identify the most effective places for intervention.

Our results demonstrate that good understanding of the problem at hand is essential for identifying how to efficiently address it. In this dissertation, I look into the spam ecosystem from the perspective of the attackers, in order to get a solid understanding of how they operate, and propose effective defenses at both the advertising and click support components of the spam value chain. I also present various limitations that come with spam feeds. Such datasets have an important role, as they are the basis of most spam-related studies. My work serves as a preliminary motivation to further refine our understanding of these limitations as a research community.

Chapter 1

Introduction

Since the first reported complaint in 1978, spam has been a major nuisance for the computer security community. The problem has continuously grown over the years, and its scale is nowadays enormous. Industry estimates suggest that spammers send well over 100 billion e-mails each day, and commercial anti-spam efforts comprise a market today with over $1B in annual revenue. Such anti-spam efforts have focused primarily on filtering, which aims at keeping spam messages away from users. At the same time, due to the plethora of spam data resources that are available, e-mail spam has also been the focus of a wide variety of measurement studies.

At its surface, the spam problem appears to be a simple concept: spammers bombard users with e-mails, and defenders are assigned with the task of blocking the delivery of such messages. In reality though, e-mail spam is an incredibly complex problem. On one hand, due to constant advancements in anti-spam efforts, great sophistication is required on behalf of spammers to successfully deliver their messages to users. On the other hand, spam has always primarily been an advertising medium. As such, it ultimately shares the underlying business model of advertising, and includes all relevant essential elements: advertising material distribution, site design, hosting, order handling, payment processing, etc. While a few years ago it was possible for a single person to handle all aspects of this business, today's complex requirements incur the need for more specialized handling. Thus today's spam ecosystem is complex, with a lot of moving and specialized

parts, and multiple parties involved. We define all the infrastructure and parties involved in the spam ecosystem, as the spam value chain.

The forefront of the spam value chain is advertising. In this case, advertising is the sending of spam e-mails to users, in an effort to attract potential customers. It is usually the case that a larger advertising campaign directly translates to higher revenue. This, in combination with the fact that e-mail costs so little to send, has driven spam sending to the enormous daily volume numbers we see today. Due to the fact that spam is so plentiful, it is tempting to dive right into the problem, and focus on the technical challenges of spam filtering. For problems that employ an enormous scale, such as this one, we often make the assumption as a community that whatever data is currently available to us, is sufficient for drawing conclusions about the global problem. We are unaware though of any systematic attempt to verify this assumption, prior to this research. Given the importance of input data in measurement studies, we begin our work with exploring whether commonly used spam data sources, can be representative and unbiased enough for making such extrapolations. We document significant variations across different feeds, and show how these variations can affect reported results.

With a good understanding of the problem and the accompanying data, we then shift our focus to the first level of the spam value chain: "advertising". Like in any advertising business, so long as the revenue driven by spam campaigns exceeds the cost, spam remains a profitable enterprise. Thus the goal of any defensive intervention at this level is e-mail filtering, and ultimately stopping users from visiting spam-advertised Web sites. The regime under which spam filtering works is reactive to your opponent's next move: receivers install filters to block spam, and spammers in turn modify their messages to evade them. In virtually any similar domain (e.g. anti-virus), maximizing the time advantage is crucial for success. With our work, we advocate shrinking the window between the time new spam gets sent, and the time it can be identified and blocked by the receiver. We do so by changing the vantage point from which we fight spam, and by focusing on a specific class of e-mails: botnet spam. In particular, it is well documented than the majority of spam e-mails nowadays is sent by just a handful of distributed

botnets. We take advantage of this fact by collecting the spam output of botnet hosts running in a controlled environment, and we use this data for building a system that allows the fast creation of efficient mail filters. We demonstrate this approach on mail traces from a range of modern botnets, and show that we can automatically filter such spam precisely and with virtually no false positives.

Finally, moving further down the spam value chain, we encounter the "click support" level. With this term, we refer to the infrastructure and mechanisms implicated when a user clicks on a spam-advertised link. Despite filtering defenses, it is unavoidable that there will be cases where we will fail to detect specific spam messages. Such messages will end up getting delivered to users, thus have a much higher chance of resulting in a visit. It is also possible for miscreants to attract clicks through means other than spam e-mail, which are outside the scope of this work. Once a click occurs, the job of the spammer is to have the user redirected to the Web site of interest. Although this may seem simple, in practice it is a challenging task, due to the wide array of defensive mechanisms deployed today. Thus multiple moving specialized parts need to be involved, and these are described in more detail in Section 2.4.1. With our work we quantify the full set of resources employed, with the focus on network resources. We also characterize the relative prospects for defensive interventions at each link in the spam value chain. Our intervention analysis is evaluated in terms of two factors: their overhead to implement and their business impact on the spam value chain. We show that the spam value chain, does not offer any major bottlenecks at the network level. Coincidentally, this is where most industry efforts are focused nowadays. With this in mind, we suggest exploring alternative non-technical means of intervention that can offer a greater potential for success.

1.1 Contributions

In this dissertation, we focus on the spam problem through a regime of "data-driven security". We first study the available data to us as researchers, and identify the accompanying characteristics and limitations. Absent this knowledge,

we show that drawing accurate conclusions about the global spam problem can be challenging. Although it is often the case that as researchers, we tend to focus primarily on the technical challenges of a problem, we demonstrate that there is great value in first gaining a deeper understanding of the problem, before attempting to tackle it. We then focus on botnet spam, which we identify as the most popular class of spam e-mail, and we develop a spam filtering system that aims to improve the response time of current approaches. We also evaluate different defensive intervention strategies for disrupting the spam value chain. We do so by first doing a holistic analysis of the surrounding ecosystem, and by then examining both the impact of these interventions, and their cost to implement. We posit that this analysis can greatly benefit future decision-making in regards to more efficiently nullifying the spam problem. The contributions of this dissertation are as follows:

- We evaluate how different spam data sources differ in content, by comparing the contents of ten distinct contemporaneous feeds of spam-advertised domain names. We document significant variations based on how such feeds are collected and show how these variations can produce differences in findings as a result. This is in contrast to today's normal approach followed by most studies, where despite the broad range of data available, a single "spam feed" is used as the sole data source. Based on our findings, we summarize some best practices for future spam measurement studies.

- We describe a system for better spam filtering, and we demonstrate our approach on mail traces from a range of modern botnets. We show that we can automatically filter such spam precisely and with virtually no false positives. Unlike more traditional approaches that view this problem from the receiver's point of view, we instead exploit the vantage point of the spammer. By instantiating and monitoring botnet hosts in a controlled environment, we are able to collect new spam as it is created, and consequently infer the underlying structure of these messages, which in turn enables efficiently blocking any subsequent similar messages.

- We present a holistic analysis that quantifies the full set of resources employed to monetize spam e-mail, with a focus on network resources. Using extensive measurements of three months of diverse spam data, and broad crawling of naming and hosting infrastructures, we analyze these network resources in terms of their volume. We relate these resources to the organizations who administer them and then use this data to characterize the relative prospects for defensive interventions at each link in the spam value chain. Ultimately, our works shows that targeting the spam problem at the network level is an enormous challenge. We demonstrate that our current approach to the problem as a community is infeasible, and we suggest exploring alternative non-technical means of intervention that can offer a greater potential for success.

Our analysis allows us to gain a deeper understanding of the spam problem, and enables defenders to identify how to better invest their often limited resources, for targeting it.

1.2 Organization

The remainder of this dissertation is organized in the following manner.

Chapter 2 provides background material on spam analysis and filtering, as well as on the spam ecosystem and related components.

Chapter 3 provides an overview of the datasets used, and describes in detail the methodology for crawling the naming and hosting infrastructure of three months of diverse spam data. We utilize the findings of this extensive measurement effort throughout the rest of the dissertation.

Chapter 4 explores the suitability of various spam data sources for different types of analyses. We analyze the contents of numerous such spam feeds, and document significant variations based on how they are collected. We show how these variations can produce differences in findings as a result, and suggest some general guidelines and good practices for related studies.

Chapter 5 describes a system for better filtering spam, by exploiting the vantage point of the spammer. By instantiating and monitoring botnet hosts in a controlled environment, we are able to monitor new spam as it is created, and consequently infer the underlying template used to generate polymorphic e-mail messages. We demonstrate this approach on mail traces from a range of modern botnets and show that we can automatically filter such spam precisely and with virtually no false positives.

Chapter 6 explores the full set of network resources employed to monetize spam e-mail. We relate these resources to the organizations who administer them and then use this data to characterize the relative prospects for defensive interventions at each link in the spam value chain.

Finally, Chapter 7 summarizes our work. Additionally, we discuss several future research directions in this space.

Chapter 1, in part, is a reprint of the material as it appears in Proceedings of the ACM Internet Measurement Conference 2012. Pitsillidis, Andreas; Kanich, Chris; Voelker, Geoffrey M.; Levchenko, Kirill; Savage, Stefan. The dissertation author was the primary investigator and author of this paper.

Chapter 1, in part, is a reprint of the material as it appears in Proceedings of the Network and Distributed System Security Symposium 2010. Pitsillidis, Andreas; Levchenko, Kirill; Kreibich, Christian; Kanich, Chris; Voelker, Geoffrey M.; Paxson, Vern; Weaver, Nicholas; Savage, Stefan. The dissertation author was the primary investigator and author of this paper.

Chapter 1, in part, is a reprint of the material as it appears in Proceedings of the IEEE Symposium on Security and Privacy 2011. Levchenko, Kirill; Pitsillidis, Andreas; Chachra, Neha; Enright, Brandon; Felegyhzi, Mark; Grier, Chris; Halvorson, Tristan; Kanich, Chris; Kreibich, Christian; Liu, He; McCoy, Damon; Weaver, Nicholas; Paxson, Vern; Voelker, Geoffrey M.; Savage, Stefan. The dissertation author was one of the primary investigators and authors of this paper.

Chapter 2

Background and Related Work

In Chapter 1 we defined spam as an advertising medium that involves multiple specialized parties. In this section, we provide background context on the problem and the surrounding ecosystem. At a high-level, we can view the problem of e-mail spam as the constant struggle between spammers on one hand, and anti-spam defenses on the other. The goal of spammers is to get e-mails delivered to users by evading existing defenses. Numerous techniques can be employed for doing so, that can differ in both their volume and sophistication, depending on the exact goals of the attacker. On the other hand, defenders need to be able to accommodate all different types of threats in their systems, with often contradicting goals and properties, and they also need to quickly adapt to any newly introduced threats. This need of being reactive to your opponent's next move is often found in computer security; a prominent example of this is the anti-virus industry. In such domains, the defenders must rely on up-to-date empirical data for developing and evaluating their systems.

Similarly, e-mail spam feeds play a crucial role when working on this problem, and a lot of effort goes into collecting such feeds. Due to the many different strategies employed by spammers though, and also because of various limitations that can arise during collection, working with spam feeds is in itself a complicated problem. In Section 2.1, we discuss this issue and the challenges involved by taking into consideration a wide array of feeds obtained from both academia and industry. We then focus specifically on one such feed type: botnet spam. This

spam vector is considered to be today's primary source of e-mail spam in terms of volume, and in Section 2.2 we describe botnets in more detail. After gaining a good understanding of inputs to spam defenses and studies, the next focus is spam filtering, and in particular methods with which we can stop e-mail messages from being delivered to users. This technique has traditionally been considered to be the primary approach for attacking the spam problem. In Section 2.3, we briefly survey currently employed approaches for filtering spam.

Successfully delivering e-mail to users is only a partial win for the spammers, though. As we have emphasized numerous times, spam has always primarily been an advertising medium. As such, the majority of spam e-mail messages sent nowadays contain links that redirect users to a Web site for purchasing advertised goods. Thus equally important is the handling of all required steps involved after a user clicks on a spam-advertised URL, since a successful sale is what ultimately generates revenue for the attackers. Multiple components are involved in this process though: site design, hosting, order handling and payment processing are all essential pieces of this chain. In Section 2.4 we present an overview of this so-called "spam value chain". We do so by providing a high-level overview of how modern spam operates nowadays, which parties it involves, and how these different parties are connected.

2.1 E-mail Spam Feeds

E-mail spam is perhaps the only Internet security phenomenon that leaves no one untouched. Everybody gets spam. Both this visibility and the plentiful nature of spam have naturally conspired to support a vast range of empirical measurement studies. Some of these have focused on how to best filter spam [7, 11, 16], others on the botnets used to deliver spam [31, 95], and others on the goals of spam, whether used as a vector for phishing [60], malware [33, 54] or, most commonly, advertising [45].

These few examples only scratch the surface, but importantly this work is collectively not only diverse in its analyses aims, but also in the range of data

sources used to drive those same conclusions. Among the spam sources included in such studies are the authors' own spam e-mail [7, 99], static spam corpora of varied provenance (e.g., Enron, TREC2005, CEAS2008) [24, 62, 77, 92, 99], open mail proxies or relays [23, 67, 68], botnet output [31], abandoned e-mail domains [6, 35], collections of abandoned e-mail accounts [90], spam automatically filtered at a university mail server [10, 70, 79, 91], spam-fed URL blacklists [56], spam identified by humans in a large Web-mail system [95, 98], spam e-mail filtered by a small mail service provider [73], spam e-mail filtered by a modest ISP [13] and distributed collections of honeypot e-mail accounts [85].

These data sources can vary considerably in volume — some may collect millions of spam messages per day, while others may gather several orders of magnitude fewer. Intuitively, it seems as though a larger data feed is likely to provide better coverage of the spam ecosystem (although, as we will show in Chapter 4, this intuition is misleading). However, an equally important concern is how differences in the *manner* by which spam is collected and reported may impact the *kind* of spam that is found.

To understand how this may be, it is worth first reflecting on the operational differences in spamming strategies. A spammer must both obtain an address list of targets and arrange for e-mail delivery. Each of these functions can be pursued in different ways, optimized for different strategies. For example, some spammers compile or obtain enormous "low-quality" address lists [38] (e.g., based on brute force address generation, harvesting of Web sites, etc.), many of which may not even be valid, while others purchase higher quality address lists that target a more precise market (e.g., customers who have purchased from an online pharmacy before). Similarly, some spam campaigns are "loud" and use large botnets to distribute billions of messages (with an understanding that the vast majority will be filtered [33]) while other campaigns are smaller and quieter, focusing on "deliverability" by evading spam filters.

These differences in spammer operations in turn can interact with differences in collection methodology. For example, spam collected via MX honeypots (accepting all SMTP connections to a quiescent domain) will likely contain broadly

targeted spam campaigns and few false positives, while e-mail manually tagged by human recipients (e.g., by clicking on a "this is spam" button in the mail client) may self-select for "high quality" spam that evades existing automated filters, but also may include legal, non-bulk commercial mail that is simply unwanted by the recipient.

In addition to properties of how spam data is *collected*, how the data is *reported* can also introduce additional limitations. For example, some data feeds may include the full contents of e-mail messages, but many providers are unwilling to do so due to privacy concerns. Instead, some may redact some of the address information, while, even more commonly, others will only provide information about the spam-advertised URLs contained with a message. Even within URL-only feeds there can be considerable differences. Some data providers may include full spam-advertised URLs, while others scrub the data to only provide the fully-qualified domain name (particularly for non-honeypot data, due to concern about side-effects from crawling such data). Sometimes data is reported in raw form, with a data record for each and every spam message, but in other cases providers aggregate and summarize. For example, some providers will de-duplicate identically advertised domains within a given time window, and domain-based blacklists may only provide a single record for each such advertised domain.

Taken together, all of these differences suggest that different kinds of data feeds may be more or less useful for answering particular kinds of questions. It is the purpose of our work in Chapter 4 to put this hypothesis on an empirical footing.

2.2 Spamming Botnets

One prominent data feed type is botnet spam. Since roughly 2004, bot-based spam distribution has emerged as the platform of choice for large-scale spam campaigns. Conceptually, spam botnets are quite simple—the spammer generates spam and then arranges to send it through thousands of compromised hosts, thereby laundering any singular origin that could be blacklisted. However,

an additional complexity is that spammers also need to generate content that is sufficiently polymorphic so that at least some of it will evade existing content filters. To describe this polymorphism, while ensuring that the underlying "messaging" is consistent, spammers have developed template-based systems. While original template-based spam generation from such templates was centralized, modern spammers now broadcast templates to individual bot hosts that in turn generate and send distinct message instances.

The template-based spamming engine of the Storm botnet has previously been described in [37], while Stern analyzed that of the Srizbi botnet [81] and first observed the opportunity for filtering techniques that "exploit the regularity of template-generated messages." In Chapter 5 we describe Judo, a spam filtering system that is a practical realization of this insight.

Closest to Judo is the Botlab system of John *et al.* [31]. Their system, contemporaneously built, also executes bots in a virtual machine environment and extracts the outbound spam e-mail messages. Indeed, Chapter 5 builds on their preliminary successes (and their data, which the authors have also graciously shared), which included using exact-matching of witnessed URL strings as a filter for future botnet spam. Our system is a generalization in that we do not assume the existence of any particular static feature (URL or otherwise), but focus on inferring the underlying template used by each botnet and from this generating comprehensive and precise regular expressions.

The system of Göbel *et al.* [20] uses a similar approach. Their system generates signatures by analyzing spam messages collected from compromised hosts as well. However, the Judo template inference algorithm, described in detail in Chapter 5, supports key additional elements, such as *dictionaries*, which make it significantly more expressive.

The AutoRE system of Xie *et al.* clusters spam messages into campaigns using heuristics about how embedded URLs are obfuscated [95]. This effort has algorithmic similarities to our system, as it too generates regular expressions over spam strings, but focuses on a single feature of spam e-mail (URLs). By contrast to these efforts, Judo is distinguished both by its generality (for example, generating

signatures for images spam or spam with "tinyurl" links for which URLs are not discriminatory) and by its design for on-line real-time use.

2.3 Current Anti-spam Approaches

Broadly speaking, anti-spam technologies deployed today fall into two categories: content-based and sender-based. Content-based approaches are the oldest and perhaps best known, focusing on filtering unwanted e-mail based on features of the message body and headers that are either anomalous (e.g., date is in the future) or consistent with undesirable messages (e.g., includes words like Rolex or Viagra). Early systems were simple heuristics configured manually, but these evolved into systems based on supervised learning approaches that use labeled examples of spam and non-spam to train a classifier using well-known techniques such as Bayesian inference [59, 75] and Support Vector Machines [16, 96]. These techniques can be highly effective (see Cormack and Lynam for an empirical comparison [14]) but are also subject to adversarial chaff and poisoning attacks [28, 49], and require great care to avoid false positives as spammers become more sophisticated at disguising their mail as legitimate.

Another class of content-based filtering approaches involves blacklisting the URLs advertised in spam [1, 3, 4, 95]. Because URL signatures are simple and require no complex training, they are more easily integrated into a closed-loop system: for example, in one study of spam sent by the Storm botnet, domains observed in templates were on average subsequently found on a URL blacklist only 18 minutes afterwards [38]. Unfortunately, URL-based systems also require comprehensive, up-to-date whitelists to avoid poisoning. They are also generally rigid in blocking *all* appearances of the URL regardless of context, and, of course, they do nothing for spam not containing a URL (e.g., stock schemes, image-based spam, and some types of phishing). The system we describe in Chapter 5 provides a fast, closed-loop response, while generating a more selective signature based on the URL (if present) and text of the spam instances.

Sender-based systems focus on the means by which spam is delivered. The assumption is that any Internet address that sends unsolicited messages is highly likely to repeat this act, and unlikely to send legitimate, desired communication. Thus, using a range of spam oracles, ranging from e-mail honeypots to user complaints, these systems track the IP addresses of Internet hosts being used to send spam. Individual mail servers can then validate incoming e-mail by querying the database (typically via DNS) to see if the transmitting host is a known spam source [32, 57]. Blacklists dealt very effectively with open e-mail relays and proxies, and forced spammers to move to botnet-based spam distribution, in which many thousands of compromised hosts under central control relay or generate spam on behalf of a single spammer [71]. As the number of hosts grows, this both reduces blacklist freshness and places scalability burdens on blacklist operators.

A related approach is sender reputation filtering, conceptually related to Internet address blacklisting. These schemes attempt to provide stronger ties between the nominal and true sender of e-mail messages in order to allow records of individual domains' communications to be employed to filter spam based on past behavior. Thus, using authentication systems such as SPF [94] or DomainKeys [44] (or heuristics such as greylisting [30]), a mapping can be made between the e-mail's originating domain (e.g., foo.com) and the mail servers authorized to send on behalf of these addresses. Having bound these together, mail receivers can then track the reputation for each sending domain (i.e., how much legitimate mail and how much spam each sends) and build filtering policies accordingly [84].

In practice, these techniques are used in combination, with their precise formulation and mixture tuned to new spam trends and "outbreaks" (e.g., image spam). We view Judo as a new component in this arsenal.

2.4 Spam Value Chain

As an advertising medium, spam ultimately shares the underlying business model of all advertising. So long as the revenue driven by spam campaigns exceeds their cost, spam remains a profitable enterprise. This glib description belies the

complexity of the modern spam business. While a decade ago spammers might have handled virtually all aspects of the business including e-mail distribution, site design, hosting, payment processing, fulfillment, and customer service [58], today's spam business involves a range of players and service providers. In this section, we review the broad elements in the spam value chain, the ways in which these components have adapted to adversarial pressure from the anti-spam community, and the prior research on applied e-crime economics that informs our work in Chapter 3 and Chapter 6.

2.4.1 How Modern Spam Works

While the user experience of spam revolves principally around the e-mail received, these constitute just one part of a larger value chain that we classify into three distinct stages: *advertising*, *click support*, and *realization*. Our discussion here reflects the modern understanding of the degree to which specialization and affiliate programs dominate the use of spam to sell products. To this end, we draw upon and expand the narrative of the "Behind Online Pharma" project [9], which documents the experience of a group of investigative journalists in exploring the market structure for online illegal pharmaceuticals, and Samosseiko's recent overview [76] of affiliate programs.

Advertising. Advertising constitutes all activities focused on reaching potential customers and enticing them into clicking on a particular URL. In this dissertation we focus on the e-mail spam vector, but the same business model occurs for a range of advertising vectors, including blog spam [63], Twitter spam [21], search engine optimization [88], and sponsored advertising [41, 42]. The delivery of e-mail spam has evolved considerably over the years, largely in response to increasingly complex defensive countermeasures. In particular, large-scale efforts to shut down open SMTP proxies and the introduction of well-distributed IP blacklisting of spam senders have pushed spammers to using more sophisticated delivery vehicles. These include botnets [22, 31, 95], Webmail spam [18], and IP prefix hijacking [72]. Moreover, the market for spam services has stratified over time;

for example, today it is common for botnet operators to rent their services to spammers on a contract basis [64].

The advertising side of the spam ecosystem has by far seen the most study, no doubt because it reflects the part of spam that users directly experience. Thus, a broad and ongoing literature examines filtering spam e-mail based on a variety of content features (as discussed in Chapter 5 and in [7, 29, 96]). Similarly, the network characteristics of spam senders have seen extensive study for characterizing botnet membership [97], identifying prefix hijacking [72], classifying domains and URLs [25, 51, 70, 93, 95], and evaluating blacklists [79, 80]. Finally, we note that most commercial anti-spam offerings focus exclusively on the delivery aspect of spam. In spite of this attention, spam continues to be delivered and thus our dissertation focuses strictly on the remaining two stages of the spam monetization pipeline.

Click support. Having delivered their advertisement, a spammer depends on some fraction of the recipients to respond, usually by clicking on an embedded URL and thus directing their browser to a Web site of interest. While this process seems simple, in practice a spammer must orchestrate a great many moving parts and maintain them against pressure from defenders.

Redirection sites. Some spammers directly advertise a URL such that, once the recipient's browser resolves the domain and fetches the content from it, these steps constitute the fullness of the promoted Web site. However, a variety of defensive measures—including URL and domain blacklisting, as well as site takedowns by ISPs and domain takedowns by registrars—have spurred more elaborate steps. Thus, many spammers advertise URLs that, when visited, redirect to additional URLs [6, 35]. Redirection strategies primarily fall into two categories: those for which a legitimate third party inadvertently controls the DNS name resource for the redirection site (e.g., free hosting, URL shorteners, or compromised Web sites), and those for which the spammers themselves, or perhaps parties working on their behalf, manage the DNS name resources (e.g., a "throwaway" domain such as `minesweet.ru` redirecting to a more persistent domain such as `greatjoywatches.com`).

Domains. At some point, a URL click will usually require domain name resources managed by the spammer or their accomplices. These names necessarily come via the services of a domain registrar, who arranges for the root-level registry of the associated top-level domain (TLD) to hold NS records for the associated registered domain. A spammer may purchase domains directly from a registrar, but will frequently purchase instead from a domain reseller, from a "domaineer" who purchases domains in bulk via multiple sources and sells to the underground trade, or directly from a spam "affiliate program" that makes domains available to their affiliates as part of their "startup package."

Interventions at this layer of the spam value chain depend significantly on the responsiveness of individual registrars and the pressure brought to bear [47]. For example, a recent industry study by LegitScript and KnujOn documents heavy concentration of spam-advertised pharmacies with domains registered through a particular set of registrars who appear indifferent to complaints [43].

Name servers. Any registered domain must in turn have supporting name server infrastructure. Thus spammers must provision this infrastructure either by hosting DNS name servers themselves, or by contracting with a third party. Since such resources are vulnerable to takedown requests, a thriving market has arisen in so-called "bulletproof" hosting services that resist such requests in exchange for a payment premium [36].

Web servers. The address records provided by the spammer's name servers must in turn specify servers that host (or more commonly proxy) Web site content. As with name servers, spam-advertised Web servers can make use of bulletproof hosting to resist takedown pressure [8, 83]. Some recent interventions have focused on effectively shutting down such sites by pressuring their upstream Internet service providers to deny them transit connectivity [13].

To further complicate such takedowns and to stymie blacklisting approaches, many spammers further obfuscate the hosting relationship (both for name servers and Web servers) using fast-flux DNS [26, 66, 69]. In this approach, domain records have short-lived associations with IP addresses, and the mapping infrastructure can spread the domain's presence over a large number of machines

(frequently many thousands of compromised hosts that in turn proxy requests back to the actual content server [12]). Furthermore, recently innovators have begun packaging this capability to offer it to third parties on a contract basis as a highly resilient content-hosting service [15].

Stores and Affiliate Programs. Today, spammers operate primarily as advertisers, rarely handling the back end of the value chain. Such spammers often work as affiliates of an online store, earning a commission (typically 30–50%) on the sales they bring in [76]. The affiliate program typically provides the storefront templates, shopping cart management, analytics support, and even advertising materials. In addition, the program provides a centralized Web service interface for affiliates to track visitor conversions and to register for payouts (via online financial instruments such as WebMoney). Finally, affiliate programs take responsibility for contracting for payment and fulfillment services with outside parties. Affiliate programs have proven difficult to combat directly—although, when armed with sufficient legal jurisdiction, law enforcement has successfully shut down some programs [17].

Realization. Finally, having brought the customer to an advertised site and convinced them to purchase some product, the seller realizes the latent value by acquiring the customer's payment through conventional payment networks, and in turn fulfilling their product request.

Payment services. To extract value from the broadest possible customer base, stores try to support standard credit card payments. A credit card transaction involves several parties in addition to the customer and merchant. The details around payment services are outside the scope of our current work.

Fulfillment. Finally, a store arranges to fulfill an order[1] in return for the customer's payment. For physical goods such as pharmaceuticals and replica products, this involves acquiring the items and shipping them to the customer. Global business-to-business Web sites such as Alibaba, ECPlaza, and ECTrade offer connections with a broad variety of vendors selling a range of such goods, including

[1]In principle, a store could fail to fulfill a customer's order upon receiving their payment, but this would both curtail any repeat orders and would lead to chargebacks through the payment card network, jeopardizing their relationship with payment service providers.

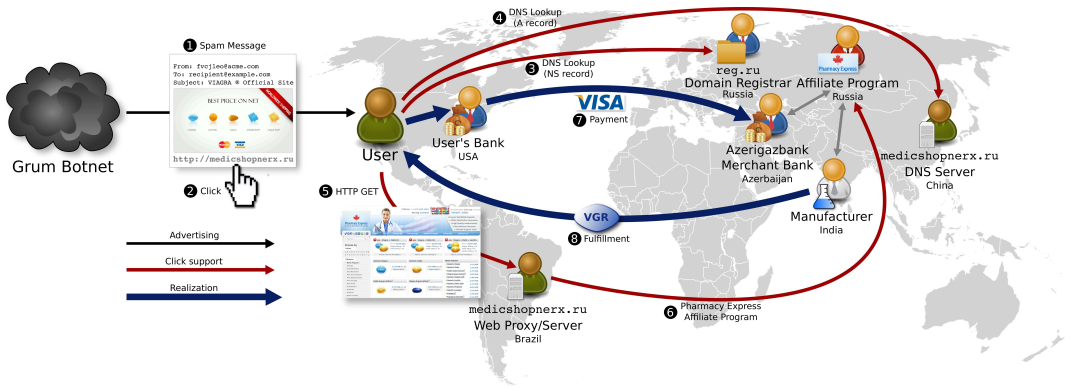


Figure 2.1: Infrastructure involved in a single URL's value chain, including advertisement, click support and realization steps.

prepackaged drugs—both brand (e.g., Viagra) and off-brand (e.g., sildenafil citrate capsules)—and replica luxury goods (e.g., Rolex watches or Gucci handbags). Generally, suppliers will offer direct shipping service ("drop shipping"), so affiliate programs can structure themselves around "just in time" fulfillment and avoid the overhead and risk of warehousing and shipping the product themselves.[2] Fulfillment for virtual goods such as software, music, and videos can proceed directly via Internet download.

2.4.2 Pharmacy Express: An Example

Figure 2.1 illustrates the spam value chain via a concrete example from the empirical data collected in Chapter 3. On October 27th, the Grum botnet delivered an e-mail titled *VIAGRA® Official Site* (❶). The body of the message includes an image of male enhancement pharmaceutical tablets and their associated prices (shown). The image provides a URL tag and thus when clicked (❷) directs the user's browser to resolve the associated domain name, `medicshopnerx.ru`. The machine providing name service resides in China, while hosting resolves to a machine in Brazil (❹). The user's browser initiates an HTTP request to the machine (❺), and receives content that renders the storefront for "Pharmacy Express," a

[2]Individual suppliers can differ in product availability, product quality, the ability to manage the customs process, and deliver goods on a timely basis. Consequently, affiliate programs may use different suppliers for different products and destinations.

brand associated with the Mailien pharmaceutical affiliate program based in Russia (❻).

After selecting an item to purchase and clicking on "Checkout", the storefront redirects the user to a payment portal served from `payquickonline.com` (this time serving content via an IP address in Turkey), which accepts the user's shipping, e-mail contact, and payment information, and provides an order confirmation number. Subsequent e-mail confirms the order, provides an *EMS* tracking number, and includes a contact e-mail for customer questions. The bank that issued the user's credit card transfers money to the acquiring bank (❼). Ten days later the product arrives, blister-packaged, in a cushioned white envelope with postal markings indicating a supplier named PPW based in Chennai, India as its originator (❽).

Chapter 2, in part, is a reprint of the material as it appears in Proceedings of the ACM Internet Measurement Conference 2012. Pitsillidis, Andreas; Kanich, Chris; Voelker, Geoffrey M.; Levchenko, Kirill; Savage, Stefan. The dissertation author was the primary investigator and author of this paper.

Chapter 2, in part, is a reprint of the material as it appears in Proceedings of the Network and Distributed System Security Symposium 2010. Pitsillidis, Andreas; Levchenko, Kirill; Kreibich, Christian; Kanich, Chris; Voelker, Geoffrey M.; Paxson, Vern; Weaver, Nicholas; Savage, Stefan. The dissertation author was the primary investigator and author of this paper.

Chapter 2, in part, is a reprint of the material as it appears in Proceedings of the IEEE Symposium on Security and Privacy 2011. Levchenko, Kirill; Pitsillidis, Andreas; Chachra, Neha; Enright, Brandon; Felegyhzi, Mark; Grier, Chris; Halvorson, Tristan; Kanich, Chris; Kreibich, Christian; Liu, He; McCoy, Damon; Weaver, Nicholas; Paxson, Vern; Voelker, Geoffrey M.; Savage, Stefan. The dissertation author was one of the primary investigators and authors of this paper.

Chapter 3

Meet the Data

Throughout this dissertation, we use a wide array of different spam data feeds. These feeds originate from either our own collection pipeline, or have been obtained from both academia and industry. In general, we choose to either (a) use raw spam data or (b) use crawler data, depending on which is the most suitable option for the given problem. The former case is self-explanatory: we utilize each feed as is, as input to our algorithms or evaluation methodologies. In such cases, we describe the exact data sources and methodology we use, in the respective chapters. Chapter 4 in part and Chapter 5 utilize such data. In the latter case, we rely on spam feeds that have been processed through our crawler pipeline, described in this chapter. Here, we are not interested in the actual content of e-mails or in spam-advertised URLs. We focus instead on the various elements of the spam value chain, which we extract by using spam data feeds as our input. Chapter 4 in part and Chapter 6 utilize data from our crawler pipeline.

In this chapter, we document in detail how our crawler pipeline works. The high-level goal of this pipeline is to identify and quantify the different elements of the spam value chain, which we have previously described in Section 2.4.1. Figure 3.1 concisely summarizes our data sources and methods. We start with a variety of full-message spam feeds, URL feeds, and our own botnet-harvested spam (❶). Feed parsers extract embedded URLs from the raw feed data for further processing (❷). A DNS crawler enumerates various resource record sets of the URL's domain, while a farm of Web crawlers visits the URLs and records HTTP-level

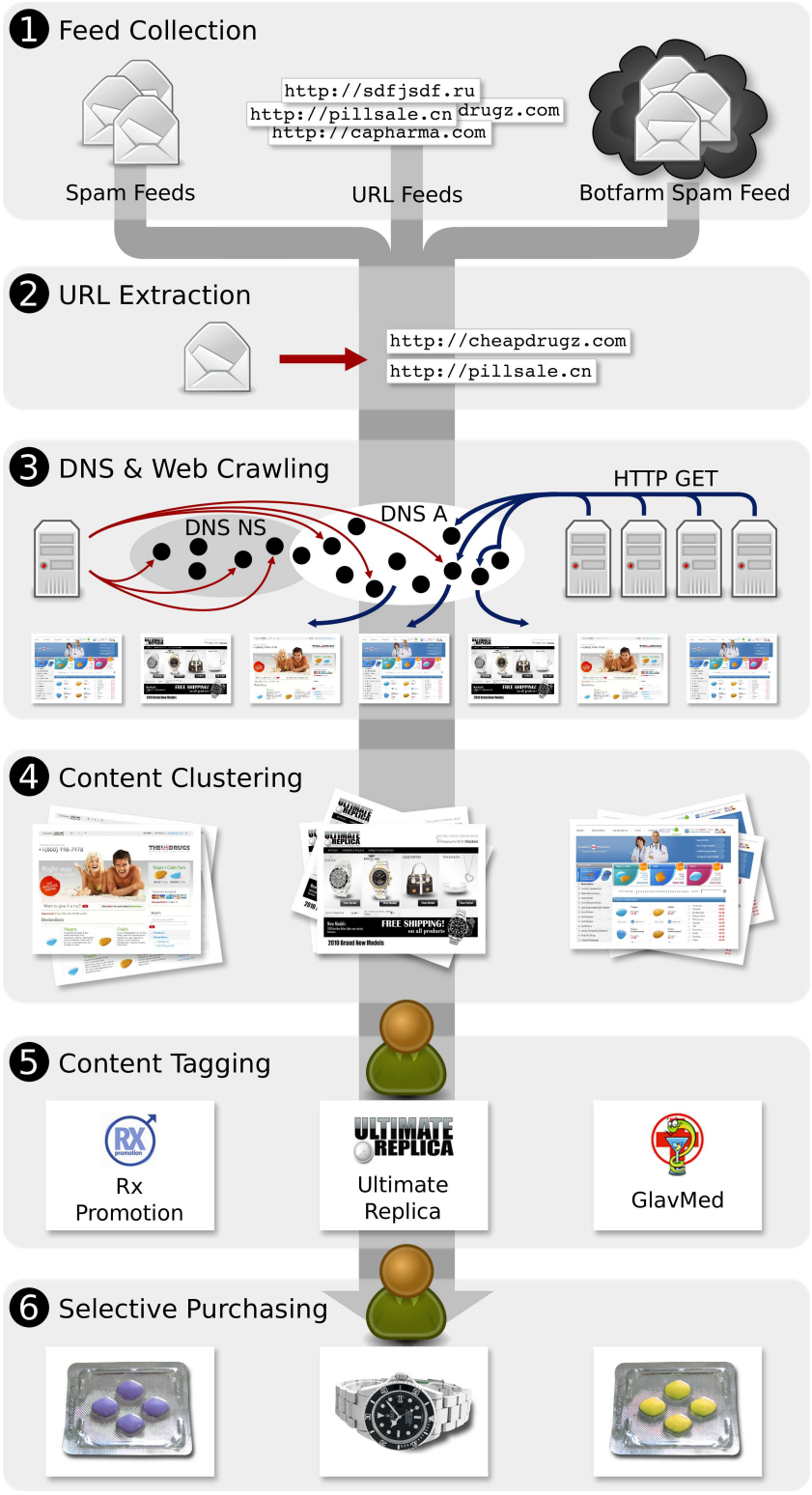


Figure 3.1: Our data collection and processing workflow.

Table 3.1: Feeds of spam-advertised URLs used in this dissertation. We collected feed data from August 1, 2010 through October 31, 2010.

Feed	Type	Received URLs	Unique Domains
Hu	Human identified	10,733,231	1,051,211
MX_1	MX honeypot	32,548,304	100,631
MX_2	MX honeypot	198,871,030	2,127,164
MX_3	MX honeypot	12,517,244	67,856
Ac_1	Seeded honey accounts	30,991,248	79,040
Ac_2	Seeded honey accounts	73,614,895	35,506
Hyb	Hybrid	451,603,575	1,315,292
Bot	Botnet	158,038,776	13,588,727
Cutwail	Botnet	3,267,575	65
Grum	Botnet	11,920,449	348
MegaD	Botnet	1,221,253	4
Rustock	Botnet	141,621,731	13,588,306
Other	Botnet	7,768	4
Total		968,918,303	17,813,952

interactions and landing pages (❸). A clustering tool clusters pages by content similarity (❹). A content tagger labels the content clusters according to the category of goods sold, and the associated affiliate programs (❺). We then make targeted purchases from each affiliate program (❻), and store the feed data and distilled and derived metadata in a database for subsequent analysis in Section 6.2. (Steps ❺ and ❻ are partially manual operations, the others are fully automated.) From this data, we in turn identify those sites advertising three popular classes of goods—pharmaceuticals, replica luxury goods and counterfeit software—as well as their membership in specific affiliate programs around which the overall business is structured.

The rest of this chapter describes these steps in detail.

3.1 Collecting Spam-Advertised URLs

Our dissertation is driven by a broad range of data sources of varying types, some of which are provided by third parties, while others we collect ourselves. Since

the goal of this dissertation is to decompose the spam ecosystem, it is natural that our seed data arises from spam e-mail itself. More specifically, we focus on the URLs embedded within such e-mail, since these are the vectors used to drive recipient traffic to particular Web sites. To support this goal, we obtained seven distinct URL feeds from third-party partners (including multiple commercial anti-spam providers), and harvested URLs from our own botfarm environment.

For this dissertation, we used the data from these feeds from August 1, 2010 through October 31, 2010, which together comprised nearly 1 billion URLs. Table 3.1 summarizes our feed sources along with the "type" of each feed, the number of URLs received in the feed during this time period, the number of distinct registered domains in those URLs, and a concise label (e.g., Ac_2) which indicates the feed type. One feed, Hyb, we identify as a "hybrid." We do not know the exact collection methodology it uses, but we believe it is a hybrid of multiple methods and we label it as such. Also note that the "bot" feeds tend to be focused spam sources, while the other feeds are spam sinks comprised of a blend of spam from a variety of sources. Further, individual feeds, particularly those gathered directly from botnets, can be heavily skewed in their makeup. For example, we received over 11M URLs from the Grum bot, but these only contained 348 distinct registered domains. Conversely, the 13M distinct domains produced by the Rustock bot are artifacts of a "blacklist-poisoning" campaign undertaken by the bot operators that comprised millions of "garbage" domains [89]. Thus, one must be mindful of these issues when analyzing such feed data in aggregate.

From these feeds we extract and normalize embedded URLs and insert them into a large multi-terabyte Postgres database. The resulting "feed tables" drive virtually all subsequent data gathering.

3.1.1 Types of Spam Domain Sources

The spam domain sources used in this dissertation fall into four categories: botnet-collected, MX honeypots, seeded honey accounts and human identified. In Chapter 4 we will also introduce a fifth category: blacklists. Each category comes

with its own unique characteristics, limitations and tradeoffs that we discuss briefly here.

Botnet Botnet datasets result from capturing instances of bot software and executing them in a monitored, controlled environment such that the e-mail they attempt to send is collected instead. Since the e-mail collected is only that sent by the botnet, such datasets are highly "pure": they have no false positives under normal circumstances.[1] Moreover, if we assume that all members of a botnet are used in a homogeneous fashion, then monitoring a single bot is sufficient for identifying the spamming behavior of the entire botnet. Botnet data is also highly accessible since a researcher can run an instance of the malware and obtain large amounts of botnet spam without requiring a relationship with any third-party security, mail or network provider [31]. Moreover, since many studies have documented that a small number of botnets are the primary source of spam e-mail messages, in principle such datasets should be ideally suited for spam studies [31, 52]. Finally, these datasets have the advantage of often being high volume, since botnets are usually very aggressive in their output rate.

MX honeypot MX honeypot spam is the result of configuring the MX record for a domain to point to an SMTP server that accepts all inbound messages. Depending on how these domains are obtained and advertised, they may select for different kinds of spam. For example, a newly registered domain will only capture spam using address lists created via brute force (i.e., sending mail to popular user names at every domain with a valid MX). By contrast, MX honeypots built using abandoned domains or domains that have become quiescent over time may attract a broader set of e-mail, but also may inadvertently collect legitimate correspondence arising from the domain's prior use. In general MX honeypots have low levels of false positives, but since their accounts are not in active use they will only tend to capture spam campaigns that are very broadly targeted and hence have high volume. Since high-volume campaigns are easier to detect, these same campaigns

[1] However, see Section 4.3.1 for an example of domain poisoning carried out by the Rustock botnet.

are more likely to be rejected by anti-spam filters. Thus, some of the most prevalent spam in MX-based feeds may not appear frequently in Web mail or enterprise e-mail inboxes.

Seeded honey accounts Like MX honeypots, seeded honey accounts capture unsolicited e-mail to accounts whose sole purpose is to receive spam (hence minimizing false positives). However, unlike MX honeypots, honey accounts are created across a range of e-mail providers, and are not limited to addresses affiliated with a small number of domains. However, since these e-mail addresses must also be seeded—distributed across a range of vectors that spammers may use to harvest e-mail address lists (e.g., such as forums, Web sites and mailing lists)—the "quality" of a honey account feed is related both to the number of accounts and how well the accounts are seeded. The greater operational cost of creating and seeding these accounts means that researchers generally obtain honey account spam feeds from third parties (frequently commercial anti-spam providers).

Honey accounts also have many of the same limitations as MX-based feeds. Since the accounts are not active, such feeds are unlikely to capture spam campaigns targeted using social network information (i.e., by friends lists of real e-mail users) or by mailing lists obtained from compromised machines [37]. Thus, such feeds mainly include low-quality campaigns that focus on volume and consequently are more likely to be captured by anti-spam filters.

Human identified These feeds are those in which humans actively nominate e-mail messages as being examples of spam, typically through a built-in mail client interface (i.e., a "this is spam" button). Moreover, since it is primarily large Web mail services that provide such user interfaces, these datasets also typically represent the application of human-based classification at very large scale (in our case hundreds of millions of e-mail accounts). For the same reason, human identified spam feeds are not broadly available and their use is frequently limited to large Web mail providers or their close external collaborators.

Human identified spam feeds are able to capture "high quality" spam since, by definition, messages that users were able to manually classify must also have

evaded any automated spam filters. As mentioned before, however, such feeds may underrepresent the high-volume campaigns since they will be pre-filtered before any human encounters them. Another limitation is that individuals do not have a uniform definition of what "spam" means and thus human identified spam can include legitimate commercial e-mail as well (i.e., relating to an existing commercial relationship with the recipient). Finally, temporal signals in human-identified spam feeds are distorted because identification occurs at human time scales.

Domain blacklists Domain blacklists are the last category of spam-derived data we consider and are the most opaque since the method by which they are gathered is generally not documented publicly.[2] In a sense, blacklists are meta-feeds that can be driven by different combinations of spam source data based on the organization that maintains them. Among the advantages of such feeds, they tend to be broadly available (generally for a nominal fee) and, because they are used for operational purposes, they are professionally maintained. Unlike the other feeds we have considered, blacklists represent domains in a binary fashion—either a domain is on the blacklist at time t or it is not. Consequently, while they are useful for identifying a range of spam-advertised domains, they are a poor source for investigating questions such as spam volume.

While these are not the only kinds of spam feeds in use by researchers (notably omitting automatically filtered spam taken from mail servers, which we did not pursue in our work due to privacy concerns), they capture some of the most popular spam sources as well as a range of collection mechanisms.

3.1.2 False Positives

No spam source is pure and all feeds contain false positives. In addition to feed-specific biases (discussed above), there is a range of other reasons why a domain name appearing in a spam feed may have little to do with spam.

[2]However, they are necessarily based on some kind of real-time spam data since their purpose is to identify spam-advertised domains that can then be used as a dynamic feature in e-mail filtering algorithms.

First, false positives occur when legitimate messages are inadvertently mixed into the data stream. This mixing can happen for a variety of reasons. For example, MX domains that are lexically similar to other domains may inadvertently receive mail due to sender typos (see Gee and Kim for one analysis of this behavior [19]). The same thing can occur with honeypot accounts (but this time due to username typos). We have also experienced MX honeypots receiving legitimate messages due to a user specifying the domain in a dummy e-mail address created to satisfy a sign-up requirement for an online service (we have found this to be particularly an issue with simple domain names such as "test.com").

The other major source of feed pollution is chaff domains: legitimate domains that are not themselves being advertised but co-occur in spam messages. In some cases these are purposely inserted to undermine spam filters (a practice well documented by Xie *et al.* [95]), in other cases they are simply used to support the message itself (e.g., image hosting) or are non-referenced organic parts of the message formatting (e.g., DTD reference domains such as w3.org or microsoft.com). Finally, to bypass domain-based blacklists some spam messages will advertise "landing" domains that in turn redirect to the Web site truly being promoted. These landing domains are typically either compromised legitimate Web sites, free hosting Web services (e.g., Google's Blogspot, Windows Live domains or Yahoo's groups) or Web services that provide some intrinsic redirection capability (e.g., bit.ly), as described in Chapter 6. We discuss in more detail how these issues impact our feeds in Section 4.3.1.

3.2 Crawler Data

The URL feed data subsequently drives active crawling measurements that collect information about both the DNS infrastructure used to name the site being advertised and the Web hosting infrastructure that serves site content to visitors. We use distinct crawlers for each set of measurements.

DNS Crawler

We developed a DNS crawler to identify the name server infrastructure used to support spam-advertised domains, and the address records they specify for hosting those names. Under normal use of DNS this process would be straightforward, but in practice it is significantly complicated by *fast flux* techniques employed to minimize central points of weakness. Similar to the work of [27], we query servers repeatedly to enumerate the set of domains collectively used for click support (Section 2.4.1).

From each URL, we extract both the fully qualified domain name and the registered domain suffix (for example, if we see a domain `foo.bar.co.uk` we will extract both `foo.bar.co.uk` as well as `bar.co.uk`). We ignore URLs with IPv4 addresses (just 0.36% of URLs) or invalidly formatted domain names, as well as duplicate domains already queried within the last day.

The crawler then performs recursive queries on these domains. It identifies the domains that resolve successfully and their authoritative domains, and filters out unregistered domains and domains with unreachable name servers. To prevent fruitless domain enumeration, it also detects wildcard domains (`abc.example.com`, `def.example.com`, etc.) where all child domains resolve to the same IP address. In each case, the crawler exhaustively enumerates all A, NS, SOA, CNAME, MX, and TXT records linked to a particular domain.

The crawler periodically queries new records until it converges on a set of distinct results. It heuristically determines convergence using standard maximum likelihood methods to estimate when the probability of observing a new unique record has become small. For added assurance, after convergence the crawler continues to query domains daily looking for new records (ultimately timing out after a week if it discovers none).

Web Crawler

The Web crawler replicates the experience of a user clicking on the URLs derived from the spam feeds. It captures any application-level redirects (HTML, JavaScript, Flash), the DNS names and HTTP headers of any intermediate servers

Table 3.2: Summary results of URL crawling. We crawl the registered domains used by over 98% of the URLs received.

Stage	*Count*	
Received URLs	968,918,303	
Distinct URLs	93,185,779	(9.6%)
Distinct domains	17,813,952	
Distinct domains crawled	3,495,627	
URLs covered	950,716,776	(98.1%)

and the final server, and the page that is ultimately displayed—represented both by its DOM tree and as a screenshot from a browser. Although straightforward in theory, crawling spam URLs presents a number of practical challenges in terms of scale, robustness, and adversarial conditions.

For this dissertation we crawled nearly 15 million URLs, of which we successfully visited and downloaded correct Web content for over 6 million (unreachable domains, blacklisting, etc., prevent successful crawling of many pages).[3] To manage this load, we replicate the crawler across a cluster of machines. Each crawler replica consists of a controller managing over 100 instances of Firefox 3.6.10 running in parallel. The controller connects to a custom Firefox extension to manage each browser instance, which incorporates the Screengrab! extension [61] to capture screen shots (used for manual investigations). The controller retrieves batches of URLs from the database, and assigns URLs to Firefox instances in a round-robin fashion across a diverse set of IP address ranges.[4]

Table 3.2 summarizes our crawling efforts. Since there is substantial redundancy in the feeds (e.g., fewer than 10% of the URLs are even unique), crawling every URL is unnecessary and resource inefficient. Instead, we focus on crawling URLs that cover the set of registered domains used by all URLs in the feed. Except in rare instances, all URLs to a registered domain are for the same affiliate

[3]By comparison, the spam hosting studies of Anderson *et al.* and Konte *et al.* analyzed 150,000 messages per day and 115,000 messages per month respectively [6, 35].

[4]Among the complexities, scammers are aware that security companies crawl them and blacklist IP addresses they suspect are crawlers. We mitigate this effect by tunneling requests through proxies running in multiple disparate IP address ranges.

program. Thus, the crawler prioritizes URLs with previously unseen registered domains, ignores any URLs crawled previously, and rate limits crawling URLs containing the same registered domain—both to deal with feed skew as well as to prevent the crawler from being blacklisted. For timeliness, the crawler visits URLs within 30 minutes of appearing in the feeds.

We achieve nearly complete coverage: Over 98% of the URLs received in the raw feeds use registered domains that we crawl. Note that we obtain this coverage even though we crawled URLs that account for only 20% of the nearly 18 million distinct registered domains in the feeds. This outcome reflects the inherent skew in the feed makeup. The vast majority of the remaining 80% of domains we did not crawl, and the corresponding 2% URLs that use those domains, are from the domain-poisoning spam sent by the Rustock bot and do not reflect real sites (Section 3.1).

3.2.1 Content Clustering and Tagging

The crawlers provide low-level information about URLs and domains. In the next stage of our methodology, we process the crawler output to associate this information with higher-level spam business activities.

Note that in this dissertation we exclusively focus on businesses selling three categories of spam-advertised products: pharmaceuticals, replicas, and software. We chose these categories because they are reportedly among the most popular goods advertised in spam [50]—an observation borne out in our data as well.[5]

To classify each Web site, we use *content clustering* to match sites with lexically similar content structure, *category tagging* to label clustered sites with the category of goods they sell, and *program tagging* to label clusters with their specific affiliate program and/or storefront brand. We use a combination of automated and manual analysis techniques to make clustering and tagging feasible for our large datasets, while still being able to manageably validate our results.

[5]We did not consider two other popular categories (pornography and gambling) for institutional and procedural reasons.

Table 3.3: Breakdown of clustering and tagging results.

Stage	*Pharmacy*	*Software*	*Replicas*	*Total*
URLs	346,993,046	3,071,828	15,330,404	365,395,278
Domains	54,220	7,252	7,530	69,002
Web clusters	968	51	20	1,039
Programs	30	5	10	45

Table 3.3 summarizes the results of this process. It lists the number of received URLs with registered domains used by the affiliate programs we study, the number of registered domains in those URLs, the number of clusters formed based on the contents of storefront Web pages, and the number of affiliate programs that we identify from the clusters. As expected, pharmaceutical affiliate programs dominate the data set, followed by replicas and then software. We identify a total of 45 affiliate programs for the three categories combined, that are advertised via 69,002 distinct registered domains (contained within 38% of all URLs received in our feeds). We next describe the clustering and tagging process in more detail.

Content clustering

The first step in our process uses a clustering tool to group together Web pages that have very similar content. The tool uses the HTML text of the crawled Web pages as the basis for clustering. For each crawled Web page, it uses a q-gram similarity approach to generate a fingerprint consisting of a set of multiple independent hash values over all 4-byte tokens of the HTML text. After the crawler visits a page, the clustering tool computes the fingerprint of the page and compares it with the fingerprints representing existing clusters. If the page fingerprint exceeds a similarity threshold with a cluster fingerprint (equivalent to a Jaccard index of 0.75), it places the page in the cluster with the greatest similarity. Otherwise, it instantiates a new cluster with the page as its representative.

Category tagging

The clusters group together URLs and domains that map to the same page content. The next step of category tagging broadly separates these clusters into those selling goods that we are interested in, and those clusters that do not (e.g., domain parking, gambling, etc). We are intentionally conservative in this step, potentially including clusters that turn out to be false positives to ensure that we include all clusters that fall into one of our categories (thereby avoiding false negatives).

We identify interesting clusters using generic keywords found in the page content, and we label those clusters with *category tags*—"pharma", "replica", "software"—that correspond to the goods they are selling. The keywords consist of large sets of major brand names (Viagra, Rolex, Microsoft, etc.) as well as domain-specific terms (herbal, pharmacy, watches, software, etc.) that appear in the storefront page. These terms are tied to the content being sold by the storefront site, and are also used for search engine optimization (SEO). Any page containing a threshold of these terms is tagged with the corresponding keyword. The remaining URLs do not advertise products that we study and they are left untagged.

Even with our conservative approach, a concern is that our keyword matching heuristics might have missed a site of interest. Thus, for the remaining untagged clusters, we manually checked for such false negatives, i.e., whether there were clusters of storefront pages selling one of the three goods that should have a category tag, but did not. We examined the pages in the largest 675 untagged clusters (in terms of number of pages) as well as 1,000 randomly selected untagged clusters, which together correspond to 39% of the URLs we crawled. We did not find any clusters with storefronts that we missed.[6]

[6]The lack of false negatives is not too surprising. Missing storefronts would have no textual terms in their page content that relate to what they are selling (incidentally also preventing the use of SEO); this situation could occur if the storefront page were composed entirely of images, but such sites are rare.

Program tagging

At this point, we focus entirely on clusters tagged with one of our three categories, and identify sets of distinct clusters that belong to the same affiliate program. In particular, we label clusters with specific *program tags* to associate them either with a certain affiliate program (e.g., EvaPharmacy—which in turn has many distinct storefront brands) or, when we cannot mechanically categorize the underlying program structure, with an individual storefront "brand" (e.g., Prestige Replicas). From insight gained by browsing underground forum discussions, examining the raw HTML for common implementation artifacts, and making product purchases, we found that some sets of the these brands are actually operated by the same affiliate program.

In total, we assigned program tags to 30 pharmaceutical, 5 software, and 10 replica programs that dominated the URLs in our feeds. Table 3.4 enumerates these affiliate programs and brands, showing the number of distinct registered domains used by those programs, and the number of URLs that use those domains. We also show two aggregate programs, Mailien and ZedCash, whose storefront brands we associated manually based on evidence gathered on underground Web forums (later validated via the purchasing process).[7] The "feed volume" shows the distribution of the affiliate programs as observed in each of the spam "sink" feeds (the feeds *not* from bots), roughly approximating the distribution that might be observed by users receiving spam.[8]

To assign these affiliate program tags to clusters, we manually crafted sets of regular expressions that match the page contents of program storefronts. For some programs, we defined expressions that capture the structural nature of the software engine used by all storefronts for a program (e.g., almost all EvaPharmacy sites contained unique hosting conventions). For other programs, we defined expressions that capture the operational modes used by programs that used mul-

[7]Note, ZedCash is unique among programs as it has storefront brands for each of the herbal, pharmaceutical and replica product categories.

[8]We remove botnet feeds from such volume calculations because their skewed domain mix would bias the results unfairly towards the programs they advertise.

tiple storefront templates (e.g., GlavMed).[9] For others, we created expressions for individual storefront brands (e.g., one for Diamond Replicas, another for Prestige Replicas, etc.), focusing on the top remaining clusters in terms of number of pages. Altogether, we assigned program tags to clusters comprising 86% of the pages that had category tags.

We manually validated the results of assigning these specific program tags as well. For every cluster with a program tag, we inspected the ten most and least common page DOMs contained in that cluster, and validated that our expressions had assigned them their correct program tags. Although not exhaustive, examining the most and least common pages validates the pages comprising both the "mass" and "tail" of the page distribution in the cluster.

Not all clusters with a category tag ("pharma") had a specific program tag ("EvaPharmacy"). Some clusters with category tags were false positives (they happened to have category keywords in the page, but were not storefronts selling category goods), or they were small clusters corresponding to storefronts with tiny spam footprints. We inspected the largest 675 of these clusters and verified that none of them contained pages that should have been tagged as a particular program in our analysis.

Chapter 3, in part, is a reprint of the material as it appears in Proceedings of the ACM Internet Measurement Conference 2012. Pitsillidis, Andreas; Kanich, Chris; Voelker, Geoffrey M.; Levchenko, Kirill; Savage, Stefan. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in part, is a reprint of the material as it appears in Proceedings of the IEEE Symposium on Security and Privacy 2011. Levchenko, Kirill; Pitsillidis, Andreas; Chachra, Neha; Enright, Brandon; Felegyhzi, Mark; Grier, Chris; Halvorson, Tristan; Kanich, Chris; Kreibich, Christian; Liu, He; McCoy, Damon; Weaver, Nicholas; Paxson, Vern; Voelker, Geoffrey M.; Savage, Stefan. The dissertation author was one of the primary investigators and authors of this paper.

[9] We obtained the full source code for all GlavMed and RX–Promotion sites, which aided creating and validating expressions to match their templates.

Table 3.4: Breakdown of the pharmaceutical, software, and replica affiliate programs advertising in our URL feeds.

Affiliate Program		*Distinct Domains*	*Received URLs*	*Feed Volume*
RxPrm	RX–Promotion	10,585	160,521,810	24.92%
Mailn	Mailien	14,444	69,961,207	23.49%
PhEx	Pharmacy Express	14,381	69,959,629	23.48%
EDEx	ED Express	63	1,578	0.01%
ZCashPh	ZedCash (Pharma)	6,976	42,282,943	14.54%
DrMax	Dr. Maxman	5,641	32,184,860	10.95%
Grow	Viagrow	382	5,210,668	1.68%
USHC	US HealthCare	167	3,196,538	1.31%
MaxGm	MaxGentleman	672	1,144,703	0.41%
VgREX	VigREX	39	426,873	0.14%
Stud	Stud Extreme	42	68,907	0.03%
ManXt	ManXtenz	33	50,394	0.02%
GlvMd	GlavMed	2,933	28,313,136	10.32%
OLPh	Online Pharmacy	2,894	17,226,271	5.16%
Eva	EvaPharmacy	11,281	12,795,646	8.7%
WldPh	World Pharmacy	691	10,412,850	3.55%
PHOL	PH Online	101	2,971,368	0.96%
Aptke	Swiss Apotheke	117	1,586,456	0.55%
HrbGr	HerbalGrowth	17	265,131	0.09%
RxPnr	RX Partners	449	229,257	0.21%
Stmul	Stimul-cash	50	157,537	0.07%
Maxx	MAXX Extend	23	104,201	0.04%
DrgRev	DrugRevenue	122	51,637	0.04%
UltPh	Ultimate Pharmacy	12	44,126	0.02%
Green	Greenline	1,766	25,021	0.36%
Vrlty	Virility	9	23,528	0.01%
RxRev	RX Rev Share	299	9,696	0.04%
Medi	MediTrust	24	6,156	0.01%
ClFr	Club-first	1,270	3,310	0.07%
CanPh	Canadian Pharmacy	133	1,392	0.03%
RxCsh	RXCash	22	287	<0.01%
Staln	Stallion	2	80	<0.01%
	Total	54,220	346,993,046	93.18%
Royal	Royal Software	572	2,291,571	0.79%
EuSft	EuroSoft	1,161	694,810	0.48%
ASR	Auth. Soft. Resellers	4,117	65,918	0.61%
OEM	OEM Soft Store	1,367	19,436	0.24%
SftSl	Soft Sales	35	93	<0.01%
	Total	7,252	3,071,828	2.12%
ZCashR	ZedCash (Replica)	6,984	13,243,513	4.56%
UltRp	Ultimate Replica	5,017	10,451,198	3.55%
Dstn	Distinction Replica	127	1,249,886	0.37%
Exqst	Exquisite Replicas	128	620,642	0.22%
DmdRp	Diamond Replicas	1,307	506,486	0.27%
Prge	Prestige Replicas	101	382,964	0.1%
OneRp	One Replica	77	20,313	0.02%
Luxry	Luxury Replica	25	8,279	0.01%
AffAc	Aff. Accessories	187	3,669	0.02%
SwsRp	Swiss Rep. & Co.	15	76	<0.01%
WchSh	WatchShop	546	2,086,891	0.17%
	Total	7,530	15,330,404	4.73%
	Grand Total	69,002	365,395,278	100%

Chapter 4

Taster's Choice: A Comparative Analysis of Spam Feeds

E-mail spam feeds are considered an invaluable tool for studying the spam problem. On one hand, they often serve as the basis of measurement studies, which in turn help to strengthen our understanding of the problem. On the other hand, they are used for both developing and evaluating algorithms that aim to stop spam. By better understanding the available data, and by extension the adversary, we greatly enhance our ability to address the challenges involved. In this chapter, we focus on the analysis of spam feeds.

4.1 Introduction

It is rare in the measurement of Internet-scale phenomena that one is able to make comprehensive observations. Indeed, much of our community is by nature opportunistic: we try to extract the most value from the data that is available. However, implicit in such research is the assumption that the available data is *sufficient* to reach conclusions about the phenomena at scale. Unfortunately, this is not always the case and some datasets are too small or too biased to be used for all purposes. A common security measurement domain where this issue appears is e-mail spam.

On the one hand e-mail spam is plentiful—everyone gets it—and thus is deceptively easy to gather. At the same time, the scale of the e-mail spam problem is enormous. Industry estimates (admittedly based on unknown methodology) suggest that spammers sent well over 100 billion e-mails each day in 2010 [39]. If true, then even a spam corpus consisting of 100,000 messages per day would constitute only *one ten thousandth of one percent* of what occurred globally. Thus, except for researchers at the very largest e-mail providers, we are all forced to make extrapolations by many orders of magnitude when generalizing from available spam data sources. Further, in making these extrapolations, we must assume both that our limited samples are sufficiently unbiased to capture the general behavior faithfully and that the behavior is large enough to be resolved via our measurements (concretely, that spam is dominated by small collections of large players and not vice versa). However, we are unaware of any systematic attempt to date to examine these assumptions and how they relate to commonly used data sources.

To explore these questions, we compare contemporaneous spam data from ten different data feeds, both academic and commercial, gathered using a broad range of different collection methodologies. To address differences in content, we focus on the Internet domain names advertised by spam messages in such feeds, using them as a *key* to identify like messages. Using this corpus, corresponding to over a billion messages distributed over three months, we characterize the relationships between its constituent data sources. In particular, we explore four questions about "feed quality": purity (how much of a given feed is actually spam?), coverage (what fraction of spam is captured in any particular feed?), timing (can a feed can be used to determine the start and end of a spam campaign?) and proportionality (can one use a single feed to accurately estimate the relative volume of different campaigns?).

Overall, we find that there are significant differences across distinct spam feeds and that these differences can frequently defy intuition. For example, our lowest-volume data source (comprising just over 10 million samples) captures more spam-advertised domain names than all other feeds combined (even though these other feeds contain two orders of magnitude more samples). Moreover, we find

that these differences in turn translate into analysis limitations; not all feeds are good for answering all questions. In the remainder of this chapter, we place this problem in context, describe our data sources and analysis, and summarize some best practices for future spam measurement studies.

4.2 Data and Methodology

In this chapter we compare ten distinct sources of spam data (which we call *feeds*), listed in Table 4.1. For this study, in addition to the feeds introduced earlier in Chapter 3, we also include two additional blacklist feeds (DBL and URIBL) which we receive by subscription. Furthermore we combine all the botnet data into a single feed.

These feeds range in their level of detail from full e-mail content to only domain names of URLs in messages. Comparisons between them are by necessity limited to the lowest common denominator, namely domain names. In the remainder of this chapter we treat each feed as a source of spam-advertised domains, regardless of any additional information available.

By comparing feeds at the granularity of domain names, we are implicitly restricting ourselves to spam containing URLs, that is, spam that is a Web-oriented advertisement in nature, at the exclusion of some less common classes of spam (e.g., malware distribution or advance fee fraud). Fortunately, such advertising spam is the dominant class of spam today.[1]

Up to this point, we have been using the term "domain" very informally. Before going further, however, let us say more precisely what we mean: a *registered domain*—in this chapter, simply a *domain*—is the part of a fully-qualified domain name that its owner registered with the registrar. For the most common top-level domains, such as COM, BIZ, and EDU, this is simply the second-level domain (e.g., "ucsd.edu"). All domain names at or below the level of registered domain (e.g., "cs.ucsd.edu") are administered by the registrant, while all domain names above (e.g., "edu") are administered by the registry. Blacklisting generally operates at

[1] One recent industry report [87] places Web-oriented advertising spam for pharmaceuticals at over 93% of all total spam volume.

Table 4.1: Summary of spam domain sources (feeds) used in this chapter. The first column gives the feed mnemonic used throughout.

Feed	Type	Received URLs	Unique Domains
Hu	Human identified	10,733,231	1,051,211
MX_1	MX honeypot	32,548,304	100,631
MX_2	MX honeypot	198,871,030	2,127,164
MX_3	MX honeypot	12,517,244	67,856
Ac_1	Seeded honey accounts	30,991,248	79,040
Ac_2	Seeded honey accounts	73,614,895	35,506
Hyb	Hybrid	451,603,575	1,315,292
Bot	Botnet	158,038,776	13,588,727
URIBL	Blacklist	n/a	144,758
DBL	Blacklist	n/a	413,392

the level of registered domains, because a spammer can create an arbitrary number of names under the registered domain name to frustrate fine-grained blacklisting below the level of registered domains.

With the exception of the two blacklists, we collected and processed the feeds used in this chapter in the context of our crawling effort, described in Chapter 3. Because we obtained the blacklist feeds after the completion of that work, we could not systematically crawl all of their domains. Thus the blacklist entries listed in the table only include the subset that *also* occurred in one of the eight base feeds. While this bias leads us to undercount the domains in each feed (thus underrepresenting their diversity), this effect is likely to be small. The DBL feed contained 13,175 additional domains that did not occur in any of our other base feeds (roughly 3% of its feed volume) while the URIBL feed contained only 3,752 such domains (2.5% of its feed volume).

4.3 Analysis

We set out to better understand the differences among sources of spam domains available to the researcher or practitioner. The value of a spam domain feed, whether used in a production system for filtering mail or in a measurement

study, ultimately lies in how well it captures the characteristics of spam. In this chapter we consider four qualities: *purity*, *coverage*, *proportionality*, and *timing*.

Purity is a measure of how much of a feed is actually spam domains, rather than benign or non-existent domains.

Coverage measures how much spam is captured by a feed. That is, if one were to use the feed as an oracle for classifying spam, coverage would measure how much spam is correctly classified by the oracle.

Proportionality is how accurately a feed captures not only the domains appearing in spam, but also their relative frequency. If one were tasked with identifying the top 10 most spammed domains, for example, proportionality would be the metric of interest.

Timing is a measure of how accurately the feed represents the period during which a domain appears in spam. Most often with timing we care about how quickly a domain appears in the feed after it appears in spam in the wild.

Unfortunately, all of the measures above presuppose the existence of an ultimate "ground truth," a platonic absolute against which all feeds can be compared. Sadly, we have no such feed: barring the challenges of even defining what such a feed would contain, the practical difficulty of capturing all spam (however defined) is immense. We can still gain useful insight, however, by comparing feeds to each other. In particular, for coverage and timing, we combine all of our feeds into one aggregate super-feed, taking it as our ideal. For proportionality, we measure the relative frequency of spam domains in incoming mail seen by a large Web mail provider, allowing us to compare the relative frequencies of domains in a spam feed to their frequencies in a representative e-mail feed.

In the remainder of this section, we evaluate the spam domain feeds available to us with respect to the qualities described above.

4.3.1 Purity

The purity of a feed is a measure of how much of the feed is actually spam, rather than benign or non-existent domains. Very simply, purity is the fraction of the feed that are spam domains. We refer to these spam domains appearing in

Table 4.2: Positive and negative indicators of feed purity. See Section 4.3.1 for discussion.

Feed	DNS	HTTP	Tagged	ODP	Alexa
Hu	88%	55%	6%	1%	1%
DBL	100%	72%	11%	<1%	<1%
URIBL	100%	85%	22%	2%	2%
MX_1	96%	83%	20%	9%	8%
MX_2	6%	5%	<1%	<1%	<1%
MX_3	97%	83%	16%	9%	7%
Ac_1	95%	82%	20%	8%	5%
Ac_2	96%	88%	33%	10%	11%
Bot	<1%	<1%	<1%	<1%	<1%
Hyb	64%	51%	2%	12%	10%

the feed as *true positives*, and non-spam domains appearing in the feed as *false positives*. Purity is thus akin to *precision* in Information Retrieval or *positive predictive value* in Statistics.

The importance of purity varies from application to application. If the feed is used to directly filter spam (by marking any message containing a domain appearing in the feed as spam), then purity is of paramount importance. On the other hand, for a measurement study, where spam domains are visited and further analyzed, low purity may tax the measurement system, but generally has little impact on the results once filtered.

Operationally, the nature of the false positives matters as well. While non-existent domains appearing in the feed are merely a nuisance, benign domains can lead to highly undesirable false positives in the filtering context.

Table 4.2 shows several purity indicators for each feed. The first three (*DNS*, *HTTP*, and *Tagged*) are positive indicators—larger numbers mean higher purity. The last two (Alexa and ODP) are negative indicators, with larger numbers meaning lower purity.

Non-existent Domains

The DNS column shows the proportion of domains in the feed that were registered, based on several major TLDs. Specifically, we checked the DNS zone files for the COM, NET, ORG, BIZ, US, AERO, and INFO top-level domains between April 2009 and March 2012, which bracket the measurement period by 16 months before and 16 months after. Together these TLDs covered between 63% and 100% of each feed. We report the number of domains in these TLDs that appeared in the zone file.

Blacklists, seeded honey accounts, and two of the three MX honeypot feeds consisted largely of real domains (over 95%). Human-identified spam and the hybrid feed were lower, at 88% and 64%, levels at which non-registered domains pose little harm operationally or experimentally.

Two feeds—Bot and MX_2—exhibit unusually low registration levels, however. Most of these relate to a single phenomenon, a period of several weeks during which the Rustock botnet was sending randomly-generated domains [47, 89]. Such bogus domains cost spammers nearly nothing to generate, while costing spam filter maintainers and spam researchers considerably more in dealing with them.

The HTTP column shows the fraction of domains in the feed that responded to an HTTP request (with a code 200 reply) made to any of the URLs received from the feed during the measurement period. Like the DNS registration measure, HTTP responses indicate that a feed contains live URLs (whether spam or not). Some amount of HTTP failures are inevitable, and we see success rates in the 51% to 88% range for most feeds, with the exception of the same two feeds—Bot and MX_2—discussed above.

Known Spam

An HTTP response still does not mean that a domain is not a benign domain accidentally included in the list. To get at the true spam domains, we turn to the Web content tagging carried out in Chapter 3. Recall from Section 3.1 that these are domains that lead to storefronts associated with known online pharmacies, replica stores, or software stores.

Such domains constituted 11–33% of domains in high-purity feeds. Note that while these domains are less than a third of all domains in a feed, they cover the bulk of the spam by volume.

Benign Domains

Finally, the ODP and Alexa columns indicate the number of domains in the feed that appeared in Open Directory Project [65] listings and the Alexa top 1 million Web sites [5] list. We expect that nearly all of these domains are benign, and their appearance in a feed is erroneous.[2]

There are at least three reasons why a benign domain might appear in spam. Benign domains may be included in a message by the spammer. A phishing e-mail, for example, may contain some legitimate links to the service being phished. In some cases, a legitimate e-mail may be inadvertently sent to a honeypot or honey account. For example, if an MX honeypot uses an abandoned, previously-used domain, it may still receive legitimate traffic from its former life. A third cause of benign domains appearing in spam are legitimate services being used by the spammer as a redirection mechanism. By using a URL shortening service, for example, the spammer can evade domain blacklists by hiding behind an established domain.

Using spam domain feeds to drive a production spam filtering system thus runs the risk of false positives. Because blacklists are intended specifically for this task, they have the fewest false positives: only 2% of domains in the URIBL feed, and less than 1% of DBL domains, intersected the ODP and Alexa lists.

Removing Impurities

In the analysis ahead of us, these impurities skew the results and thus obscure the picture. To better understand the useful contributions of each feed, we remove all non-responsive domains and all domains we believe are likely benign.

[2]While nothing prohibits a spam domain from appearing on the Alexa list or in the Open Directory listings, these domains are usually short-lived because their utility, and therefore use, is reduced with domain blacklisting. We expect both lists to be overwhelmingly composed of domains incompatible with spam advertising.

Specifically, for each feed, we take only the set of domains for which we receive at least one successful HTTP response and from this set remove all domains appearing on the Open Directory and Alexa list. (These are the domains listed in the *HTTP* column of Table 4.2 less those counted in the *ODP* and *Alexa* columns.) We call these ***live*** domains.

In several instances, data collected by the Web crawler (Chapter 3) allows us to see deeper into the nature of domain, namely into the affiliate programs and affiliates behind each domain. For this analysis, however, we are limited to the set of ***tagged*** domains. We remove Alexa and ODP domains from this set as well. Table 4.3 shows the number of distinct domains of each type in the feed. In the remainder of the chapter, we state explicitly whether a measurement uses live or tagged domains.

We have chosen to explicitly remove Alexa-listed and ODP-listed domains from the set of live and tagged domains used in the remainder of the chapter. As discussed in Section 4.3.1, live and even tagged domains may contain domains in the Alexa and ODP listings. An otherwise benign domain may be tagged if it is abused by a spammer as a redirection mechanism, as noted above. Unfortunately, the stakes are high when it comes to such false positives. These same Alexa and ODP domains—comprising less than 2% of the domains in a blacklist—are disproportionately more popular than spam domains. Figure 4.3 shows the fraction of spam messages containing such domains. In many feeds, these handful of benign domains comprise as much as 90% of live domain volume. Working at the granularity of registered domains, even a single URL redirecting to a spam site can affect the standing of an entire domain.

Practitioners must take great care in choosing which domains to blacklist and whether to blacklist each instance at the registered name or finer granularity. It is not the purpose of this dissertation, however, to design the perfect blacklist or blacklisting mechanism, and so we leave the question of how best to deal with potential false positives without a full and satisfactory resolution. For our analysis, we take the conservative approach and remove such suspect domains.

Table 4.3: Feed domain coverage showing total number of distinct domains (*Total* column) and number of domains exclusive to a feed (*Excl.* column).

	All Domains		Live Domains		Tagged Domains	
Feed	Total	Excl.	Total	Excl.	Total	Excl.
Hu	1,051,211	534,060	564,946	191,997	64,116	11,356
DBL	413,355	0	298,685	0	46,058	0
URIBL	144,721	0	119,417	0	30,891	0
MX_1	100,631	4,523	72,634	1,434	19,482	29
MX_2	2,127,164	1,975,081	93,638	6,511	18,055	4
MX_3	67,856	6,870	49,659	2,747	10,349	2
Ac_1	79,040	3,106	58,002	798	15,242	2
Ac_2	35,506	3,049	26,567	972	11,244	31
Bot	13,588,727	13,540,855	21,265	3,728	2,448	0
Hyb	1,315,292	1,069,074	496,893	322,215	25,993	1,285

4.3.2 Coverage

Roughly speaking, coverage is a measure of how many spam domains a feed contains. In an operational context, greater coverage—more spam domains—means more spam filtered. For a measurement study or system evaluation, more spam domains means more comprehensive results. In this section, we consider how coverage varies across our ten spam domain feeds. But domains do not exist in a vacuum: they are a projection of external entities into the domain name system, and it is often these entities that are the object of our interest. In the world of spam, these take the form of affiliate programs and affiliates. In Section 4.3.2 we compare feeds on the visibility they provide into that world.

Domains

Table 4.3 shows the number of live and tagged domains in each feed in the *Total* column. Recall that *live* domains are those that resulted in at least one successful Web visit to a URL containing the domain, while *tagged* domains are those for which the final Web site is a known storefront (Section 3.1).

In absolute terms, whether one considers live domains or tagged domains, the largest contributor of unique instances is the human-identified spam domain

feed Hu, despite also being the smallest feed in terms of absolute volume (see Table 4.1). The reason for this coverage is undoubtedly that this particular provider has hundreds of millions of accounts and thus their customers are likely to be targets of virtually any spam campaign. In turn, we believe that the reason this feed has low volume is that as users identify e-mails as "spammy" the included domains are used to filter subsequent inbound messages. Thus, high-volume campaigns are unlikely to have high representation in such a feed.

Clearly, if one had to choose only one feed to provide maximum coverage, it would be that feed. Unfortunately, outside large mail providers, such data is not widely available to the research community. Instead, the readily-available blacklists—DBL and URIBL—are an excellent alternative, providing more tagged domains than any other feed besides Hu.

Exclusive domains So far, we have been comparing feeds in terms of their absolute coverage: the total number of spam domains contributed. Given a choice of one feed, one may well pick the largest one by this measure. A feed's value, however, may be in its differential contribution, that is, in the domains it provides that are in no other feed. We term domains that occur in exactly one feed *exclusive* domains. Across our feeds, 60% of all live domains and 19% of all tagged domains were exclusive to a single feed.

Table 4.3 shows number of exclusive domains provided by each feed in the *Excl.* column. The relationship between the numbers of distinct domains in a feed and the number of exclusive domains is also shown graphically in Figure 4.1; the left plot shows this relationship for live domains, and the right plot shows it for tagged domains. In both plots, the x axis denotes to the number of distinct domains on a logarithmic scale, while the y axis denotes number of exclusive domains in each feed on a logarithmic scale. Dotted lines denote fixed exclusive domain proportions. For example, the Hyb feed lies just under the 100% line, indicating that most of its live domains—just over 65%—are exclusive.

Figure 4.1 makes apparent that the Hu and Hyb feeds make the greatest contribution in terms of the distinct number of domains they contribute as well as the number of domains exclusive to each feed. The number of tagged domains

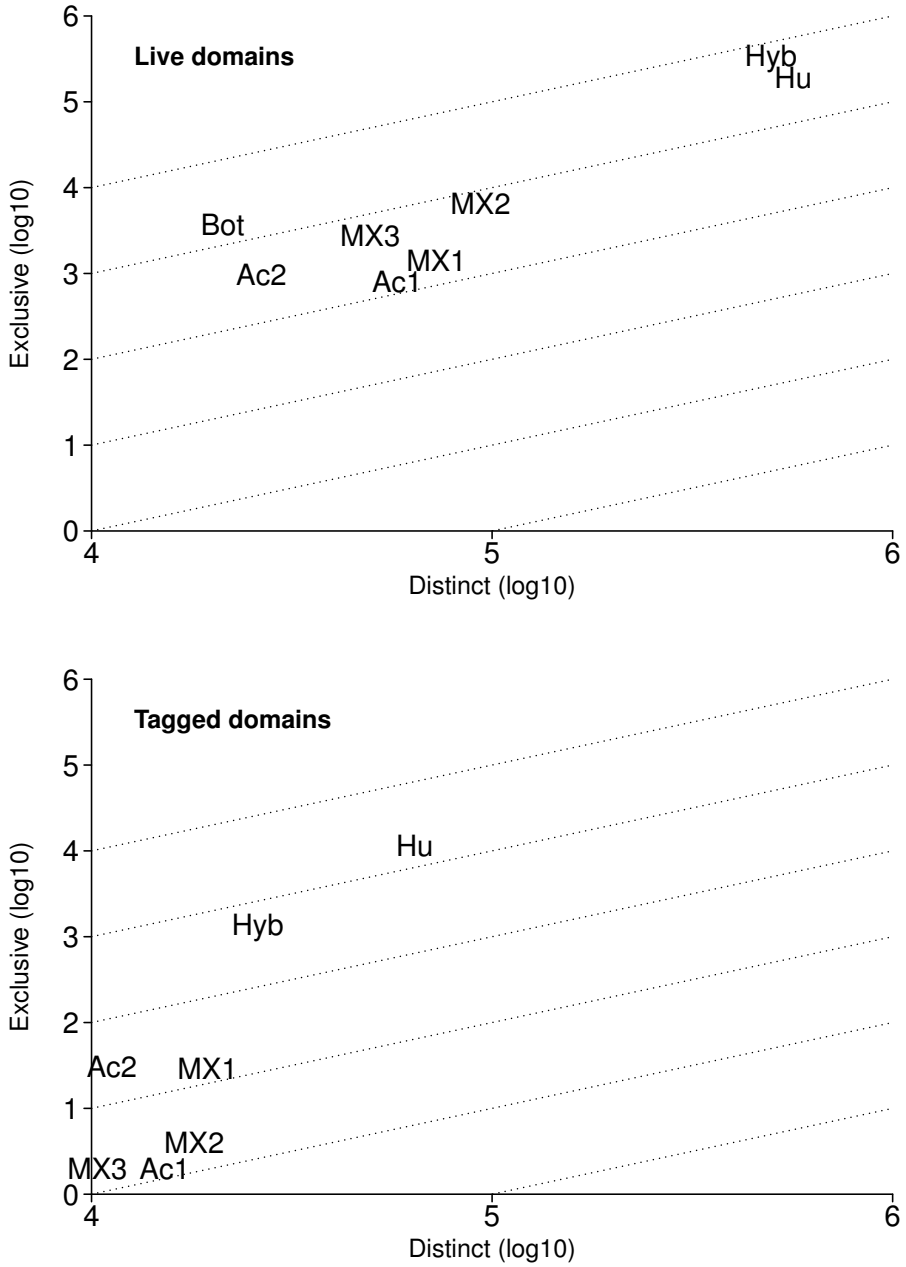


Figure 4.1: Relationship between the total number of domains contributed by each feed and the number of domains exclusive to each.

is about an order of magnitude less in each feed than the number of live domains, suggesting that spam belonging to the categories represented by the tagged domains—online pharmacies, replica shops, and counterfeit software stores—is a small fraction of all spam. This is not so, however. As we will see in Section 4.3.3, these domains dominate the feeds in volume.

Figure 4.1 and Table 4.3 put the Bot feed in perspective. Although extremely valuable in identifying which domains are being spammed by which botnet, its contribution to the big picture is more limited. None of its tagged domains were exclusive, not a surprising fact given that bots are renowned for indiscriminate high-volume spamming. The roughly 3,700 exclusive live domains in the Bot feed are likely the result of the domain poisoning described earlier (Section 4.3.1), as fewer than 1% of all domains were legitimate (Table 4.2).

Pairwise comparison In the preceding discussion of exclusive contribution, we were implicitly asking which feed, if it were excluded, would be missed the most. Next we consider the question of each feed's differential contribution with respect to another feed. Equivalently, we are asking how many domains from one feed are also in another. (Removing non-responsive and benign domains is particularly important for a meaningful comparison here.)

Figure 4.2 shows pairwise domain overlap as a matrix, with live domains plotted on the left and tagged domains on the right. For two feeds A and B, the cell in row A column B shows how many domains (percent) from feed B are in feed A, as well as the absolute number of such domains. Formally, the top and bottom numbers show

$$|A \cap B|/|B| \quad \text{and} \quad |A \cap B|.$$

For example, in the left-hand matrix, the cell in row Ac_1 column MX_1 indicates that Ac_1 and MX_1 have approximately 47,000 live domains in common, and that this number is 65% of the MX_1 feed. Note that these same 47,000 live domains constitute 81% of the Ac_1 feed (row MX_1 column Ac_1). In addition, the right-most column, labeled *All* contains the union of all domains across all feeds. The numbers

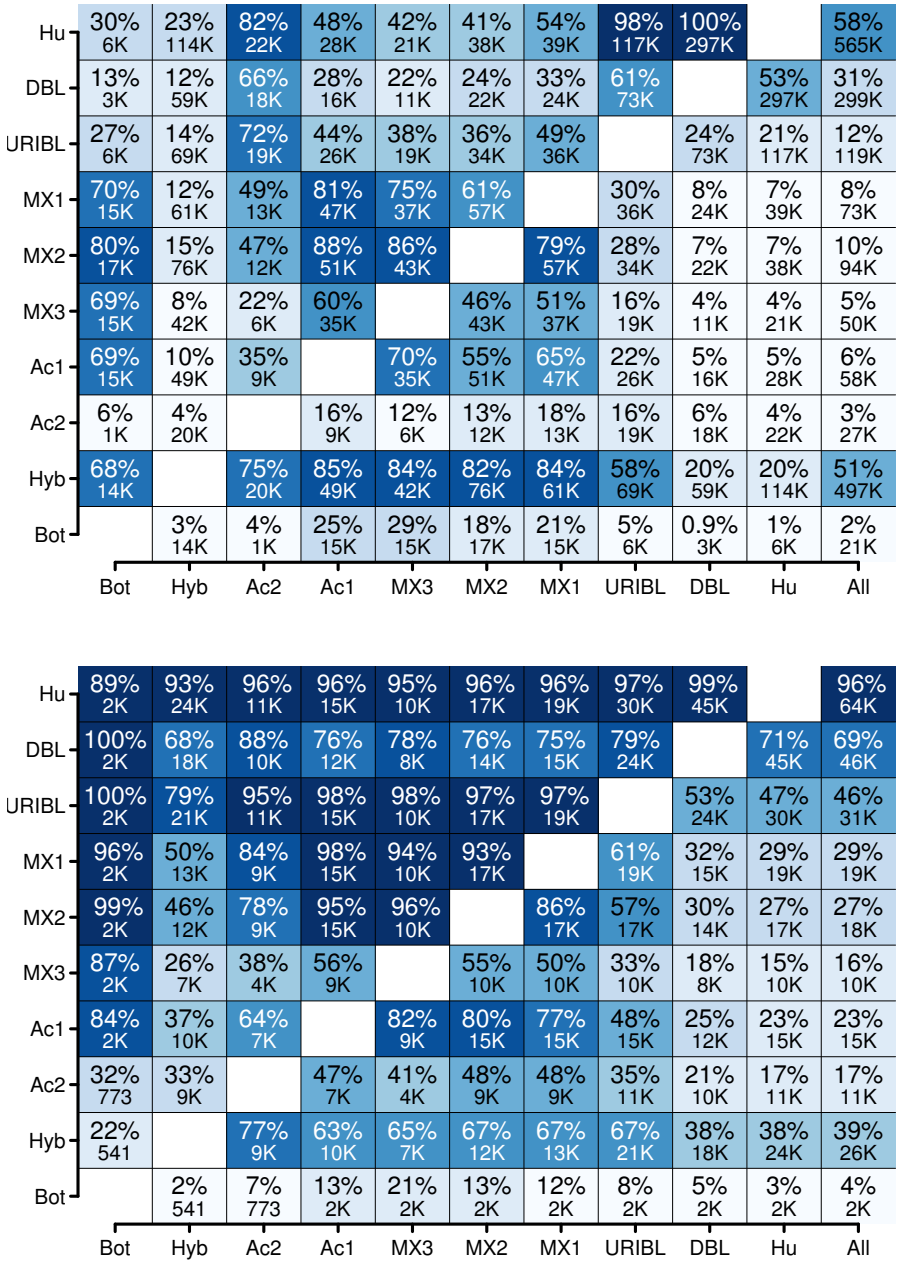


Figure 4.2: Pairwise feed domain intersection, shown for live (top) and tagged domains (bottom).

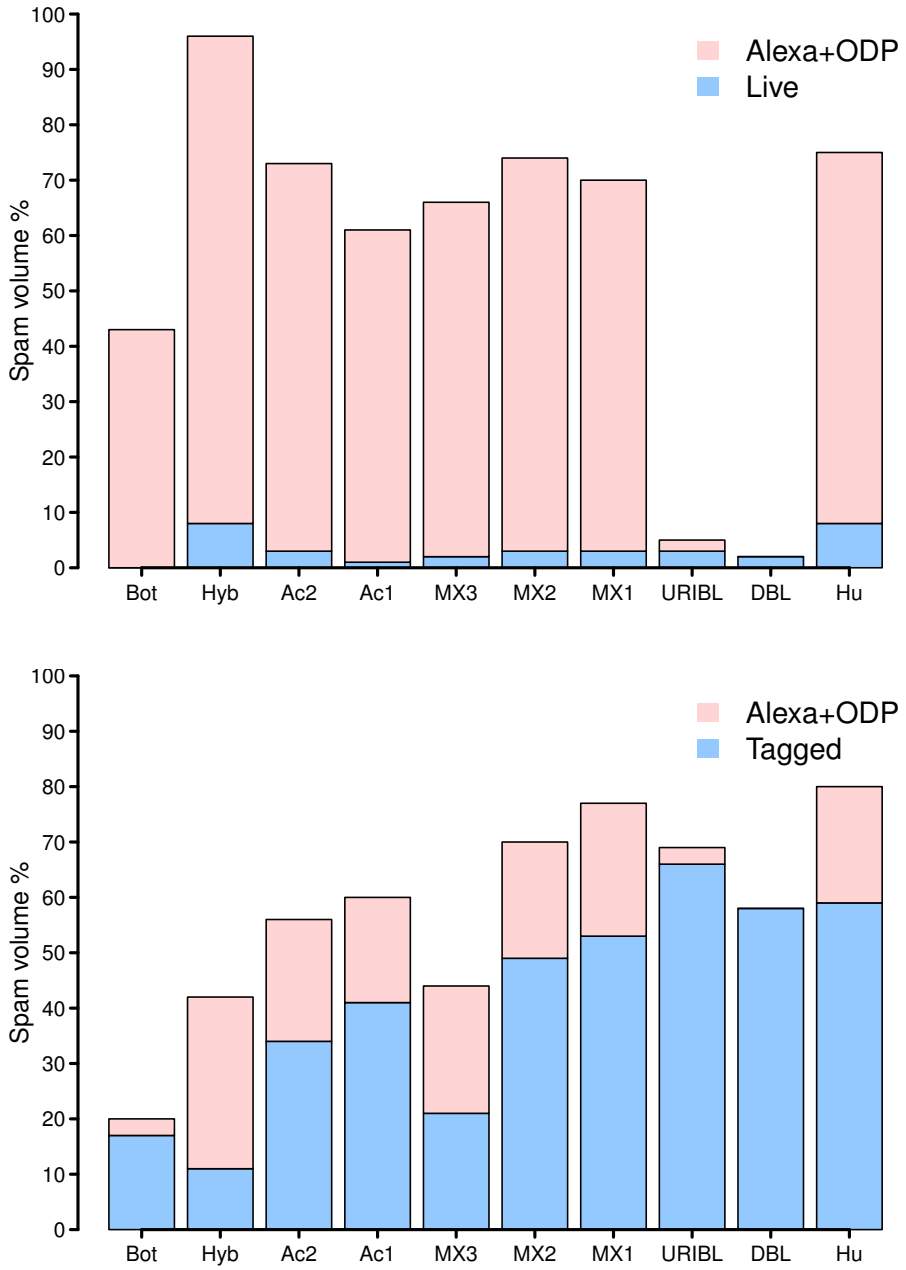


Figure 4.3: Feed volume coverage shown for live (top) and tagged domains (bottom).

in the *All* column thus indicate what proportion of all spam domains (the union of all feeds) is covered by a given feed.

Figure 4.2 once again highlights the coverage of the Hu and Hyb feeds. The Hyb feed covers 51% of all live domains (the union of all non-blacklist feeds), while the Hu feed covers 58%; the two feeds together covering 98% (not shown in matrix) of all live domains. When restricted to tagged domains only (Figure 4.2 right), the coverage of the Hu feed is an astounding 96%, while the contribution of Hyb drops to 39%. In fact, restricting the domains to tagged only (right-hand matrix) excludes many of the benign domains appearing in Hyb from the *All* column, *improving* the coverage of most feeds with respect to *All*.

Figure 4.2 also reveals that most feeds—especially Ac_1, MX_1, MX_2, and MX_3—are quite effective at capturing bot-generated spam domains. These feeds range from 12% to 21% bot-generated (tagged domains), although the true number is likely higher given the limited set of bots included in the Bot feed. In turn, URIBL is quite effective at capturing these honeypot feeds (MX_1, MX_2, MX_3, Ac_1, and Ac_2), and both blacklists considerably overlap each other. Moreover, blacklists have a non-trivial overlap with the Hu feed. Despite these higher numbers, though, a gap still exists, as blacklists cannot replace the human identified dataset. Overall, this is a strong indicator of the strength of human-identified feeds, while also stressing the significance of blacklists.

Volume

While there are millions of URLs and thousands of domains spammed daily, the number of messages in which each appears can vary dramatically. We call the number of messages advertising a domain the *volume* of that domain. Here we consider the coverage of each feed with respect to the relative volume of spam it covers. To estimate this quantity, we solicited the help of a large Web mail provider to measure the volume of spam domains at their incoming mail servers.

The incoming mail oracle We refer to this data source as our *incoming mail oracle*. For this measurement, we collected all live domains seen across all feeds,

and submitted them to the cooperating mail provider. The provider reported back to us the number of messages (normalized) containing each spam domain, as seen by their incoming mail servers over five days during the measurement period. This provider handles mail for hundreds of millions of users. Although the measurement collected is not a perfectly uniform sample of all spam globally, we believe it to be a reasonable representative. Given the limited duration of the measurement—five days versus three months of feed data—these results should be interpreted with caution.

Figure 4.3 shows the volume of spam covered by the live and tagged domains in each feed. Recall that both live and tagged domains specifically exclude domains listed in the Alexa 1 million and domains appearing in the Open Directory Project listings (Section 4.3.1). In the figure, we have included the volume due to these Alexa and ODP domains occurring in each feed, shown stacked on top of the live and tagged volume bars. Before removing Alexa and ODP domains, the volume of live domains is dominated by these potential false positives. Among tagged domains, the volume attributed to Alexa and ODP domains (before exclusion) is much lower. These are domains which may have been used by the spammer as a redirection mechanism, either by abusing a legitimate service or via compromise. Of the feeds, the blacklists show the highest purity, as noted in Section 4.3.1.

With the Alexa and ODP domains excluded from the set of tagged domains, the URIBL blacklist provides the greatest coverage, followed by the Hu feed and DBL blacklist. At the opposite end, the Hyb feed provides only about a sixth of the coverage (by tagged domain volume) compared to URIBL, DBL, and Hu. Although it has nearly an order of magnitude more domains, its spam volume coverage is less than the Bot feed. One possibility is that this feed contains spam domains not derived from e-mail spam.

Affiliate Programs

Up to this point, our focus has been the domains occurring in feeds, with the implicit understanding that domains represent a spam campaign. The relationship between a campaign and the domains it uses can be complex: a domain may

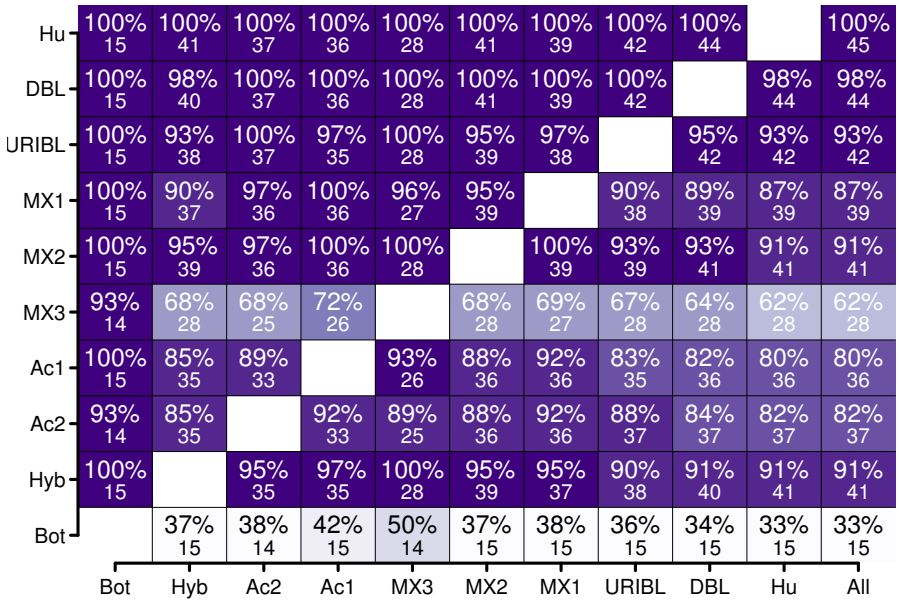


Figure 4.4: Pairwise feed similarity with respect to covered affiliate programs.

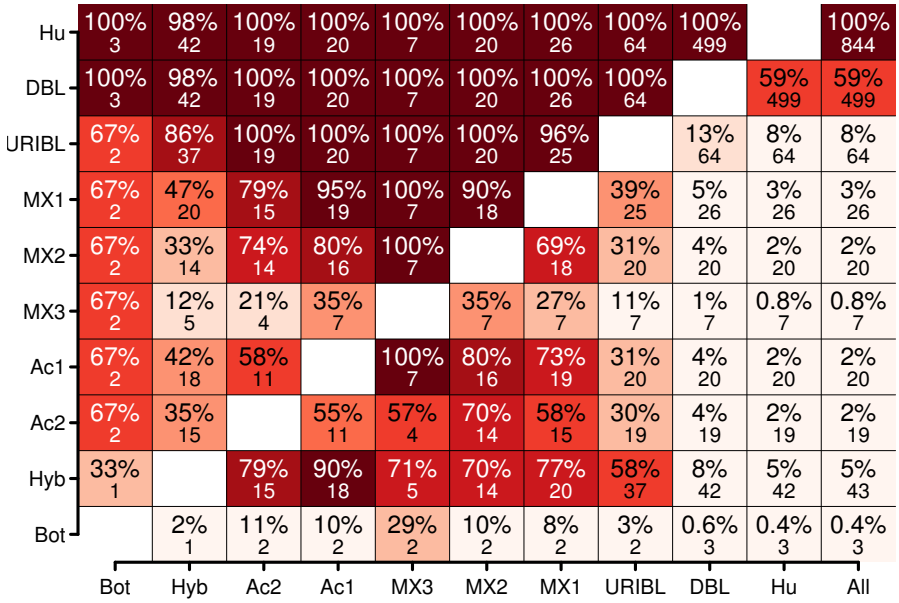


Figure 4.5: Pairwise feed similarity with respect to covered RX-Promotion affiliate identifiers.

be used in multiple campaigns, and a campaign may continuously cycle through several domains.

In fact, there is another level of structure beyond domains and campaigns: affiliate programs. Today, spammers operate primarily as advertisers, working with an affiliate program and earning a commission (typically 30–50%). The affiliate program handles Web site design, payment processing, customer service and fulfillment [76].

The prior Web crawling effort (Chapter 3) identified 45 leading affiliate programs specializing in pharmaceutical sales, replica luxury goods, and "OEM" software (this classification includes all the major players in each category that advertise via spam). We use the classification results of this project to define the tagged domains (Section 4.3.1). Here, we explore the tags themselves, that is, the affiliate programs associated with domains. Specifically, we consider the coverage of each feed with respect to *affiliate programs*.

Figure 4.4 shows the proportion of programs covered by each feed, relative to other feeds and all feeds combined. The representation is the same as Figure 4.2: each cell indicates the number of programs represented by the two feeds given by the row and column labels, and the top number of each cell expresses this intersection relative to the second program (identified by the column). For example, the cell in row MX_1 column Hyb indicates that 37 affiliate programs are seen in both feeds, and that these 37 programs represent approximately 90% of the 41 programs appearing in the Hyb feed.

Generally, most feeds do a good job covering all programs. The MX_3 feed has the second worst coverage, covering only 62% of all programs. Not surprisingly, the Bot feed has the worst coverage: only 15 programs. This poor coverage is partly because botnet operators frequently act as affiliates themselves and thus only advertise for a modest number of programs where they have spent the time to establish themselves. Even a botnet for rent will have a modest number of users and thus any given botnet will typically spam for a small number of programs in a given time.

RX-Promotion

In the affiliate marketing model for spam-advertised goods, the store site is usually hosted by the affiliate program itself, and not the spammer. For the affiliate program to determine which affiliate should receive credit for a sale, the URL itself must uniquely identify the affiliate. The most common mechanism is to assign each (major) affiliate a handful of dedicated domains.[3] Any sales generated via those domains are automatically credited to the appropriate affiliate.

One program, RX-Promotion, embeds an affiliate identifier in the page source of the storefront itself. This embedding allowed us to extract affiliate identifiers and map them to domains. In total, we were able to identify 846 distinct affiliate identifiers.

Affiliate coverage Figure 4.5 shows the pairwise feed comparison matrix for RX-Promotion affiliate identifier coverage. Similar to affiliate program coverage, the human-identified feed Hu contributes the largest number of distinct affiliates. In this case, however, the difference between Hu and other feeds is more pronounced, with more than 40% of the affiliates found exclusively in Hu. The remaining feeds follow the same pattern as before. The MX honeypots (especially MX_1 and MX_2) continue to offer slightly greater coverage than seeded honey accounts (Ac_1 and Ac_2). Most striking is the paucity of affiliate IDs in the botnet feeds, confirming our earlier suspicion that botnet-originated affiliate program spam is associated with a single individual (the botnet operator).

Revenue coverage The ultimate measure of an affiliate's success is the revenue he generates for the program. By this measure, a feed's value lies not in how many affiliates it covers, but in how much revenue it covers.

In a recent dispute between rival pharmacy affiliate programs, a number of RX-Promotion documents were leaked to the public [55]. One such document available to us lists annual revenue generated by each RX-Promotion affiliate in 2010. Using these revenue statistics, we calculate affiliate coverage *weighted by*

[3] New domains must be constantly registered and assigned, as domains quickly become ineffective because of blacklisting.

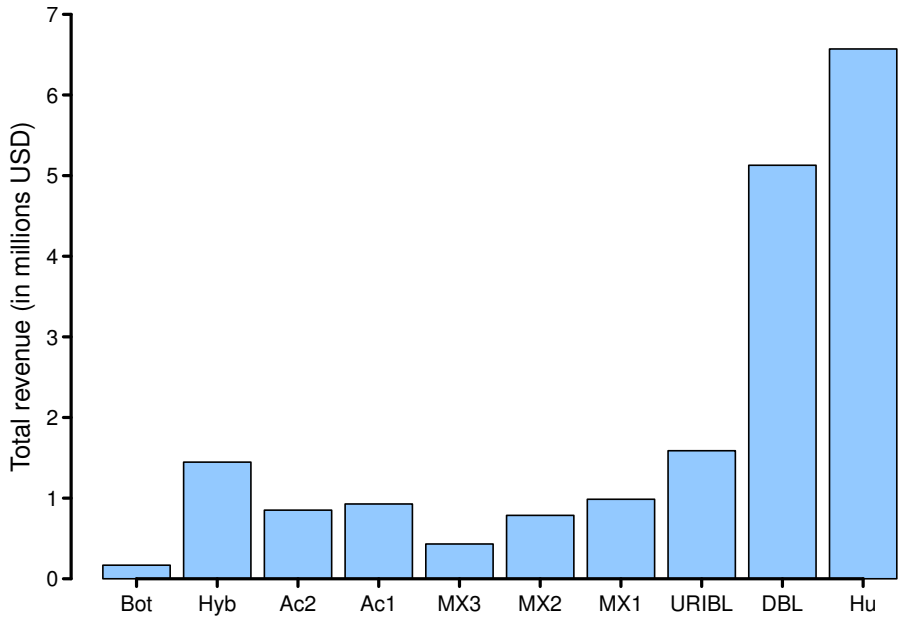


Figure 4.6: RX-Promotion affiliate coverage of each feed weighted by each affiliate's 2010 revenue.

affiliate revenue. Figure 4.6 shows the revenue-weighted affiliate coverage of each feed.

The domains advertised in the smaller campaigns only found in Hu and DBL generate an order of magnitude more revenue than the sites advertised by bots and typically five times more than those sites seen in MX and honey account feeds. Overall, the results generally follow the affiliate coverage shown in Figure 4.5, although the revenue-weighted results indicate a bias toward higher-revenue affiliates. While DBL covers only 59% of Hu affiliates, these affiliates represent over 78% of revenue covered by Hu.

4.3.3 Proportionality

An anti-spam system seeks to identify as many spam messages as possible, and in this context volume is a natural measure of a domain's importance. A blacklist that identifies the top 100 spammed domains by volume will identify more spam than a list of the same size consisting of infrequent domains. Similarly, domain take-downs are best prioritized to target high-volume domains first. To

make these judgments, a spam domain feed must contain not only the domains themselves, but also their observed volume.

It happens that some of our feeds do provide volume information: each domain is listed with the number of times a domain was seen in spam, allowing relative domain volume and rank to be estimated. This section considers only feeds with volume information; the Hyb feed, Hu feed and both blacklist feeds (DBL and URIBL) have no associated volume information and are thus excluded from this analysis.

Empirical domain distribution and rank The volumes associated with each domain define an empirical distribution on domains. That is, if a domain i has reported volume c_i in a feed, then the empirical domain probability distribution is c_i/m, where m is the total volume of the feed (i.e., $m = \sum_i c_i$).

Variation distance Variation distance (also called "statistical difference" in some areas of Computer Science) is a straightforward metric frequently used to compare distributions. Formally, given two probability distributions (feeds) P and Q, let p_i be the empirical probability of domain i in P, and q_i the probability of the same domain in Q. (If a domain does not occur in a feed, its empirical probability is 0.) The variation distance is given by:

$$\delta = \frac{1}{2} \sum_i |p_i - q_i|.$$

Variation distance takes on values between 0 and 1, where $\delta = 0$ if and only if $P = Q$ (domains have the same probability in both), and $\delta = 1$ if P and Q are disjoint (no domains in common). Figure 4.7 shows pairwise measures of variation distance of tagged domains. (Because we round values to two decimal places, a variational distance of 1 in the figure may still allow for some domain overlap.)

Kendall rank correlation coefficient Variation distance places more weight on more frequently occurring domains. In some cases, only the relative ranks of domains are of interest, and not the magnitudes of the empirical probabilities. The Kendall rank correlation coefficient (also called Kendall's tau-b) allows us to

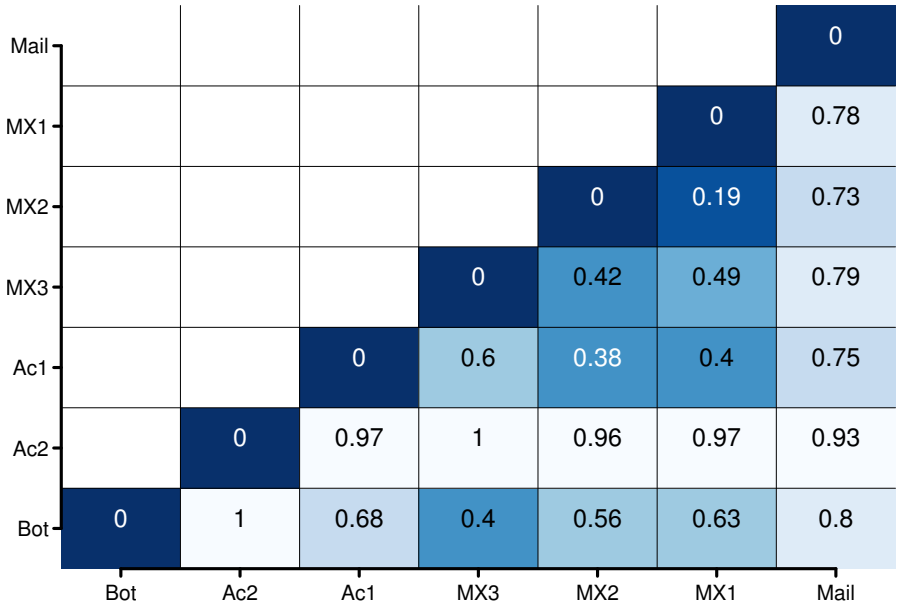


Figure 4.7: Pairwise variational distance of tagged domains frequency across all feeds. Shading is inverted (larger values are darker).

compare the relative ranking of domains between two distributions. In the case where all probabilities are distinct,

$$\tau = \frac{1}{n(n-1)} \sum_{i \neq j} \operatorname{sgn}\big[(p_i - p_j)(q_i - q_j)\big].$$

where $\operatorname{sgn}(x)$ is the familiar signum function. The sum is over all domains common to both feeds being compared, and n is the number of such domains. The Kendall rank correlation coefficient takes on values between -1 and 1, with 0 indicating no correlation, 1 indicating perfect positive correlation, and -1 indicating perfect negative correlation. If there are ties, i.e., $p_i = p_j$ or $q_i = q_j$ for some $i \neq j$, the denominator $n(n-1)$ must be adjusted to keep the full range between -1 to 1; we refer the reader to an appropriate Statistics textbook for details.

Figure 4.8 shows the pairwise tagged domain Kendall rank correlation coefficient between all feed pairs.

Pairwise comparison Figures 4.7 and 4.8 show how well each pair of feeds agree in domain volume and rank. (The *Mail* column will be described shortly.) Qualitatively, both variation distance and Kendall rank correlation coefficient show

similar results. The MX honeypot feeds and the Ac_1 honey account feeds exhibit similar domain distributions; these four also have many domains in common as seen in Figure 4.2.

The Bot feed brings a small number of domains, many of which also occur in the MX honeypot feeds and the Ac_1 feed (Figure 4.2). The volume of these domains, however, is significant; so much so, that in terms of domain proportions, the MX_3 feed is more like the Bot feed than any other feed, including the remaining MX honeypots.

The similarity in coverage and empirical domain probability distributions indicates that, roughly speaking, one MX honeypot feed is as good as another. By their nature, MX honeypots are targets of high-volume spammers who spam randomly-generated names at all registered domains. By this process, it is just as easy to stumble upon one MX honeypot as another.

Comparison to real mail In Section 4.3.2 we reported on the fraction of incoming spam—as seen by a major Web mail provider—covered by each feed. Here we use the same incoming mail oracle to determine the real-world relative volumes of spam domains, and compare those numbers to the relative domain volumes reported by each feed. We use only tagged domains appearing in at least one spam feed in the comparison: in the calculation of δ and τ, we set $p_i = 0$ for any domain i not appearing in the union of all spam feeds.

The *Mail* column in Figures 4.7 and 4.8 shows these results. The MX_2 feed comes closest to approximating the domain volume distribution of live mail, with Ac_1 coming close behind. As with coverage, the Ac_2 feed stands out as being most unlike the rest.

4.3.4 Timing

For both sides of the spam conflict, time is of the essence. For a spammer, the clock starts ticking as soon as a domain is advertised. It is only a matter of time before the domain is blacklisted, drastically reducing the deliverability of spam. While a domain is still clean, the spammer must maximize the number of

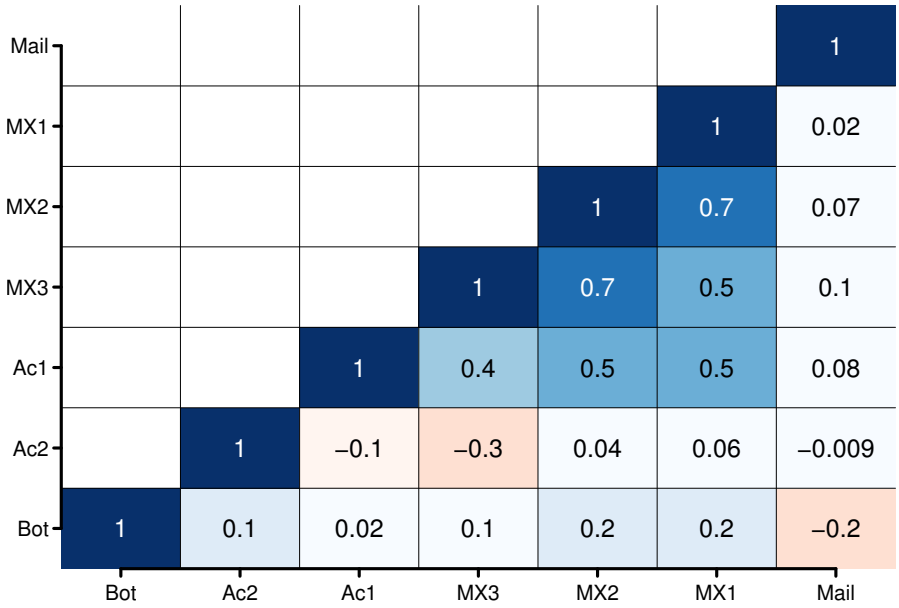


Figure 4.8: Pairwise Kendall rank correlation coefficient of tagged domain frequency across all feed pairs.

messages delivered to potential customers. On the other side, blacklist maintainers strive to identify and blacklist spam domains as quickly as possible to maximize the volume of spam captured.

In this section we consider how well each spam feed captures the timing of spam campaigns. Specifically, we identify how quickly each feed lists spam domains, and, for feeds driven by live mail, how accurately they identify the end of a spam campaign. Unless noted otherwise, we restrict our analysis to tagged domains because we have the most confidence in their provenance.

Ideally, we would like to compare the time a domain first appears in spam with the time it first appears in a spam feed. Lacking such perfect knowledge about the start of each spam campaign, we instead take the *earliest* appearance time of a domain across all feeds as the *campaign start time*, and the *last* appearance time of a domain in live mail-based feeds as the *campaign end time*. For this analysis, we exclude the Bot feed because its domains have little overlap with the other feeds. As a result, including them greatly diminishes the number of domains that

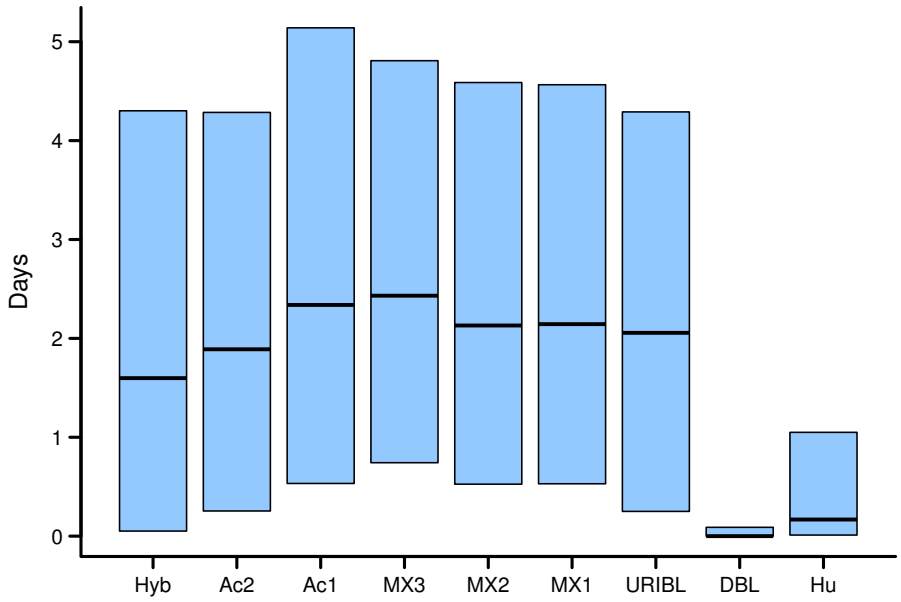


Figure 4.9: Relative first appearance time of domains in each feed. Campaign start time calculated from all feeds except Bot. Solid lines are medians; boxes range from the 25th to the 75th percentile.

appear in the intersection of the feeds, and hence the number of domains that we can consider.

Taking the campaign start time and end time as described above, we define the *relative first appearance time* for a domain in a particular feed to the time between campaign start and its first appearance in the feed. In other words, we take campaign start time as "time zero" and calculate the relative first appearance time relative to this time. Put another way, the relative first appearance time is thus the *latency* of a feed with respect to a domain.

First Appearance Time

Figure 4.9 shows the distribution of relative first appearance times of domains in each feed. The bottom of the box corresponds to the 25th percentile, the top denotes the 75th percentile, and the solid bar inside the box denotes the median.

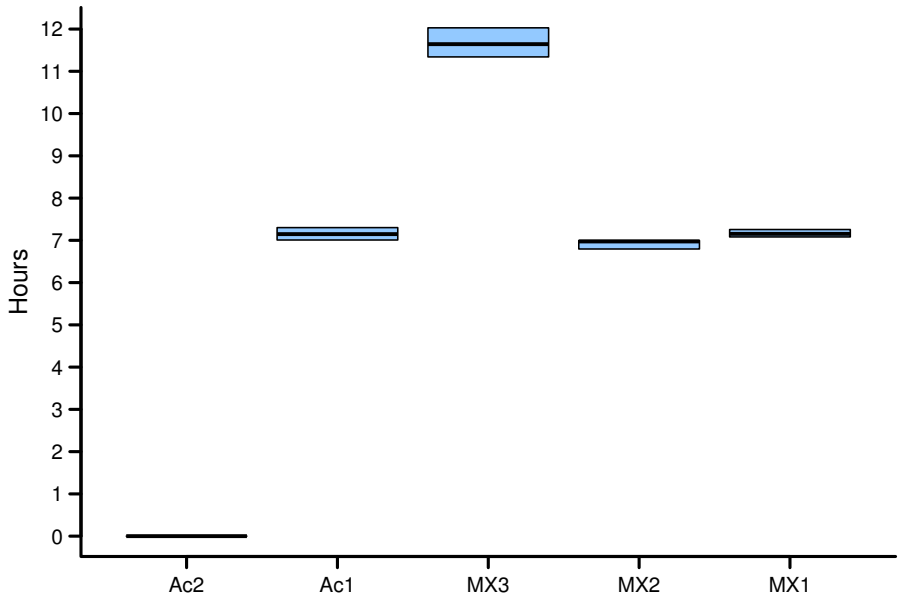


Figure 4.10: Relative first appearance time of domains in each feed. Campaign start time calculated from MX honeypot and honey account feeds only. Solid lines are medians; boxes range from the 25th to the 75th percentile.

Both Hu and DBL are excellent early warnings of spam campaigns since they see most domains soon after they are used. The Hu feed sees over 75% of the domains within a day after they appear in any feed, and 95% within three days; DBL is delayed even less, with over 95% appearing on the blacklist within a day. Once again, the nature of these feeds lends themselves to observing wide-spread spam activity quickly: Hu has an enormous net for capturing spam, while DBL combines domain information from many sources. In contrast, the other feeds have much later first appearance times: they do not see roughly half of the domains until two days have passed, 75% until after four days, and 95% after ten. Operationally, by the time many of the domains appear in these feeds, spammers have already had multiple days to monetize their campaigns.

Of course, these results depend on both the set of domains that we consider and the sets of feeds we use to define campaign start times. When performing the same analysis on the larger set of live domains that appear in the same set of feeds, the first appearance times remain very similar to Figure 4.9: even for the broader

set of domains, Hu and DBL see the domains multiple days earlier than the other feeds.

However, changing the set of *feeds* we consider does change relative first appearance times. Figure 4.10 shows similar results as Figure 4.9, but with the Hu, Hyb, and blacklist feeds removed. (We chose these feeds because, as discussed further below, they all contain domains reported by users, which affects the last appearance times of domains.) Restricting the feeds we use to determine campaign start times reduces the total set of domains, but also increases the likelihood that a domain appears in all traces. When we focus on just the MX honeypot and account traces in this way, we see that relative to just each other they continue to have consistent first appearance times with each other, but the relative first appearance times are now very short (roughly less than a day). As with other metrics, these results show that timing estimates are quite relative and fundamentally depend on the feeds being considered.

Last Appearance Time

Last appearance times are often used to estimate when spam campaigns end. Figure 4.11 shows the time between the last appearance of a domain in a feed and the domain's campaign end time. As with Figure 4.10 we focus on only a subset of the feeds where the last appearance of a domain likely corresponds to when a spammer stopped sending messages using the domain: the MX honeypots and honeypot account feeds. Feeds like Hu, Hyb, and the blacklists all have domains reported by users. Since user reports fundamentally depend on when users read their mail and report spam, they may report spam long after a spammer has sent it.

Consistent with the first appearance times for the honeypot feeds, the feeds are similar to each other for last appearance times as well. The difference with the baseline are relatively short (a day or less), but have longer tails (the 95th percentiles for most are over a week).

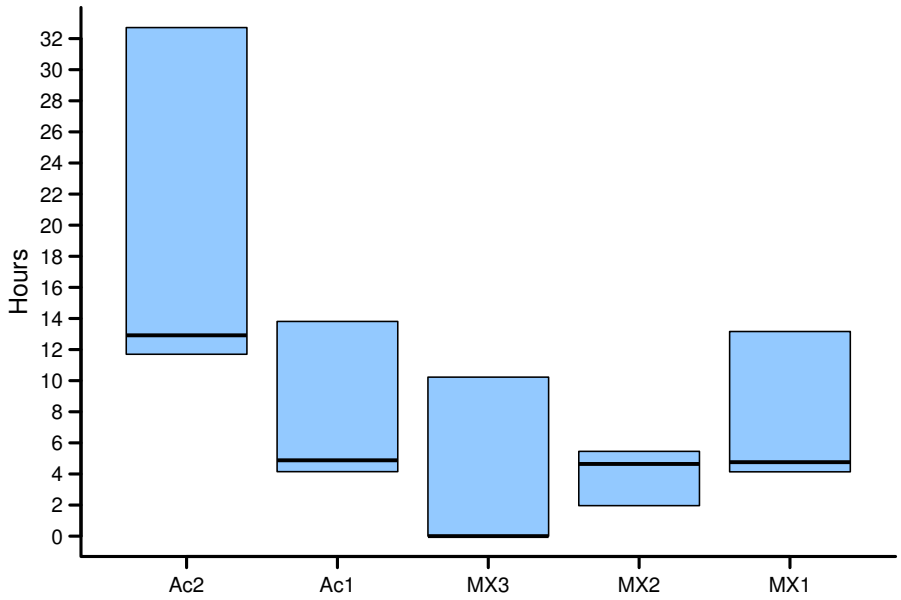


Figure 4.11: Distribution of differences between the last appearance of a domain in a particular and the domain campaign end calculated from an aggregate of the same five feeds. Solid lines are medians; boxes range from the 25th to the 75th percentile.

Duration

Another common metric for spam campaigns is their duration: how long spammers advertise domains to attract customers. Figure 4.12 shows the differences in time durations of domains advertised in spam as observed by each feed relative to estimated campaign duration (campaign end time minus campaign start time). For each feed we calculate the *lifetime* of a domain in the feed using the first and last appearance of a domain just in that feed. Then we compute the difference between the domain lifetime in a feed and the estimated campaign duration. (Campaign duration is computed from the same five feeds and is always at least as long as the domain lifetime in any feed.) The box plots in the graph summarize the distributions of these differences across all domains in each feed.

The differences in duration estimates for the honeypot feeds are also consistent with their first and last appearance time estimates. The duration estimates across feeds are similar to each other, the duration estimates differ from the base-

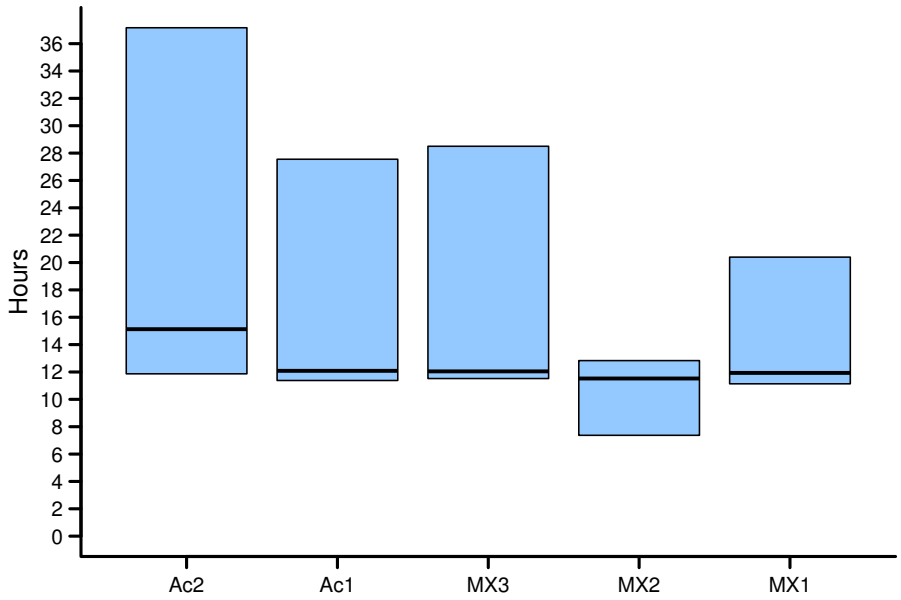


Figure 4.12: Distribution of differences between domain lifetime estimated using each feed and the domain campaign duration computed from an aggregate of those same five feeds. Solid lines are medians; boxes range from the 25th to the 75th percentile.

line by less than a day for half of the domains and roughly a day for 75% of the domains. The distribution tails are longer, though, with outliers underestimating durations by multiple weeks.

4.4 Summary

Most measurement studies focus on using data to infer new facts about the world. This goal is why we measure things—to put truth on an empirical footing. However, occasionally it is necessary to perform introspective studies such as this one to understand the limits of what we can conclude from available data.

While our analysis is not comprehensive, we have found significant variation among the ten feeds we did study. Based on these findings we recommend that researchers consider four different challenges whenever using spam data:

- Limited purity. Even the best spam feeds include benign domains and these domains should be anticipated in analyses. We should identify the "kinds" of benign domains that appear in a dataset and determine if their existence will bias results—in particular when spam feed data will be correlated with other data sources.

- Coverage limitations. MX and honey account spam sources are inherently biased towards loud broad campaigns. If we desire a broader view of what is advertised via spam and are unable to strike an arrangement with a large e-mail provider, operational domain blacklists are the next best source of such information.

- Temporal uncertainty. Studies of spam campaign timing should recognize how timing error can be introduced via different feeds. Botnet-based feeds are among the best for timing information, but naturally coverage is limited. Other feeds provide highly accurate "onset" information (e.g., blacklists and human-identified feeds) but may not provide a correspondingly accurate ending timestamp. This area is one where combining the features of different feeds may be appropriate.

- Lack of proportionality. It is tempting to measure the prevalence of one kind of spam in a feed and extrapolate to the entire world—"25% of all spam advertises eBooks!" or "My spam filter can block 99.99% of all spam". However, the significant differences in the makeup of the feeds we have studied suggests that any such conclusion is risky. For example, spam filter results trained on botnet output may have little relevance to a large Web mail provider. In general, we advise making such claims based on knowledge of the source data set. For example, MX-based honeypots may be appropriate for characterizing relative prevalence among distinct high volume spam campaigns.

While it is important to be aware of the limitations and challenges of spam feeds, an even more interesting question is what feeds one should use for related studies. The clear answer, as shown by our results, is that there is no perfect

feed. Instead, the choice should be closely related to the questions we are trying to answer. It is still possible, though, to provide some general guidelines that would apply for most cases:

- Human identified feeds, which are provided by large mail providers, will usually be the best choice for most studies. They provide a clear advantage when it comes to coverage, due to their wide exposure, and allow for visibility inside low-volume campaigns. They do so with reasonable purity, but due to the presence of the human factor, filtering is required. On the other hand, we should avoid human identified feeds when we are interested in timing, and especially last appearance information.

- If it is not possible to get access to human identified feeds, due to their limited availability, high-quality blacklist feeds offer very good coverage and first appearance information. They also offer the best purity since they are usually commercially maintained, and have low false positives as their primary goal. Similar to human identified feeds, they are less useful for studies that rely on last appearance or duration information.

- When working with multiple feeds, the priority should be to obtain a set that is as diverse as possible. Additional feeds of the same type offer reduced added value, and this situation is especially true in the case of MX honeypot feeds.

- It is very challenging to obtain accurate information regarding volume and provide conclusions that apply to the entirety of the spam problem. Given our limited view into the global spam output, all results are inherently tied to their respective input datasets.

In a sense, the spam research community is blessed by having so many different kinds of data sources available to it. In many other measurement regimes the problem of bias is just as great, but the number of data sources on hand is far fewer. However, with great data diversity comes great responsibility. It is no longer reasonable to take a single spam feed and extrapolate blindly without

validation. This dissertation provides a basic understanding of the limitations of existing feeds and provides a blueprint for refining this understanding further.

Chapter 4, in part, is a reprint of the material as it appears in Proceedings of the ACM Internet Measurement Conference 2012. Pitsillidis, Andreas; Kanich, Chris; Voelker, Geoffrey M.; Levchenko, Kirill; Savage, Stefan. The dissertation author was the primary investigator and author of this paper.

Chapter 5

Botnet Judo: Fighting Spam with Itself

The first component of the spam value chain is advertising. This is essentially the sending of spam e-mails to users, in an effort to attract potential customers to the advertised Web sites. Botnets are considered to be today's primary source of e-mail spam in terms of volume [52]. We have described botnets in Chapter 3 and analyzed the properties of feeds generated by botnet spam in Chapter 4. Stopping spammers from delivering e-mail messages to users has traditionally been considered the primary approach for defending against spam. In this chapter, we describe a filtering system which focuses primarily on botnet spam, and evaluate the feasibility of disrupting the spam value chain at the advertising level.

5.1 Introduction

Reactive defenses, in virtually any domain, depend on the currency of their intelligence. How much do you know about your opponent's next move and how quickly can you act on it? Maximizing the time advantage of such information is what drives governments to launch spy satellites and professional sports teams to employ lip readers. By the same token, a broad range of competitors, from commodities traders to on-line gamers, all seek to exploit small time advantages

to deliver large overall gains. As Benjamin Franklin famously wrote, "Time is money."

Today's spammers operate within this same regime. Receivers install filters to block spam. Spammers in turn modify their messages to evade these filters for a time, until the receiver can react to the change and create a new filter rule in turn. This pattern, common to spam, anti-virus and intrusion detection alike, dictates that the window of vulnerability for new spam depends on how quickly it takes the receiver to adapt.

We advocate shrinking this time window by changing the vantage point from which we fight spam. In particular, it is widely documented that all but a small fraction of today's spam e-mail is transmitted by just a handful of distributed botnets [31, 52], and these, in turn, use template-based macro languages to specify how individual e-mail messages should be generated [37, 81]. Since these templates describe *precisely* the range of polymorphism that a spam campaign is using at a given point in time, a filter derived from these templates has the potential to identify such e-mails *perfectly* (i.e., never generating false positives or false negatives).

In this chapter we describe *Judo*, a system that realizes just such an approach—quickly producing precise mail filters essentially equivalent to the very templates being used to create the spam. However, while spam templates can be taken directly from the underlying botnet "command and control" channel, this approach can require significant manual effort to reverse engineer each unique protocol [33, 37]. Instead, we use a *black-box* approach, in which individual botnet instances are executed in a controlled environment and the spam messages they attempt to send are analyzed on-line. We show that using only this stream of messages we can quickly and efficiently produce a comprehensive regular expression that captures all messages generated from a template while avoiding extraneous matches. For example, in tests against live botnet spam, we find that by examining roughly 1,000 samples from a botnet (under a minute for a *single* energetic bot, and still less time if multiple bots are monitored) we can infer precise filters that produce zero false positives against non-spam mail, while matching virtually

all subsequent spam based on the same template. While template inference is by no means foolproof (and accordingly we examine the anticipated arms race), we believe that this approach, like Bayesian filtering, IP blacklisting and sender reputation systems, offers a significant new tool for combating spam while being cheap and practical to deploy within existing anti-spam infrastructure.

In particular, we believe our work in this chapter offers three contributions. First, we produce a general approach for inferring e-mail generation templates in their entirety, subsuming prior work focused on particular features (e.g., mail header anomalies, subject lines, URLs). Second, we describe a concrete algorithm that can perform this inference task in near real-time (only a few seconds to generate an initial high-quality regular expression, and fractions of a second to update and refine it in response to subsequent samples), thereby making this approach feasible for deployment. Finally, we test this approach empirically against live botnet spam, demonstrate its effectiveness, and identify the requirements for practical use.

5.2 Template-based Spam

Spam templates are akin to form letters, consisting of text interspersed with substitution macros that are instantiated differently in each generated message. Unlike form letters, spam templates use macros more aggressively, not only to personalize each message but also to avoid spam filtering based on message content. Figure 5.1 shows a typical template from the Storm botnet (circa Feb. 2008) together with an instance of spam created from the template and a Judo regular expression signature generated from 1,000 message instances (Section 5.3 below describes our algorithm for generating such signatures). Note that this regular expression was created by observing the messages sent by the botnet, and *without any prior knowledge of the underlying template* (we show the template for reference only).

Template Elements. We can divide the bulk of macros used in templates into two types: *noise* macros that produce a sequence of random characters from a

```
Received: from %^C0%^P%^R2-6^%:qwertyuiopasdfghjklzxcvbnm^%.%^P%^R2-6^%: ▷
          qwertyuiopasdfghjklzxcvbnm^%^% ([%^C6%^I^%.%^I^%.%^I^%.%^I^%^%]) ▷
          by %^A^% with Microsoft SMTPSVC(%^Fsvcver^%); %^D^%
Message-ID: <%^Z^%.%^R1-9^%0%^R0-9^%0%^R0-9^%0%^R0-9^%@%^C1%^Fdomains^%^%>
Date: %^D^%
From: <%^Fnames^%@%^V1^%>
User-Agent: Thunderbird %^Ftrunver^%
MIME-Version: 1.0
To: %^O^%
Subject: %^Fstormline^%
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Content-Transfer-Encoding: 7bit

%^G%^Fstormline^% http://%^Fstormlink2^%/^%
```

```
Received: from auz.xwzww ([132.233.197.74]) by dsl-189-188-79-63.prod- ▷
          infinitum.com.mx with Microsoft SMTPSVC(5.0.2195.6713); Wed, 6 Feb ▷
          2008 16:33:44 -0800
Message-ID: <id012345.99066044044@experimentalist.org>
Date: Wed, 6 Feb 2008 16:33:44 -0800
From: <katiera@experimentalist.org>
User-Agent: Thunderbird 2.0.0.14 (Windows/20080421)
MIME-Version: 1.0
To: victim@spam-target.com
Subject: Get Facebook's FBI Files
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Content-Transfer-Encoding: 7bit

FBI may strike Facebook http://GoodNewsGames.com/
```

```
From: <.+@.+\..+>
User-Agent: Thunderbird 2\.0\.0\.14 \(Windows/200(80421\)|70728\))
MIME-Version: 1\.0
To: .+@.+\..+
Subject: (Get Facebook's FBI Files|...|The F\.B\.I\. has a new way of ▷
      tracking Facebook)
Content-Transfer-Encoding: 7bit

(FBI may strike Facebook|FBI wants instant access to Facebook|...|The ▷
F\.B\.I\. has a new way of tracking Facebook) http://(GoodNewsGames|...| ▷
StockLowNews)\.com/
```

Figure 5.1: Fragment of a template from the Storm template corpus, together with a typical instantiation, and the regular expression produced by the template inference algorithm from 1,000 instances. The subject line and body were captured as dictionaries (complete dictionaries omitted to save space). This signature was generated without any prior knowledge of the underlying template.

specified character alphabet, and *dictionary* macros that choose a random phrase from a list given in a separate "dictionary" included with the template. For example, in the Storm template shown in Figure 5.1, the "`%^P ... ^%`" macro generates a string of a given length using a given set of characters, while the "`%^F ... ^%`" macro inserts a string from a list in the specified dictionary. Similar functionality exists in other template languages (e.g., the *rndabc* and *rndsyn* macros in Reactor Mailer [81]). In general, noise macros are used to randomize strings for which there are few semantic rules (e.g., message-ids), while dictionary macros are used for content that must be semantically appropriate in context (e.g., subject lines, sender e-mail addresses, etc.). In addition to these two classes, there are also special macros for dates, sender IP addresses, and the like. We deal with such macros specially (Section 5.3.2).

Thus, to a first approximation, our model assumes that a message generated from a template will consist of three types of strings: invariant text, *noise* macros producing random characters as described above, and *dictionary* macros producing a text fragment from a fixed list (the dictionary).

Real-time Filtering. The nature of template systems documented by our earlier work [37] as well as Stern [81] suggests that templates can be used to *identify*—and thus filter—any mail instances generated from a template. It is relatively straightforward to convert a Storm template, for example, into a regular expression by converting macros to corresponding regular expressions: noise macros become repeated character classes, and dictionary macros become a disjunction of the dictionary elements. Such a regular expression signature will then match all spam generated from the template. Unfortunately, obtaining these templates requires reverse-engineering the botnet "command and control" channel—a highly time-consuming task. Instead, our template inference algorithm, described in the next section, creates such signatures by observing multiple instances of the spam from the same template.

Figure 5.2 shows a diagram of a Judo spam filtering system. The system consists of three components: a "bot farm" running instances of spamming botnets in a contained environment; the signature generator; and the spam filter. The

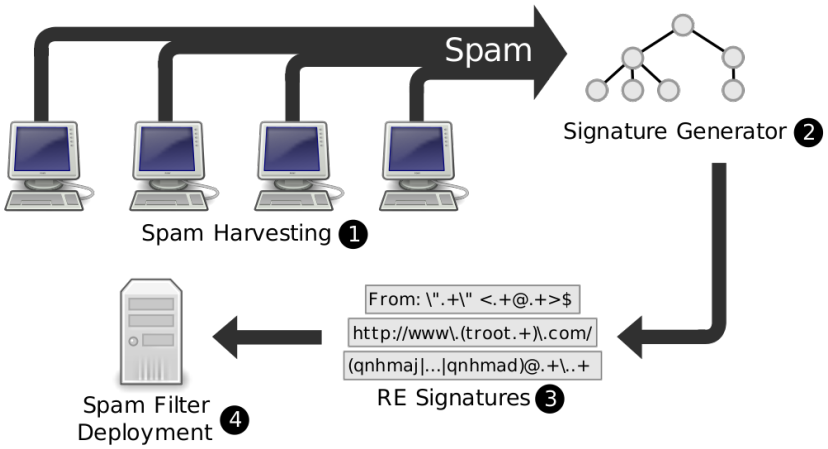


Figure 5.2: Automatic template inference makes it possible to deploy template signatures as soon as they appear "in the wild:" bots (❶) running in a contained environment generate spam processed by the Judo system (❷); signatures (❸) are generated in real time and disseminated to mail filtering appliances (❹).

system intercepts all spam sent by bots and provides the specimens to a signature generator. The signature generator maintains a set of regular expression signatures for spam sent by each botnet, updating the set in real time if necessary. We can then immediately disseminate new signatures to spam filters.

System Assumptions. Our proposed spam filtering system operates on a number of assumptions. First and foremost, of course, we assume that bots compose spam using a template system as described above. We also rely on spam campaigns employing a small number of templates at any point in time. Thus, we can use templates inferred from one or a few bot instances to produce filters effective at matching the spam being sent by all other bot instances. This assumption appears to hold for the Storm botnet [38]; in Section 5.4.3 we empirically validate this assumption for other botnets. Finally, as a simplification, our system assumes that the first few messages of a new template do not appear intermixed with messages from other new templates. Since we infer template content from spam output, rather than extracting it directly from botnet command and control messages, interleaved messages from several new templates will cause us to infer an amalgamated template—the product of multiple templates—which is consequently less

precise than ideal. This assumption could be relaxed by more sophisticated spam pre-clustering, but we do not explore doing so in this work.

5.3 The Signature Generator

In this section we describe the Judo system, which processes spam generated by bots and produces regular expression signatures. The system operates in real time, updating the set of signatures immediately in response to new messages. We begin with a description of the template inference algorithm—the heart of Judo—which, given a set of messages, generates a matching regular expression signature. We then describe how we incorporate domain knowledge, such as the header-body structure of e-mail messages, into the basic algorithm. Finally, we show how we use the template inference algorithm to maintain a *set* of signatures matching all messages seen up to a given point in time.

5.3.1 Template Inference

The template inference algorithm produces a regular expression signature from a set of messages assumed to be generated from the same spam template. As described in Section 5.2, a template consists of invariant text and macros of two types: *noise* macros which generate a sequence of random characters, and *dictionary* macros which choose a phrase at random from a list (the dictionary). Proceeding from this model, the algorithm identifies these elements in the text of the input messages, and then generates a matching regular expression. Throughout this section, we use the simple example in Figure 5.3 to illustrate the steps of our algorithm.

Anchors

The first step of the algorithm is to identify invariant text in a template, that is, fragments of text that appear in every message. We call such fragments *anchors* (because they "anchor" macro-generated portions of a message). Invariant text like "Best prices!" and "http://" in Figure 5.3 are examples of anchors.

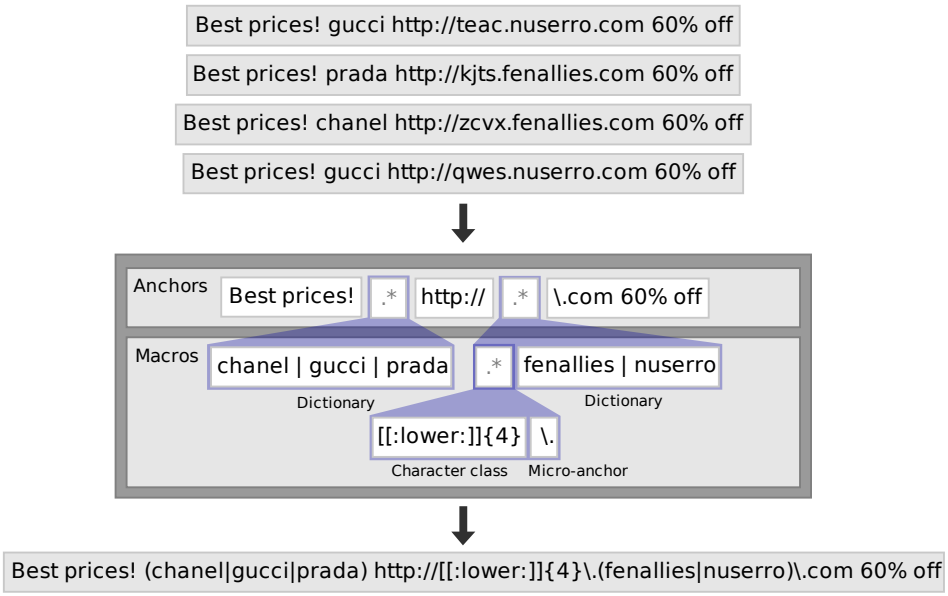


Figure 5.3: Template inference algorithm example showing excerpts from template-based spam messages, the invariant text and macros inferred from the spam, and the resulting regular expression signature.

We start by extracting the longest ordered set of substrings having length at least q that are common to every message. Parameter q determines the minimum length of an anchor. Setting $q = 1$ would simply produce the longest common subsequence of the input messages, which, in addition to the anchors, would contain common characters, such as spaces, which do not serve as useful anchors. Large minimum anchor lengths, on the other hand, may exclude some useful short anchors. In our experiments with a variety of spamming botnets, we found that $q = 6$ produces good results, with slightly smaller or larger values also working well.

We note that finding this ordered set of substrings is equivalent to computing the longest common subsequence (LCS) over the q-gram sequence of the input, i.e., the sequence of substrings obtained by sliding a length-q window over each message. Unfortunately, the classic dynamic programming LCS algorithm is quadratic in the length of its inputs.

As an optimization we first identify all substrings of length at least q that are common to all input messages (using a suffix tree constructed for these mes-

sages). We then find longest common subsequence of substrings (i.e., treating each substring as a single "character") using the classic dynamic programming algorithm. Typical messages thus collapse from several thousand bytes to fewer than ten common substrings, resulting in essentially linear-time input processing.

Macros

Following anchor identification, the next step is to classify the variant text found *between* anchors. In the simplest case this text corresponds to a single macro, but it may also be the back-to-back concatenation of several macros. In general, our algorithm does not distinguish between a single macro and the concatenation of two or more macros, unless the macros are joined by "special" characters — non-alphanumeric printing characters such as "@" and punctuation. We call these special characters *micro-anchors* and treat them as normal anchors. Our next step, then, is to decide whether text between a pair of anchors is a dictionary macro, a noise macro, or a micro-anchor expression, itself consisting of a combination of dictionary and noise macros separated by micro-anchor characters.

Dictionary Macros. In a template, a dictionary macro is instantiated by choosing a random string from a list. In Figure 5.3, for instance, the brand names "gucci", "prada", and "chanel" would be generated using such a macro. Given a set of strings found between a pair of anchors, we start by determining whether we should represent these instances using a dictionary macro or a noise macro. Formally speaking, every macro can be represented as a dictionary, in the sense that it is a set of strings. However, if we have not seen every possible instance of a dictionary, a dictionary representation will be necessarily incomplete, leading to false positives. Thus we would like to determine whether what we have observed is the *entirety* of a dictionary or not. We formulate the problem as a hypothesis test: the null hypothesis is that there is an unobserved dictionary element. We take the probability of such an unobserved element to be at least the empirical probability of the least-frequent observed element. Formally, let n be the number of distinct strings in m samples, and let f_n be the empirical probability of the least-frequent element (i.e., the number of times it occurs in the sample, divided by m). Then

the probability of observing fewer than n distinct strings in m samples drawn from a dictionary containing $n+1$ elements is at most $(1-f_n/(1+f_n))^m$. For the brand strings in Figure 5.3, this value is at most $(1-0.25/1.25)^4 \approx 0.41$. In practice, we use a 99% confidence threshold; for the sake of example, however, we will assume the confidence threshold is much lower. If the dictionary representation is chosen, we group the distinct strings $\alpha_1, \ldots, \alpha_n$ into a disjunction regular expression $(\alpha_1 | \ldots | \alpha_n)$ to match the variant text. Otherwise, we check whether it might be a micro-anchor expression.

Micro-Anchors. A micro-anchor is a substring that consists of non-alphanumeric printing characters too short to be a full anchor. Intuitively, such strings are more likely to delimit macros than ordinary alphanumeric characters, and are thus allowed to be much shorter than the minimum anchor length q. We again use the LCS algorithm to identify micro-anchors, but allow only non-alphanumeric printing characters to match. In Figure 5.3, the domain names in the URLs are split into smaller substrings around the "." micro-anchor. Once micro-anchors partition the text, the algorithm performs the dictionary test on each set of strings delimited by the micro-anchors. Failing this, we represent these strings as a noise macro.

Noise Macros. If a set of strings between a pair of anchors or micro-anchors fails the dictionary test, we consider those strings to be generated by a noise macro (a macro that generates random characters from some character set). In Figure 5.3, the host names in the URLs fail the dictionary test and are treated as a noise macro. The algorithm chooses the smallest character set that matches the data from the set of POSIX character classes `[:alnum:]`, `[:alpha:]`, etc., or a combination thereof. If all strings have the same length, the character class expression repeats as many times as the length of the string, i.e., the regular expression matches on both character class and length. Otherwise, we allow arbitrary repetition using the "$*$" or "$+$" operators.[1]

[1] We also experimented with the alternative of allowing a range of lengths, but found such an approach too restrictive in practice.

When generating each signature, we also add the constraint that it must contain at least one anchor or dictionary node. If this constraint is violated, we consider the signature as *unsafe* and discard it.

5.3.2 Leveraging Domain Knowledge

As designed, the template inference algorithm works on arbitrary text. By exploiting the structure and semantics of e-mail messages, however, we can "condition" the input to greatly improve the performance of the algorithm. We do two kinds of "conditioning," as described next.

Header Filtering. The most important pre-processing element of the Judo system concerns headers. We ignore all but the following headers: "MIME-Version," "Mail-Followup-To," "Mail-Reply-To," "User-Agent," "X-MSMail-Priority," "X-Priority," "References," "Language," "Content-Language," "Content-Transfer-Encoding," and "Subject." We specifically exclude headers typically added by a mail transfer agent. This is to avoid including elements of the spam collection environment, such as the IP address of the mail server, in signatures. We also exclude "To" and "From" headers; if the botnet uses a list of e-mail addresses in alphabetical order to instantiate the "To" and "From" headers, portions of the e-mail address may be incorrectly identified as anchors.

The resulting headers are then processed individually by running the template inference algorithm on each header separately. A message must match all headers for a signature to be considered a match.

Special Tokens. In addition to dictionary and noise macros, bots use a small class of macros for non-random variable text. These macros generate dates, IP addresses, and the like. If the output of a date macro, for example, were run through the template inference algorithm, it would infer that the year, month, day, and possibly hour are anchors, and the minutes and seconds are macros. The resulting signature would, in effect, "expire" shortly after it was generated. To cope with this class of macros, we perform the following pre- and post-processing steps. On input, we replace certain well-known tokens (currently: dates, IP addresses, and multi-part message delimiters) with special fixed strings that the template

inference algorithm treats as anchors. After the algorithm produces a regular expression signature, it replaces these fixed strings with regular expressions that capture all instances of the macro.

5.3.3 Signature Update

The Judo system processes e-mail messages generated by a botnet, creating a set of signatures that match those messages. The need for a signature *set*, rather than a single signature, arises because several templates may be in use at the same time. Recall that the template inference algorithm relies heavily on anchors (common text) to locate macros. If the template inference algorithm were given messages generated from different templates, only strings common to all templates would be identified as anchors, leading the algorithm to produce a signature that is too general. Ideally, then, we would like to maintain one signature per template.

Unfortunately, because we do not know which template was used to generate a given message, we cannot simply group messages by template and apply the template inference algorithm separately to each. The situation is not as dire as it seems, however. If we already *have* a good signature for a template, we can, by definition, easily identify messages generated by the template. Thus, if new templates are deployed incrementally, we can use the template inference algorithm only on those messages which do not already match an existing signature.

On receiving a new message, the algorithm first checks if the message matches any of its existing signatures for the botnet in question. If it does, it ignores the message, as there is already a signature for it. Otherwise, it places the message into a *training buffer*. When the training buffer fills up, it sends the message to the template inference algorithm to produce a new signature. The size of the training buffer is controlled by a parameter k, which determines the trade-off between signature selectivity and training time. If the training buffer is too small, some dictionaries may be incomplete—the template inference algorithm will emit a noise macro instead. On the other hand, a very large training buffer means waiting longer for a usable signature. A very large training buffer increases the chances of mixing messages from two different templates, decreasing signature

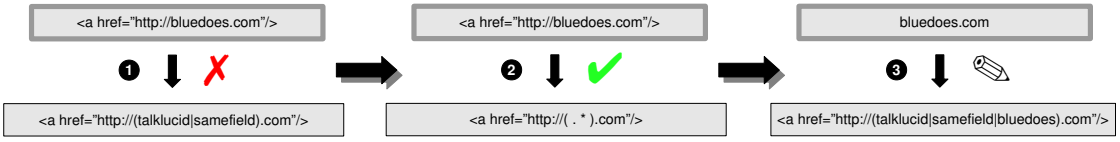


Figure 5.4: The second chance mechanism allows the updating of signatures: when a new message fails to match an existing signature (❶), it is checked again only against the anchor nodes (❷); if a match is found, the signature is updated accordingly (❸).

accuracy. Thus we would like to use a training buffer as small as necessary to generate good signatures.

In experimenting with the signature generator, we found that no single value of k gave satisfactory results. The Storm botnet, for example, tended to use large dictionaries requiring many message instances to classify, while in the extreme case of a completely invariant template (containing no macros), signature generation would be delayed unnecessarily, even though the first few messages are sufficient to produce a perfect signature. We developed two additional mechanisms to handle such extreme cases more gracefully: the *second chance mechanism* and *pre-clustering*.

Second Chance Mechanism. In many cases, a good signature can be produced from a small number of messages, even though many more are required to fully capture dictionaries. Furthermore, the dictionary statistical test is rather conservative (to avoid false negatives). To combine the advantage of fast signature deployment with the eventual benefits of dictionaries, we developed a "second chance" mechanism allowing a signature to be updated after it has been produced. When a new message fails to match an existing signature, we check if it would match any existing signatures consisting of anchors only. Such *anchor signatures* are simply ordinary regular expression signatures (called *full signatures*) with the macro-based portions replaced by the ".∗" regular expression. If the match succeeds, the message is added to the training buffer of the signature and the signature is updated. This update is performed incrementally without needing to rerun a new instance of the inference algorithm.

Pre-Clustering. Whereas the second chance mechanism helps mitigate the effects of a small training buffer, pre-clustering helps mitigate the effects of a large training buffer. Specifically, a large training buffer may intermix messages from different templates, resulting in an amalgamated signature. With pre-clustering, unclassified messages are clustered using *skeleton signatures*. A skeleton signature is akin to an anchor signature used in the second chance mechanism, but is built with a larger minimum anchor size q, and as such is more permissive. In our experiments, we set $q = 6$ for anchor signatures and $q = 14$ for skeleton signatures. Further evaluation indicated that slight variations of these values only have a minimal impact on the overall performance.

The pre-clustering mechanism works as follows. Each skeleton signature has an associated training buffer. When a message fails to match a full signature or an anchor signature (per the second chance mechanism), we attempt to assign it to a training buffer using a skeleton signature. Failing that, it is added to the unclassified message buffer. When this buffer has sufficient samples (we use 10), we generate a skeleton regular expression from them and assign them to the skeleton's training buffer. When a training buffer reaches k messages ($k = 100$ works well), a full signature is generated. The signature and its training buffer are moved to the signature set, and the signature is ready for use in a production anti-spam appliance. In effect, the pre-clustering mechanism mirrors the basic signature update procedure (with skeleton signatures instead of full and anchor signatures).

As noted in Section 5.2, our system does not currently support a complete mechanism for pre-clustering messages into different campaigns. Instead, our current mechanism relies on the earlier assumption that the first few messages of a new template do not appear intermixed with messages from other new templates—hence our decision to group together every 10 new unclassified messages and treat them as a new cluster. Note that it is perfectly acceptable if these first 10 messages from a new template are interleaved with the messages of a template for which we already have generated a signature. In this case, the messages of the latter will be filtered out using the existing regular expression, and only the messages of the

former will enter the unclassified message buffer. From our experience, this choice has provided us with very good results, although a more sophisticated clustering method could be a possible future direction.

5.3.4 Execution Time

Currently the execution time of the template inference algorithm observed empirically is linear in the size of the input. Based on our experience, the length of messages generated by different botnets varies significantly. The largest average length we have observed among all botnets was close to 6,000 characters. The selected training buffer size k (introduced in Section 5.3.3), along with the average length of e-mails, determine the total size of the input. Under this worst-case scenario, the algorithm requires 2 seconds for $k = 50$ and 10 seconds for $k = 500$, on a modest desktop system. Signature updates execute much faster, as they are performed incrementally. The execution time in this case depends on a wide range of factors, but an average estimate is between $50 - 100$ ms. The focus of our implementation has always been accuracy rather than execution time, thus we expect several optimizations to be possible.

5.4 Evaluation

The principal requirements of a spam filtering system are that it should be both *safe* and *effective*, meaning that it does not classify legitimate mail as spam, and it correctly recognizes the targeted class of spam. Our goal is to experimentally demonstrate that Judo is indeed safe and effective for filtering botnet-originated spam.

Our evaluation consists of three sets of experiments. In the first, we establish the effectiveness of the template inference algorithm on spam generated synthetically from actual templates used by the Storm botnet. Next, we run the Judo system on *actual* spam sent by four different bots, measuring its effectiveness against spam generated by the same bot. In our last set of experiments, we execute

Table 5.1: Legitimate mail corpora used to assess signature safety throughout the evaluation.

Corpus	Messages
SpamAssassin 2003 [53]	4,150
TREC 2007[86] (non-spam only)	25,220
lists.gnu.org [2] (20 active lists)	479,413
Enron [34]	517,431

a real deployment scenario, training and testing on *different* instances of the same bot. In all cases, Judo was able to generate effective signatures.

In each set of experiments, we also assess the safety of the Judo system. Because Judo signatures are so specific, they are, by design, extremely safe; signatures generated in most of our experiments generated no false positives. We start by describing our methodology for evaluating signature safety.

5.4.1 Signature Safety Testing Methodology

By their nature, Judo signatures are highly specific, targeting a single observed campaign. As such, we expect them to be extremely *safe*: Judo signatures should never match legitimate mail. We consider this to be Judo's most compelling feature.

The accepted metric for safety is the *false positive rate* of a signature with respect to a corpus of legitimate (non-spam) mail, i.e., the proportion of legitimate messages incorrectly identified as spam. Throughout the evaluation we report the false positive rate of the generated signatures; Section 5.4.5 presents a more detailed analysis.

Corpora. We used four corpora of legitimate mail, together totaling over a million messages, summarized in Table 5.1: the SpamAssassin "ham" corpus dated February 2003 [53], the 2007 TREC Public Spam Corpus restricted to messages labelled non-spam [86], 20 active mailing lists from lists.gnu.org spanning August 2000 to April 2009 [2], and the Enron corpus [34].

Age bias. Recall that Judo signatures consist of regular expressions for a message's header as well as the body. To avoid potential age bias, we tested our signatures with all but the subject and body regular expressions removed. This is to safeguard against age-sensitive headers like "User-Agent" causing matches to fail on the older corpora. It is worth noting that using only subject and body is a significant handicap because the remaining headers can act as additional highly discriminating features.

5.4.2 Single Template Inference

The template inference algorithm is the heart of the Judo system. We begin by evaluating this component in a straightforward experiment, running the template inference algorithm directly on training sets generated from single templates. By varying the size of the training set, we can empirically determine how much spam is necessary to achieve a desired level of signature effectiveness. Our metric of effectiveness is the *false negative rate* with respect to instances of spam generated from the same template. In other words, the false negative rate is the proportion of spam test messages that do not match the signature. Because the template is known, we can also (informally) compare it with the generated signature. Figure 5.1 from Section 5.2 shows an example.

Methodology

We generated spam from real templates and dictionaries, collected during our 2008 study of Storm botnet campaign orchestration [38]. The templates covered three campaigns: a self-propagation campaign from August 1–2, 2008 (4,210 templates, 1,018 unique), a pharmaceutical campaign from the same time period (4,994 templates, 1,271 unique), and several low-priced stock campaigns between June and August 2008 (1,472 templates, all unique). Each one of these templates

Table 5.2: False negative rates for spam generated from Storm templates as a function of the training buffer size k. Rows report statistics over templates. The stock spam table also shows the number of templates s for which a signature was generated (for self-propagation and pharmaceutical templates, a signature was generated for every template); in cases where a signature was not generated, every instance in the test set was counted as a false negative. At $k = 1000$, the false positive rate for all signatures was zero.

(a) Self-propagation and pharmaceutical spam.

	False Negative Rate			
k	95%	99%	Max	Avg
1000	0%	0%	0%	0%
500	0%	0%	2.53%	<0.01%
100	0%	0%	0%	0%
50	0%	0%	19.15%	0.06%
10	45.45%	58.77%	81.03%	14.16%

(b) Stock spam.

		False Negative Rate			
k	s	95%	99%	Max	Avg
1000	99.8%	0%	0.22%	100%	0.21%
500	81.8%	100%	100%	100%	18.21%
100	55.0%	100%	100%	100%	45.04%
50	42.9%	100%	100%	100%	57.25%
10	40.9%	100%	100%	100%	62.13%

had its own unique set of dictionary files. Both the self-propagation and pharmaceutical templates contained URLs; the stock campaign templates did not.[2]

For convenience, we generated Storm spam directly from these templates (rather than having to operate actual Storm bots) by implementing a Storm template instantiation tool based on our earlier reverse-engineering work on this botnet [37]. For each of the 10,676 templates, we generated 1,000 training instances and an additional 4,000 instances for testing.

We ran the template inference algorithm on the 1,000 training messages and assessed the false negative rate of the resulting signature using the 4,000-message test set.[3] To better understand the performance of the Judo system, we then pushed it to the "breaking point" by using smaller and smaller training sets to generate a signature. We use k to denote the training set size.

Results

As expected, the template inference algorithm generated effective signatures. Both self-propagation and pharmaceutical campaigns were captured perfectly, with no false negatives. For the stock campaign, 99% of the templates had a false negative rate of 0.22% or lower.

Table 5.2 also shows Judo's performance as we decrease the number of training instances k. In effect, k is a measure of how "hard" a template is. We separate results for templates with URLs (self-propagation and pharmaceutical) and without (stock) to establish that our algorithm is effective for both types of spam. Rows correspond to different numbers of training messages, and columns to summary statistics of the range of the false negative rates. For example, when training a regular expression on just $k = 10$ messages from a URL-based campaign

[2]Although less common in terms of Storm's overall spam volume [38], we included non-URL spam to determine how much of our system's effectiveness stems from learning the domains of URLs appearing in the spam, compared to other features.

[3]We choose 1,000 as the training set size in part because Storm generated roughly 1,000 messages for each requested work unit. We note that 1,000 messages represents a very small training set compared with the amount of spam generated by bots from a given template: in our 2008 study of spam conversion rates [33] we observed almost 350 million spam messages for one spam campaign generated from just 9 templates. Thus, we can generate potent signatures nearly immediately after a new template is deployed, and use that signature for the duration of a large spam campaign.

template, the signature yielded a false negative rate of 45.45% or less on 95% of such templates, and a false negative rate of 81.03% for the template that produced the worst false negative rate. Such high false negative rates are not surprising given just 10 training instances; with just 50 training messages, it exhibits no false negatives for 99% of such Storm templates.[4]

Column s in Table 5.2b also shows the number of templates for which a signature was generated (all templates resulted in a signature for self-propagation and pharmaceutical spam). Recall from Section 5.3 that we discard a signature if it is found to be unsafe. This was the only case in our evaluation where this occurred. For such templates for which we do not generate a signature, we calculate a 100% false negative rate. For this test, our input was stock templates which did not contain URLs. These messages were also very short and consisted entirely of large dictionaries: characteristics that make messages particularly difficult to characterize automatically from a small number of samples.

Signatures from the self-propagation and pharmaceutical templates produced no false positives in three of the four legitimate mail corpora, regardless of the value of k. For the stock templates, the result was again zero for $k = 1000$. We present a detailed breakdown of these results in Section 5.4.5.

5.4.3 Multiple Template Inference

In the previous section we examined the case of spam generated using a single template. In practice, a bot may be switching between multiple templates without any indication. In this part of the evaluation we test the algorithm on a "live" stream of messages generated by a single bot, classifying each message as it is produced.

[4]One peculiarity in Table 5.2a merits discussion: the 2.53% maximum false negative rate for $k = 500$ arises due to a single template out of the 9,204 total; every other template had a false negative rate of 0%. For this template, when transitioning from $k = 100$ to $k = 500$ training samples the algorithm converted the "Subject" header from a character class to a dictionary. The mismatches all come about from a single dictionary entry missing from the signature because it did not appear in the 500 training messages.

Table 5.3: Cumulative false negative rate as a function of training buffer size k and classification delay d for spam generated by a single bot instance. The "Sigs" column shows the number of signatures generated during the experiment (500,000 training and 500,000 testing messages). All signatures produced zero false positives with the only exception being the signatures for Rustock.

Cumulative False Negative Rate						
Botnet	k \ d	0	50	100	500	Sigs
Mega-D	50	0.11%	0.09%	0.07%	0.05%	5
	100	0.16%	0.13%	0.12%	0.08%	5
	500	0.54%	0.52%	0.50%	0.34%	5
Pushdo	50	0.17%	0.13%	0.10%	0.05%	8
	100	0.23%	0.20%	0.17%	0.08%	6
	500	0.72%	0.69%	0.66%	0.45%	6
Rustock	50	0.19%	0.12%	0.06%	0.05%	9
	100	0.28%	0.22%	0.15%	0.08%	9
	500	1.01%	0.95%	0.88%	0.40%	9
Srizbi	50	0.22%	0.11%	0%	0%	11
	100	0.33%	0.22%	0.11%	0%	11
	500	1.21%	1.10%	1.05%	0.79%	11

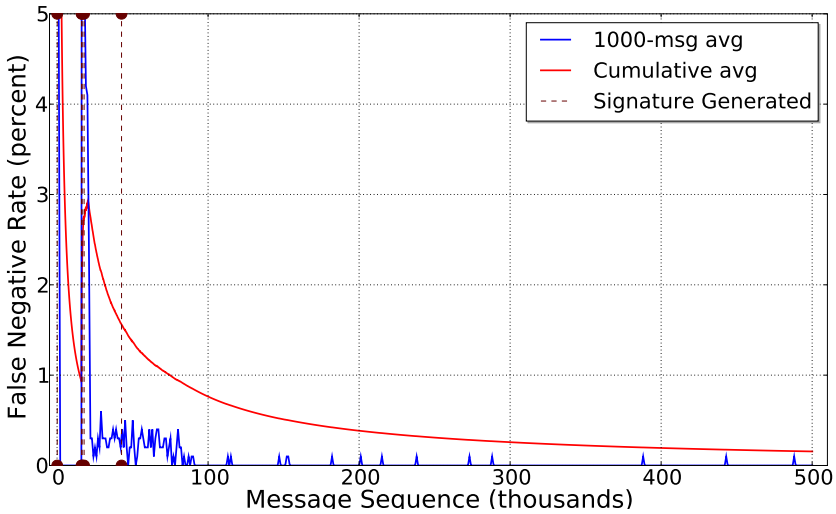
Methodology

Our spam corpus consists of bot-generated spam collected by the Botlab [31] project from the University of Washington. One instance each of the Mega-D, Pushdo, Rustock, and Srizbi bots was executed and their output collected. We split the first 1 million messages from each bot into a training and testing set by sequentially assigning messages to each set in alternation. The Judo system was then used to create signatures from the training data. In parallel with running the Judo system, we processed the testing corpus in chronological order, classifying each message using signatures generated up to that point. In other words, we consider a test message *matched* (a *true positive*) if it matches some signature generated chronologically *before* the test message; otherwise we count it as a *false negative*. Our measure of effectiveness is the false negative rate over the testing message set.

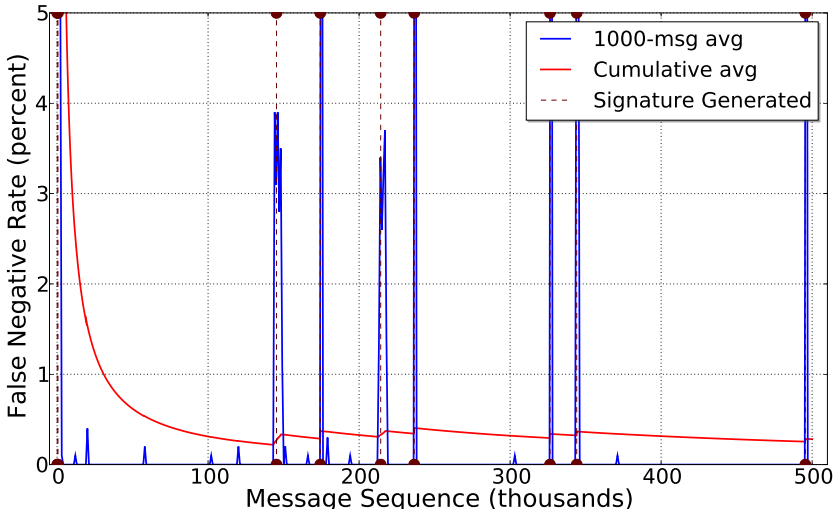
It is important to note that in practice one could employ some *delay* when matching messages against the filters: either holding up messages for a short period to wait for the generation of updated filters, or by retroactively testing messages already accepted, but not yet presented to the receiving user, against any filter updates. To simulate such a scenario, we buffered testing messages in a *classification buffer*, allowing us to delay classification. We denote the length of the classification buffer by d. The case $d = 0$ corresponds to no message buffering; in other words, messages must be classified immediately upon being received. We also evaluated signature performance with the classification delay d set to 50, 100 and 500. In a real-world deployment, we can think of this buffer as a very short delay introduced by e-mail providers before delivering incoming e-mails to inboxes.

Results

Our results confirm the effectiveness of the Judo system in this "live" setting as well. Table 5.3 shows the cumulative results for each combination of k and d, as well as the number of signatures generated for each botnet during the experiment. Two trends are evident. First, the false negative rate decreases as the classification delay d increases. This is not surprising, since the delay gives Judo time to build

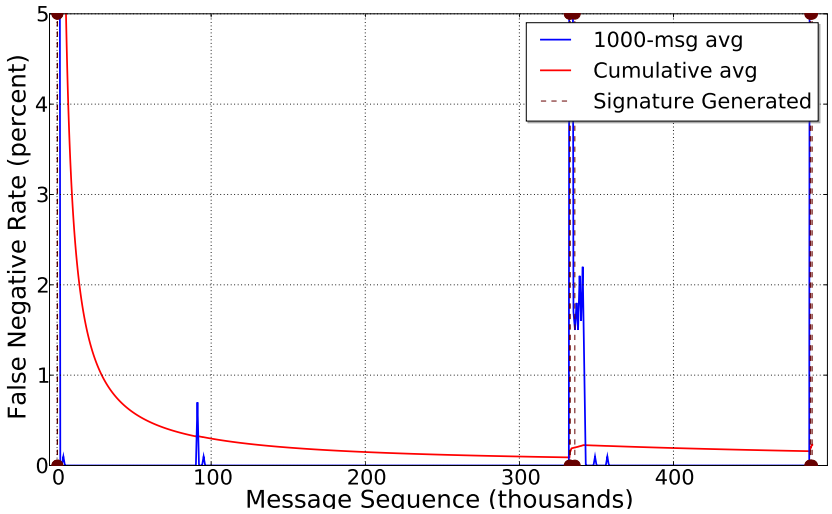


(a) Mega-D

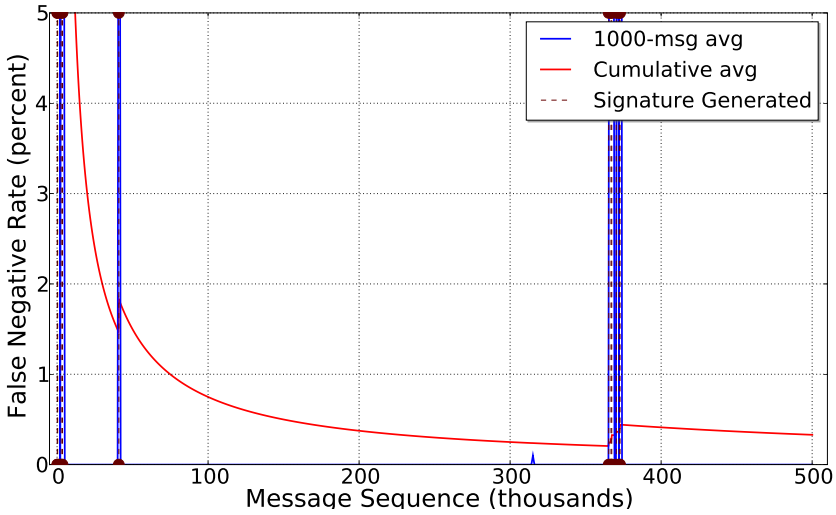


(b) Rustock

Figure 5.5: Classification effectiveness on Mega-D and Rustock spam generated by a single bot, as a function of the testing message sequence. Experiment parameters: $k = 100$, $d = 0$ (that is, 100 training messages to generate each new signature, and immediate classification of test messages rather than post facto).



(a) Pushdo



(b) Srizbi

Figure 5.6: Classification effectiveness on Pushdo, and Srizbi spam generated by a single bot, as a function of the testing message sequence. Experiment parameters: $k = 100$, $d = 0$ (that is, 100 training messages to generate each new signature, and immediate classification of test messages rather than post facto).

a signature. The second trend, an *increasing* false negative rate as k increases, may seem counterintuitive because in our previous experiment, increasing k led to a decrease in the false negative rate. This increase occurs because all spam in the testing set generated before Judo produces the signature is counted as a false negative. Classification delay helps, but even with $d = 500$, a new signature is not produced until we collect 500 messages *that match no other signature*.

Dynamic Behavior. We can better understand Judo by looking at its dynamic behavior. Figure 5.5 and Figure 5.6 show the average and cumulative false negative rate as a function of the testing messages. Dashed vertical lines indicate when Judo generated a new signature. Looking at the Mega-D plot (Figure 5.5a), we see that Judo is generating signatures during the first 20,000 messages in the testing set. After the initial flurry of signature generation, the false negative rate hovers just under 0.5%. After 100,000 testing messages, the false negative rate drops to nearly zero as the signatures are refined.

There are also two interesting observations here. First, peaks sometimes disappear from the graphs without the creation of a new signature. Normally we would expect that mismatches would be eliminated by inferring a new, previously missed underlying template. The effect in question here, though, is a result of using the second chance mechanism. For example, missing entries from a dictionary node can cause some false negative hits to occur until eventually the signature gets updated. This is done incrementally without the need to deploy a new signature, hence the absence of a dashed vertical line in the graph. The second observation is similar and relates to the creation of new signatures without any peaks appearing in the graph indicating the need for performing such an action. In this case, we need to remember that the training and testing track operate independently of each other. Thus it is sometimes the case that the training track observes a new template slightly before the testing track, and of course immediately generates a new signature. In this way, false negative hits are eliminated since the signature is already available for use when messages from the new template appear on the testing track.

We also observe that Pushdo exhibits different behavior in terms of the number of generated signatures. One would expect that the number of such signatures should be the same regardless of the parameters used. Although this holds for the other botnets, it does not for Pushdo due to the dictionary statistical test. Recall from Section 5.3 that we declare something as a dictionary only if Judo believes that it has seen every entry of it. This decision is based on the occurrences of the least-frequently observed element in the set under question. Hence, in the cases where we observe the same elements repeated over an extended number of messages, we can sometimes mis-conclude that we have seen the dictionary in its entirety. The use of a high threshold ensures that we keep such cases to a minimum. While processing Pushdo, the algorithm mistakenly classified a node as a dictionary before capturing all of its entries. As a result, Judo eventually generated multiple regular expressions for the same underlying template, with each one including a different subset of the underlying dictionaries.

5.4.4 Real-world Deployment

The previous experiments tested regular expressions produced by the template inference system against spam produced by a single bot instance. Doing so illuminates how quickly and how well the system learns a new template, but does not fully match how we would operationally deploy such filtering. We finish our evaluation with an assessment using *multiple* bot instances, one to generate the training data and the others to generate the test data. This configuration tells us the degree to which signatures built using one bot's spam are useful in filtering spam from multiple other instances. It also tests to a certain degree our assumption regarding the small number of templates actively used by botnets.

Methodology

We ran two instances of Xarvester and two of Mega-D in a contained environment akin to Botlab [31]. One of the bots was arbitrarily selected to provide the training corpus and the other the testing corpus. We also ran four instances of Rustock and six instances of Gheg. In a similar manner, one of the bots was

Table 5.4: Number of training and testing messages used in the real-world deployment experiment.

Bots	Training	Testing
Xarvester	184,948	178,944
Mega-D	174,772	171,877
Gheg	48,415	207,207
Rustock	252,474	680,000

Table 5.5: Cumulative false negative rate as a function of training buffer size k and classification delay d for spam generated by a multiple bot instances, one generating the training spam and the others the testing spam. The "Sig" column shows the number of signatures generated during the experiment. Signatures generated in this experiment produced no false positives on our corpora.

Cumulative False Negative Rate						
Botnet	k \ d	0	50	100	500	Sig
Xarvester	50	0.07%	0.04%	0.02%	0%	6
	100	0.13%	0.06%	0.03%	0%	6
	500	1.00%	0.89%	0.78%	0.02%	6
Mega-D	50	0.09%	0.06%	0.03%	0%	1
	100	0.13%	0.10%	0.07%	0%	1
	500	0.92%	0.90%	0.87%	0.64%	1
Gheg	50	0.88%	0.86%	0.84%	0.64%	3
	100	1.13%	1.11%	1.08%	0.89%	3
	500	3.56%	3.54%	3.51%	3.33%	3
Rustock	50	0.99%	0.97%	0.95%	0.75%	6
	100	1.03%	1.01%	0.98%	0.78%	6
	500	1.49%	1.47%	1.44%	1.20%	6

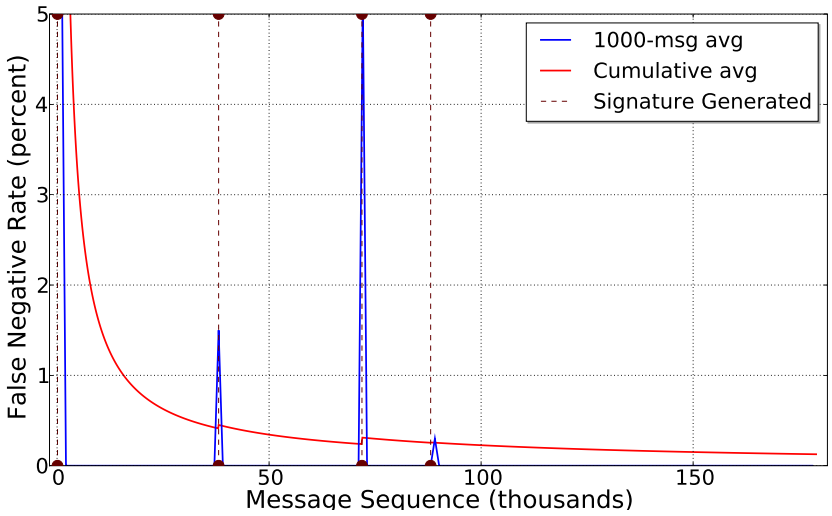
arbitrarily selected to provide the training message set, and the remaining bots, combined, the testing set. Table 5.4 shows the number of messages generated by each bot.

As in the previous experiment, we "played back" both message streams chronologically, using the training data to generate a set of signatures incrementally as described in Section 5.3.3. Again, our metric of effectiveness is the false negative rate on the testing set. To maintain the accuracy of the results we preserved the chronological order of messages in the testing track. This ordering was a consideration for both Rustock and Gheg where we merged the output of multiple bots, as described earlier.

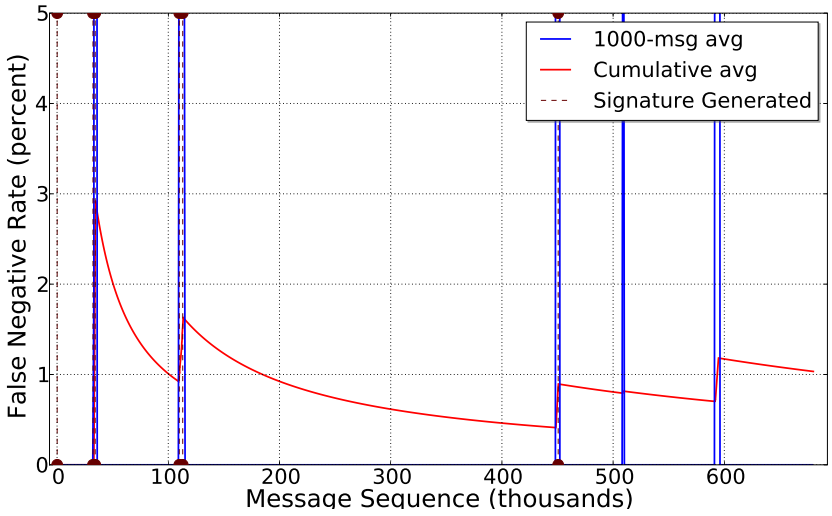
Results

Our results show Judo performed extremely well in this experiment, achieving false negative rates under 1% in most cases and generating no false positives. Table 5.5 shows the cumulative results for each combination of k and d, as well as the number of signatures generated for each botnet during the experiment. Although in all cases the training track was selected arbitrarily, in the case of Gheg we executed the experiment six times. Each time we used a different bot as the training track and the results show the worst false negative rate over these six choices.

Figure 5.7 shows the dynamic behavior of the Xarvester and Rustock bots in this experiment. Despite the fact of now using independent bots as the training and testing tracks, we see that the behavior is quite similar to the previous experiment, where only one bot was used. However, we observe slightly higher false negative rates in the cases where the training track consists of multiple bots. The reason for this higher rate is that the bots are not completely "synchronized", i.e., they do not switch to a new template or dictionary at the exact same time. What is important to note is that in all cases, even after a slight delay, such a switch does indeed occur across all hosts. Gheg exhibited similar behavior when we ran six different instances of the botnet. Recall that for our evaluation, we run the experiment six times and evaluated each one of the bots as being the provider of



(a) Xarvester



(b) Rustock

Figure 5.7: Classification effectiveness on Xarvester and Rustock spam with multiple bots: one bot was used to generate training data for the Judo system and the remaining bots to generate the testing data (1 other for Xarvester, 3 others for Rustock). Experiment parameters: $k = 100$, $d = 0$ (that is, 100 training messages to generate each new signature, and immediate classification of test messages rather than post facto).

the training set. Even when taking into consideration the results from the worst run, as presented in Table 5.5, we can still see that monitoring just a single bot suffices for capturing the output of multiple other spamming hosts. Ultimately, the only reason for the differences in these executions was the precise ordering of the messages, and how early Judo was able to deploy a new signature each time. Our real-world experience verifies to a certain extent our original assumption that spam campaigns use only a small number of templates at any point in time in current practice. Of course, spammers could modify their behavior in response; we discuss this issue further in Section 5.5.

5.4.5 False Positives

One of the most important features of Judo is the unusual safety that the generated signatures offer. When dealing with spam filtering, the biggest concern has always been falsely identifying legitimate messages as spam. Messages that fall under this category are known as false positives. In this section, we try to verify our claims and validate the safety of the signatures.

There are two main factors that affect the false positive rate of our system. The first is the fact that the generated signatures include information for both the headers and the body of the messages. In contrast to more naive methods of simply using URLs or subject lines for identifying spam e-mails, we use the additional information for minimizing the possibility of accidental matches. For the purpose of the current evaluation though, every header besides "Subject" was removed from the signatures. We made this choice for two reasons. First, we want to examine the system under a worst-case scenario, since it is straightforward to see that the presence of additional headers can only improve our false positive rates. Second, we want to remove any possible temporal bias that would pollute our results. Such bias might be the result of age-sensitive headers like "User Agent".

The second factor which contributes to the system's strong false positive results is the existence of anchor and dictionary nodes. Recall that dictionary nodes are only created when Judo estimates that it has observed every single dictionary entry. As described in Section 5.3, these nodes impose strong limitations as to

what messages a signature can match. Hence we add the constraint that all final signatures must contain at least one anchor or dictionary node. If this constraint is violated, we consider the signature *unsafe* and discard it. Although this heuristic can potentially hinder the false negative rates, it also makes sure that false positives remain very small. We confirmed that all signatures used in our evaluation were *safe*. There was only one case where it became necessary to discard signatures, as described in Section 5.4.2.

We first look at the Storm templates. Based on our dataset description in Section 5.4.2, we split this analysis into URL and non-URL (stock) templates. For the former category, which included signatures from the self-propagation and pharmaceutical templates, we had no false positives in three of the four legitimate mail corpora. In the lists.gnu.org corpus, signatures produced from 100 or fewer training messages resulted in a false positive rate of 1 in 50,000. This rate arose from a small number of cases in which dictionaries were not constructed until $k = 500$. For the remaining values of k the result was again zero matches.

Storm templates that did not contain a URL proved to be a harder workload for our system, and the only scenario where *unsafe* signatures were generated and discarded. Although URLs are not a requirement for producing good signatures, the problem was amplified in this case due to the very small length of messages generated by the Storm botnet. Hence it is sometimes the case that the system cannot obtain enough information for smaller numbers of training messages. This issue, though, is eliminated when moving to higher values of k.

For these stock templates, the 99th percentile false positive rate for $k \leq 100$ was under 0.1% across all templates, and with a maximum false positive rate of 0.4% at $k = 10$. For $k \geq 500$, the maximum false positive rate was 1 in 50,000 on the Enron corpus, and zero for the remaining three corpora. We emphasize again that we are using stripped down versions of the signatures (subject and body patterns only); including additional headers ("MIME-Version" and "Content-Transfer-Encoding") eliminated all false positives. We further validated these numbers by making sure that these additional headers were indeed included in the messages of our legitimate mail corpora. Hence we confirmed that all mis-

matches arose due to the corresponding header regular expressions failing to match, and not due to bias of our specific dataset.

The Botlab workload (Section 5.4.3) produced zero matches against all corpora for the MegaD, Pushdo and Srizbi botnets. The only exception was the signatures generated for Rustock. We had zero matches against the TREC 2007 corpus. When testing against the remaining corpora, signatures for this botnet produced an average false positive rate between 0.00021% to 0.01%. The 99th percentile false positive rate was at most 0.24% and 95th percentile at most 0.04%. When looking into this issue further we identified the source of the problem as the inability of these signatures to produce all possible dictionaries. One reason was the very high threshold used for allowing the conversion of a character class to a dictionary node. We are currently looking into this particular problem for further improvement. Once again, though, we note that incorporating additional headers gives a worst-case false positive rate of 1 in 12,500 due to signature mismatches and not because of the absence of these headers in our corpora.

For all other signatures generated from messages captured in our sandbox environment, the result was zero false positive matches across all corpora for all botnets. Note that these results correspond to our real-world deployment experiment, with signatures being generated for the very latest spam messages sent by the botnets. The structure of these messages allowed for the creation of very precise regular expressions, such as the example presented in Figure 5.8 for the MegaD botnet.

5.4.6 Response Time

One reasonable concern can be the time Judo requires for generating a new signature. Since we aim to shrink the window between the time a new campaign starts and the time we deploy a filter, being able to quickly infer the underlying template is crucial. As already shown in Section 5.3.4, execution time is not a concern for the system, as it takes under 10 seconds in almost all cases to run the algorithm. Thus the only question left to answer is how long it takes to build up the required training sets.

```
Subject ^(RE: Message|new mail|Return mail|Return Mail|Re: Order ▷
status|no-reply|Your order|Delivery Status Notification| ▷
Delivery Status Notification \(Failure\))$

^<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD>
<META http-equiv=Content-Type content="text/html; charset=(us-ascii| ▷
iso-8859-2|iso-8859-1|windows-1250|Windows-1252)">
</HEAD>
<BODY><a href="http://(talklucid|samefield|famousfall|bluedoes| ▷
meekclaim).com/" target="_blank">
<img src="http://(talklucid|samefield|famousfall|bluedoes| ▷
meekclaim).com/dyuwqlk.jpg" border="0" alt="Show picture and go to ▷
site now!"></a></BODY></HTML>$
```

Figure 5.8: Fragment of a template generated for the Mega-D botnet on August 26, 2009. Only the subject and body are displayed with full dictionaries, exactly as they were captured. Recall that templates are inferred from only the output of bots, without any access to the C&C channel and without any information regarding the underlying mechanisms used.

Obviously such a metric is dependent on the spamming rate of botnets. It is also possible that messages from multiple campaigns might be interleaved in a corpus, further complicating the exact time it takes to create the training set for each one. Despite this, knowing the spamming rate at which bots operate can give us a good high-level estimate of the time requirements imposed by Judo. Focusing on the datasets used in our real-world deployment scenario, we identified that the six Gheg bots took less than 6 hours to send all 255,622 e-mails. This translates to about 118 messages per minute, for each bot. In a similar way, the four Rustock bots we had deployed required only 20 hours to send 932,474 messages, which gives us a spamming rate of 194 messages per minute, for each bot. Considering that for modern botnets, Judo showed excellent performance for any values of $k \leq 500$, we conclude that the system requires only a few minutes for the deployment of

a new signature. In fact, the time it takes to do so directly affects all results of our evaluation in Section 5.4.3 and Section 5.4.4. Recall that while the system is creating the required training set for a campaign, any mismatches on the testing tracks are registered as false negatives. Despite this fact, we can see that Judo is still able to maintain excellent performance, hence satisfying our initial goal of a fast response time.

5.4.7 Other Content-Based Approaches

The efficacy of Judo rests on two essential characteristics: its unique vantage point at the *source* of the spam, and the template inference algorithm for creating signatures from this spam. Regarding the latter, one may naturally ask if a *simpler* mechanism suffices. In short, we believe the answer is "No." To our knowledge two such simple approaches have been advanced: subject-line blacklisting and URL domain blacklisting. We consider these in turn.

Subject-line Blacklisting. Filtering based on message subject is one of the earliest spam filtering mechanisms in deployment. Most recently, it was used by the Botlab project [31] to attribute spam to specific botnets.[5] Unfortunately, it is very easy (and in some cases desirable for the spammer) to use subject lines appearing in legitimate mail. One of the Mega-D templates found in our previous experiment used a subject-line dictionary containing "RE: Message," "Re: Order status,", "Return mail," and so on, and can be seen in Figure 5.8. Such a template cannot be effectively characterized using the message subject alone.

URL Domain Blacklisting. URL domain blacklists (e.g. [3, 4]) are lists of domains appearing in spammed URLs. In our experience, domain names appearing in spam do indeed provide a strong signal. However, there are at least two cases where domain names *alone* are not sufficient. The first case, spam not containing URLs (stock spam for example), simply cannot be filtered using URL domain signatures. (Section 5.4.2 shows that Judo is effective against this type of spam).

[5]Note that John *et al.* do not suggest that subject lines alone should be used for *identifying* spam, but for "classifying spam messages as being sent by a particular botnet" after being classified as spam.

The second type of spam uses "laundered" domains, that is, reputable services which are used to redirect to the advertised site. Most recently, for example, spam sent by the Gheg botnet was using groups.yahoo.com and google.com domains in URLs. We suspect this trend will continue as URL domain blacklisting becomes more widely adopted.

"Focused" Bayesian Signatures. An intriguing possibility is using a Bayesian classifier to train on a single campaign or even the output of a single bot, rather than a large universal corpus of spam as is conventional. Our cursory evaluation using SpamAssassin's [78] Bayesian classifier showed promise; however, in addition to a formal evaluation, a number of technical issues still need to be addressed (the larger size of Bayesian signatures, for example).

We also trained SpamAssassin's Bayesian filter on a generic spam corpus of over one thousand recent messages along with the SpamAssassin 2003 "ham" corpus. It fared poorly: from a sample of 5,180 messages from the Waledac botnet, 96% received a score of 0 (meaning "not spam") due to the complete absence of tokens seen in the generic spam corpus; none were given a score above 50 out of 100.

Enterprise Spam Filtering Appliances. We ran a subset of the spam corpora from the Gheg, MegaD, Rustock, and Waledac botnets through a major spam filtering appliance deployed on our university network. Spam from the Waledac and Rustock botnets was completely filtered based upon the URLs appearing in the message bodies, and Mega-D spam was correctly filtered via a generic "pharmaceutical spam" rule. However, only 7.5% of the spam from the Gheg botnet was correctly identified as such; these messages were German language pharmaceutical advertisements laundering URL reputation through google.com via its RSS reader application.

5.5 Discussion

There are four questions invariably asked about any new anti-spam system: how well does it filter spam, how often does it misclassify good e-mail in turn, how

easy or expensive is it to deploy and how will the spammers defeat it? We discuss each of these points briefly here.

As we have seen, template inference can be highly effective in producing filters that precisely match spam from a given botnet. Even in our preliminary prototype we have been able to produce filters that are effectively perfect for individual campaigns after only 1,000 samples. To a certain extent this result is unsurprising: if our underlying assumptions hold, then we will quickly learn the regular language describing the template. Even in less than ideal circumstances we produce filters that are very good at matching subsequent spam. The catch, of course, is that each of our filters is over-constrained to only match the spam arising from one particular botnet and thus they will be completely ineffective against any other spam.

The hidden benefit of this seeming drawback is that filters arising from template inference are unusually *safe*. Their high degree of specialization makes them extremely unlikely to match any legitimate mail and thus false positive rates are typically zero or extremely close thereto. To further validate this hypothesis, we provided the regular expressions corresponding to the data for Xarvester to a leading commercial provider of enterprise anti-spam appliances. They evaluated these filters against their own "ham" corpus and found no matches. Given this evidence, together with our own results, we argue that template inference can be safely used as a pre-filter on any subsequent anti-spam algorithm and will generally only improve its overall accuracy.

There are three aspects to the "cost" of deploying a system such as ours. The first is the complexity of capturing, executing and monitoring the output of spam bots. As more bot instances can be maintained in a contained environment, new filters can be generated more quickly. While this is by no means trivial, it is routinely done in both academia and industry and there is a broad base of tools and technology being developed to support this activity. The second issue concerns the efficiency of the template inference process itself. Here we believe the concern is moot since the algorithm is linear time and our untuned template extraction algorithm is able to generate regular expressions from 1000 messages in

under 10 seconds, and update the expression in 50-100 ms. Next, there is the issue of integration complexity since it is challenging to mandate the creation of new software systems and interfaces. However, since our approach generates standard regular expressions—already in common use in virtually all anti-spam systems—the integration cost should be minimal in practice.

Finally, we recognize that spam is fundamentally an adversarial activity, and successful deployment of our system would force spammers to react in turn to evade it. We consider the likely path of such evolution here. There are three obvious ways that spammers might attempt to stymie the template inference approach.

First, they can use technical means to complicate the execution of bots within controlled environments. A number of bots already implement extensive anti-analysis actions such as the detection of virtual machine environments and the specialization of bot instances to individual hosts (to complicate the sharing of malware samples). Moreover, some botnets require positive proof of a bot's ability to send external spam e-mail before providing spam template data. While this aspect of the botnet arms race seems likely to continue, it also constitutes the weakest *technical* strategy against template inference since there is no fundamental test to distinguish a host whose activity is monitored from one whose is not.

A more daunting countermeasure would be the adoption of more complex spam generation languages. For example, multi-pass directives (e.g., shuffling word order after the initial body is generated) could easily confound the algorithm we have described. While there is no doubt that our inference approach could be improved in turn, for complex languages the general learning problem is untenable. However, there are drawbacks in pursuing such complexity for spammers as well. Template languages emerged slightly over 5 years ago as a way to bypass distributed spam hash databases [74] and they have not changed significantly over that time. Part of the reason is that they are easy for spammers to use and reason about; a new spam campaign does not require significant testing and analysis. However, a more important reason is that there are limits to how much polymorphism can be encoded effectively in a spam message while still preserving

the underlying goal. To be effective, pitches and subject lines must be roughly grammatical, URLs must be properly specified, and so on. Randomizing the letters across such words would defeat template inference but also would likely reduce the underlying conversion rate significantly.

Finally, spammers might manage the distribution of templates in a more adversarial fashion. In particular, were each bot instance given templates with unique features then the regular expressions learned from the output of one bot would suffer from overtraining; they would be unable to generalize to spam issued from another bot in the same botnet. Depending precisely on how such features were generated, this could add significant complexity to the underlying inference problem at relatively low cost to spammers, and without significantly changing the overall "look and feel" of such messages to potential customers. We leave the challenge of joint learning across bot instances to future work should the spam ecosystem evolve in this manner.

5.6 Summary

In starting this chapter we observed that strong defenses benefit from obtaining current and high quality intelligence. This point is hardly lost on the anti-spam community and over time there have been many efforts to share information among sites, precisely to shrink the window of vulnerability between when a new kind of spam appears and a corresponding e-mail filter is installed. Historically, these efforts have been successful when the information gathering itself can be centralized and have floundered when they require bilateral sharing of mail samples (even in a statistical sense). Thus, IP-based blacklists constitute intelligence that, upon being learned, is shared quickly and widely, while content-based filter rules continue to be learned independently by each defender.

To put it another way, the receiver-oriented learning approach makes it challenging to automatically share new spam intelligence (for reasons of privacy, logistics, scale, etc.). However, given that a small number of botnets generate most spam today, this problem can be neatly sidestepped. We have shown that it

is practical to generate high-quality spam content signatures simply by observing the output of bot instances and inferring the likely content of their underlying template. Moreover, this approach is particularly attractive since the resulting regular expressions are highly specialized and thus produce virtually no false positives. Finally, while we recognize that there are a range of countermeasures that an adversary might take in response, we argue that they are not trivial for the attacker and thus that the template inference approach is likely to have value for at least a modest period of time.

Chapter 5, in part, is a reprint of the material as it appears in Proceedings of the Network and Distributed System Security Symposium 2010. Pitsillidis, Andreas; Levchenko, Kirill; Kreibich, Christian; Kanich, Chris; Voelker, Geoffrey M.; Paxson, Vern; Weaver, Nicholas; Savage, Stefan. The dissertation author was the primary investigator and author of this paper.

Chapter 6

Click Trajectories: End-to-End Analysis of the Spam Value Chain

We intuitively tend to only think about the e-mail sending aspect of the spam problem. As we have emphasized numerous times throughout this dissertation though, spam e-mails are primarily used as an advertising medium. As in any advertising business, generating revenue is an essential requirement for the viability of the whole operation. In this chapter we focus on the *click support* component of the spam value chain, which we define as the mechanism responsible for monetizing spam-advertised URLs.

6.1 Introduction

We may think of e-mail spam as a scourge—jamming our collective inboxes with tens of billions of unwanted messages each day—but to its perpetrators it is a potent marketing channel that taps latent demand for a variety of products and services. While most attention focuses on the problem of spam *delivery*, the e-mail vector itself comprises only the *visible* portion of a large, multi-faceted business enterprise. Each click on a spam-advertised link is in fact just the start of a long and complex trajectory, spanning a range of both technical and business components that together provide the necessary infrastructure needed to monetize a customer's visit. Botnet services must be secured, domains registered, name

servers provisioned, and hosting or proxy services acquired. All of these, in addition to payment processing, merchant bank accounts, customer service, and fulfillment, reflect necessary elements in the spam value chain.

While elements of this chain have received study in isolation (e.g., dynamics of botnets [31], DNS fast-flux networks [26, 69], Web site hosting [6, 35]), the relationship between them is far less well understood. Yet it is these very relationships that capture the structural dependencies—and hence the potential *weaknesses*—within the spam ecosystem's business processes. Indeed, each distinct *path* through this chain—registrar, name server, hosting, affiliate program, payment processing, fulfillment—directly reflects an "entrepreneurial activity" by which the perpetrators muster capital investments and business relationships to create value. Today we lack insight into even the most basic characteristics of this activity. How many organizations are complicit in the spam ecosystem? Which points in their value chains do they share and which operate independently? How "wide" is the bottleneck at each stage of the value chain—do miscreants find alternatives plentiful and cheap, or scarce, requiring careful husbanding?

The desire to address these kinds of questions empirically—and thus guide decisions about the most effective mechanisms for addressing the spam problem—forms the core motivation of our work. In this chapter we focus on the end-to-end resource dependencies ("Click Trajectories") behind individual spam campaigns and then analyze the relationships among them, using the data presented in Chapter 3. We characterize the resource footprint at each step in the spam value chain, the extent of sharing between spam organizations and, most importantly, the relative prospects for interrupting spam monetization at different stages of the process.

6.2 Analysis

A major goal of this dissertation is to identify any "bottlenecks" in the spam value chain: opportunities for disrupting monetization at a stage where the fewest alternatives are available to spammers (and ideally for which switching cost is high as well). Thus, in this chapter we focus directly on analyzing the degree

to which affiliate programs *share* infrastructure, by considering the *click support* (i.e., domain registration, name service and Web hosting service) phase of the spam value chain. We explore all components involved and consider the potential effectiveness of interventions for each.

6.2.1 Click Support

As described in Chapter 3 we crawl a broad range of domains—covering the domains found in over 98% of our spam feed URLs—and use clustering and tagging to associate the resulting Web sites with particular affiliate programs. This data, in combination with our DNS crawler and domain WHOIS data, allows us to associate each such domain with an affiliate program and its various click support resources (registrar, set of name server IP addresses and set of Web hosting IP addresses). However, before we proceed with our analysis, we first highlight the subtleties that result from the use of Web site *redirection*.

Redirection

As we mentioned, some Web sites will redirect the visitor from the initial domain found in a spam message to one or more additional sites, ultimately resolving the final Web page (we call the domain for this page the "final domain"). Thus, for such cases one could choose to measure the infrastructure around the "initial domains" or the "final domains".

To explain further, 32% of crawled URLs in our data redirected at least once and of such URLs, roughly 6% did so through public URL shorteners (e.g., `bit.ly`), 9% through well-known "free hosting" services (e.g., `angelfire.com`), and 40% were to a URL ending in `.html` (typically indicating a redirect page installed on a compromised Web server). In our data, we identified over 130 shortener services in use, over 160 free hosting services and over 8,000 likely-compromised Web servers. Of the remainder, the other common pattern is the use of low-quality "throw away" domains, the idea being to advertise a new set of domains, typically registered using random letters or combinations of words, whenever the previous set's traffic-drawing potential is reduced due to blacklisting [38].

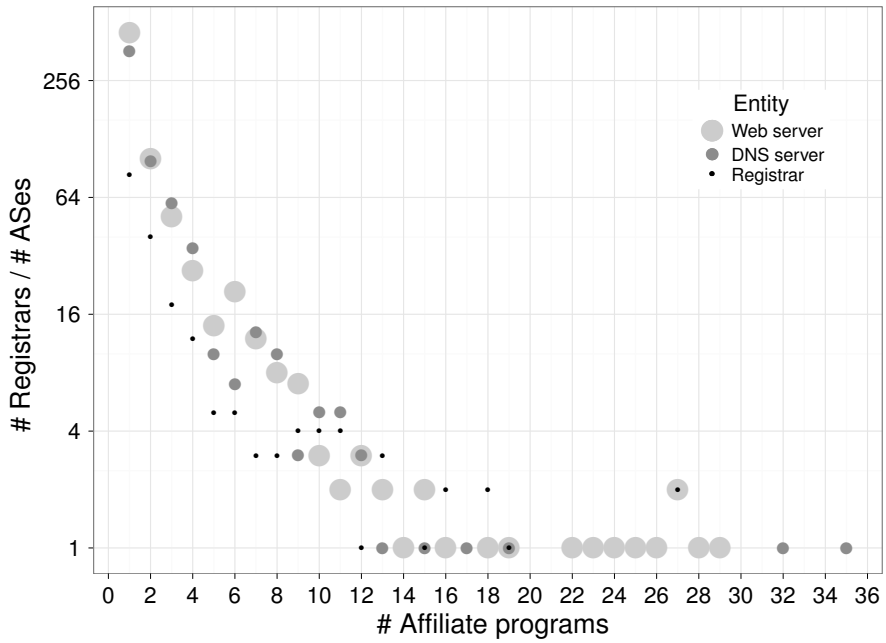


Figure 6.1: Sharing of network infrastructure among affiliate programs. Only a small number of registrars host domains for many affiliate programs, and similarly only a small number of ASes host name and Web servers for many programs. (Note y-axis is log scale.)

Given this, we choose to focus entirely on the final domains precisely because these represent the more valuable infrastructure most clearly operated by an affiliate.

Returning to our key question, we next examine the set of resources used by sites for each affiliate program. In particular, we consider this data in terms of the service organization who is responsible for the resource and how many affiliate programs make use of their service.

Network infrastructure sharing

A spam-advertised site typically has a domain name that must be resolved to access the site.[1] This name must in turn be allocated via a registrar, who has the authority to shutdown or even take back a domain in the event of abuse [48]. In addition, to resolve and access each site, spammers must also provision servers

[1]Fewer than half a percent use raw IP addresses in our study.

to provide DNS and Web services. These servers receive network access from individual ISPs who have the authority to disconnect clients who violate terms of service policies or in response to complaints.

Figure 6.1 shows that network infrastructure sharing among affiliate programs—when it occurs—is concentrated in a small number of registrars and Autonomous Systems (ASes). We use the AS number as a proxy for ISP. Many registrars and ASes host infrastructure for just one or two affiliate programs, only a small number host infrastructure for many affiliate programs, and no single registrar or AS hosts infrastructure for a substantial fraction of the programs overall. (As we will see in Section 6.2.2 however, this situation can change drastically when we weight by the volume of spam advertising each domain.) Specifically, Figure 6.1 shows the number of registrars (y-axis) that serve registered domains for a given number of affiliate programs (x-axis). Over 80 registrars, for instance, serve domains for a single affiliate program, while just two registrars (NauNet and China Springboard) serve domains for over 20 programs. For name servers and Web servers, it shows the number of ASes hosting servers for a given number of affiliate programs. Over 350 and 450 ASes host DNS and Web servers, respectively, for a single affiliate program; yet, just two and nine ASes host DNS and Web servers, respectively, for over 20 programs (including Hanaro Telecom, China Communication, and ChinaNet).

Although most registrars and ASes host infrastructure for just one affiliate program, each program could still engage many such registrars to serve their domains and many such ASes to host their DNS and Web servers. Figure 6.2 shows, though, that programs do not in general distribute their infrastructure across a large set of registrars or ASes: for most programs, each of them uses only a small fraction of registrars and ASes found in our data set. Specifically, Figure 6.2 shows the cumulative distribution of the fraction of registrars and ASes in our data set used by affiliate programs. For 50% of the affiliate programs, their domains, name servers, and Web servers are distributed over just 8% or fewer of the registrars and ASes, respectively; and 80% of the affiliate programs have their infrastructure distributed over 20% or fewer of the registrars and ASes. Only a handful of programs,

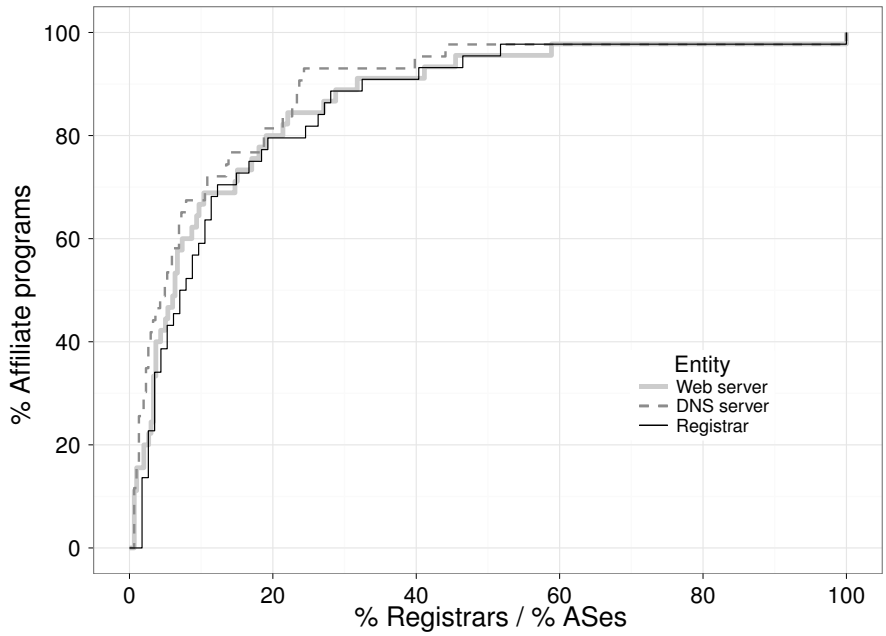


Figure 6.2: Distribution of infrastructure among affiliate programs. Only a small percentage of programs distribute their registered domain, name server, and Web server infrastructure among many registrars and ASes, respectively.

such as EvaPharmacy, Pharmacy Express, and RX Partners, have infrastructure distributed over a large percentage (50% or more) of registrars and ASes.

To summarize, there are a broad range of registrars and ISPs who are used to support spam-advertised sites, but there is only limited amounts of organized sharing and different programs appear to use different subsets of available resource providers. We did find *some* evidence of clear inter-program sharing in the form of several large groups of DNS servers willing to authoritatively resolve collections of EvaPharmacy, Mailien and OEM Soft Store domains for which they were outside the DNS hierarchy (i.e., the name servers were never referred by the TLD). This overlap could reflect a particular affiliate advertising for multiple distinct programs and sharing resources internally or it could represent a shared service provider used by distinct affiliates.

6.2.2 Intervention analysis

Finally, we now reconsider these different resources in the spam monetization pipeline, but this time explicitly from the standpoint of the defender. In particular, for any given registered domain used in spam, the defender may choose to intervene by either blocking its advertising (e.g., filtering spam) or disrupting its click support (e.g., takedowns for name servers of hosting sites). In each case, it is typically possible to employ either a "takedown" approach (removing the resource comprehensively) or cheaper "blacklisting" approach at more limited scope (disallowing access to the resource for a subset of users), but for simplicity we model the interventions in the takedown style. More importantly, we are interested in which of these interventions will have the most *impact*.

Ideally, we believe that such anti-spam interventions need to be evaluated in terms of two factors: their overhead to implement and their business impact on the spam value chain. In turn, this business impact is the sum of both the replacement cost (to acquire new resources equivalent to the ones disrupted) and the opportunity cost (revenue forgone while the resource is being replaced). While, at this point in time, we are unable to precisely quantify all of these values, we believe our data illustrates gross differences in scale that are likely to dominate any remaining factors.

To reason about the effects of these interventions, we consider the registered domains for the affiliate programs and storefront brands in our study and calculate their relative volume in our spam feeds (we particularly subtract the botnet feeds when doing this calculation as their inherent bias would skew the calculation in favor of certain programs). We then calculate the fraction of these domain trajectories that could be completely blocked (if only temporarily) through a given level of intervention at several resource tiers:

Registrar. Here we examine the effect if individual registrars were to suspend their domains which are known to be used in advertising or hosting the sites in our study.

Hosting. We use the same analysis, but instead look at the number of distinct ASs that would need to be contacted (who would then need to agree to

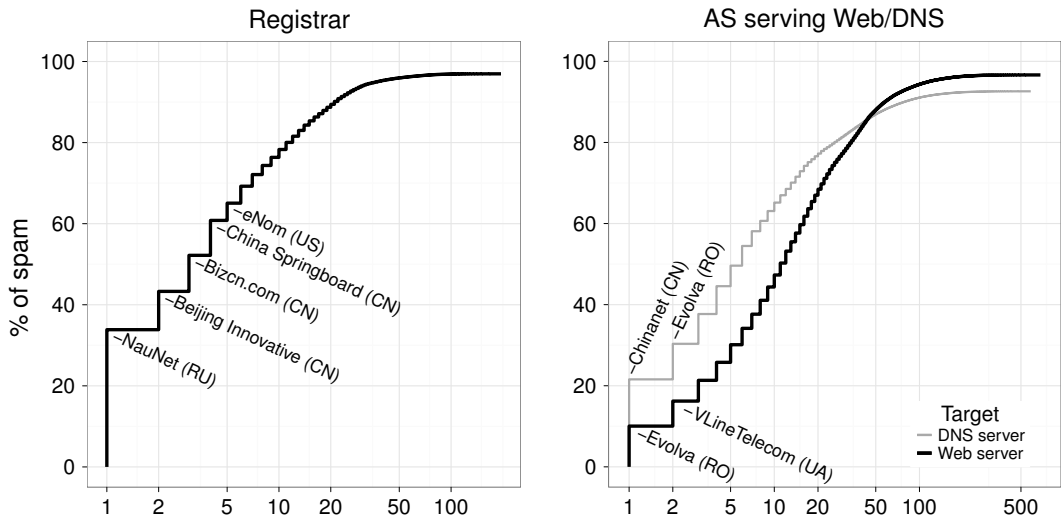


Figure 6.3: Takedown effectiveness when considering domain registrars (left) and DNS/Web hosters (right).

shut down all associated hosts in their address space) in order to interrupt a given volume of spam domain trajectories. We consider both name server and Web hosting, but in each case there may be multiple IP addresses recorded providing service for the domain. We adopt a "worst case" model that *all* such resources must be eliminated (i.e., every IP seen hosting a particular domain) for that domain's trajectory to be disrupted.

Figure 6.3 plots this data as CDFs of the spam volume in our feeds that would be disrupted using these approaches. For both registrars and hosters there are significant concentrations among the top few providers and thus takedowns would seem to be an effective strategy. For example, almost 40% of spam-advertised domains in our feeds were registered by NauNet, while a single Romanian provider, Evolva Telecom, hosts almost 9% of name servers for spam-advertised domains and over 10% of the Web servers hosting their content; in turn, over 60% of these had payments handled via a single acquirer, Azerigazbank.

However, these numbers do not tell the entire story. Another key issue is the availability of alternatives and their switching cost.

For example, while only a small number of individual IP addresses were *used* to support spam-advertised sites, the supply of hosting resources is vast, with thousands of hosting providers and millions of compromised hosts. Note that spam hosting statistics can be heavily impacted by the differences in spam volume produced by different affiliates/spammers. For example, while we find that over 80% of all spam received in this study leads to sites hosted by just 100 distinct IP addresses, there are another 2336 addresses used to host the remaining 20% of spam-advertised sites, many belonging to the same affiliate programs but advertising with lower volumes of spam e-mail. The switching cost is also low and new hosts can be provisioned on demand and for low cost. The cost of compromised proxies is driven by the market price for compromised hosts via Pay-Per-Install enterprises, which today are roughly $200/1000 for Western hosts and $5–10/1000 for Asian hosts [82]. Dedicated bulletproof hosting is more expensive, but we have seen prices as low as $30/month for virtual hosting (up to several hundred dollars for dedicated hosting).

By contrast, the situation with registrars appears more promising. The supply of registrars is smaller (roughly 900 gTLD registrars are accredited by ICANN as of this writing) and there is evidence that not all registrars are equally permissive of spam-based advertising [43]. Moreover, there have also been individual successful efforts to address malicious use of domain names, both by registries (e.g., CNNIC) and when working with individual registrars (e.g., eNom [40]). Unfortunately, these efforts have been slow, ongoing, and fraught with politics since they require global cooperation to be effective (only individual registrars or registries can take these actions). Indeed, a recent study has empirically evaluated the efficacy of *past* registrar-level interventions and found that spammers show great agility in working around such actions [47]. Ultimately, the low cost of a domain name (many can be had for under $1 in bulk) and ease of switching registrars makes such interventions difficult.

6.3 Summary

In this chapter we have described a large-scale empirical study to measure the spam value chain in an end-to-end fashion. We have described a framework for conceptualizing resource requirements for spam monetization and, using this model, we have characterized the use of key infrastructure—registrars and hosting—for a wide array of spam-advertised business interests. Finally, we have used this data to provide a normative analysis of spam intervention approaches and to offer evidence that the potential for impact with defensive interventions at the click support tier of the spam value chain, is limited. This limited impact potential is especially true when we also consider the availability of alternatives at each level, and their switching cost. Coincidentally, the network level is where most of the efforts of the computer security community are focused on nowadays, for disrupting click support. Instead of the current approach, we suggest exploring alternative non-technical means of intervention that can offer a greater potential for success. Along these lines, a follow-up study that builds upon the findings of this dissertation has identified payments to be the weakest link, by far, in the spam value chain [46]. In this work, the authors have demonstrated that the payment tier is the most concentrated and valuable asset in the spam ecosystem, with a potential for a truly effective intervention through public policy action in Western countries. The details of this are outside the scope of our dissertation.

Chapter 6, in part, is a reprint of the material as it appears in Proceedings of the IEEE Symposium on Security and Privacy 2011. Levchenko, Kirill; Pitsillidis, Andreas; Chachra, Neha; Enright, Brandon; Felegyhzi, Mark; Grier, Chris; Halvorson, Tristan; Kanich, Chris; Kreibich, Christian; Liu, He; McCoy, Damon; Weaver, Nicholas; Paxson, Vern; Voelker, Geoffrey M.; Savage, Stefan. The dissertation author was one of the primary investigators and authors of this paper.

Chapter 7

Conclusion

We live in an era where advancements in the field of computer science happen at an extremely rapid pace. On a daily basis we see new technologies, services, and products appearing, all of which introduce a new set of challenges. Unavoidably, computer security has become an integral part of this new era, and the need to quickly respond to new threats is now more important than ever. Due to this rapid pace, efficiency is extremely important nowadays since more often than not, we are faced with the problem of limited resources. This is true even for the largest organizations, thus choosing how to invest those resources more effectively has become a necessity.

At the same time, today we have an unmatched ability to gather, process and analyze data concerning Internet activity. This opens up tremendous opportunities for computer security research, by allowing us to better understand the adversary, the vulnerable users, and the efficacy of our defenses. This is important since as researchers, we often tend to solely focus on the technical challenges of the problem at hand, while ignoring other factors surrounding it. The comprehensive understanding of a problem though, often enables more effective solutions that also take into consideration factors such as the availability of alternatives for attackers and the accompanying costs.

In this dissertation we focus on the spam problem and its surrounding ecosystem. Ultimately, our goal is the disruption of the mechanism responsible for monetizing spam e-mails, which we define as the *spam value chain*. We do so by

first gaining a thorough understanding of all the available data sources which guide our research. We document significant differences across different data feeds by exploring assumptions that are commonly used by the research community, and demonstrate that the differences we find can translate into analysis limitations. We then focus on *advertising* and *click support*, the two primary stages of the spam value chain, and evaluate the potential for intervention at each. First, we target advertising through the development of a spam filtering algorithm that targets botnets, and we demonstrate its effectiveness using live data. Finally, we shift our focus to click support, where we document and quantify the end-to-end resource dependencies in use by spam campaigns today. We characterize the footprint of the different resources involved, analyze the prospects for disrupting spam monetization at the different possible levels, and show that the availability of cheap alternatives for the attackers makes defensive intervention at the network level an enormous challenge. Instead, we propose focusing as a community on seeking alternative non-technical ways for efficiently disrupting click support.

7.1 Future Directions

Given our findings regarding interventions at the click support level of the spam value chain, a first next step is exploring other potential weak points in the chain. We have already shown that targeting the problem at the network level is challenging, and the question is whether we can improve upon this situation. Based on our research, future work has already demonstrated that more efficient intervention mechanisms do exist, which target the payment part of the spam value chain. The first results of this effort are very promising and offer a rare advantage to defenders, in terms of the cost imposed for circumventing this type of protection.

At the same time, our methodology can be applied to evaluating the effectiveness of other defensive mechanisms as well. For example, domain blacklists are one such popular mechanism within the computer security community. Potential use cases include blocking access to spam-advertised domains, or to domains that host malicious content and executables. Although there have been many studies

that have evaluated the effectiveness of these blacklists in terms of their coverage (i.e., how many malicious domains do they block at any given time), we are not aware of any studies that do so in correlation to timing. In particular, we want to explore the effectiveness of these blacklists in combination with the visit patterns that users exhibit in relation to such domains. Is it, for example, the case that user visits drop dramatically enough after a certain time interval, that blacklisting is essentially not needed beyond that time threshold? Today we have the data for exploring this kind of question, and this type of timing analysis based on real-world visit patterns can be extended to other defensive mechanisms as well.

7.2 Final Thoughts

We expect that in the foreseeable future, the spam problem will remain a constant struggle between attackers and defenders. This dissertation has focused on e-mail which is the most popular spam vector today. Eventually, as anti-spam defenses become more sophisticated, it is expected that miscreants will shift their focus to other, more easily targeted vectors. We believe though that our data-driven approach, and the ability to challenge existing assumptions by leveraging the vast amount of data we have available today, can be applied equally successfully to future threats as well. By targeting each problem at its core and focusing on its weak points, we can ensure maximum impact for both current and future defensive mechanisms.

Bibliography

[1] jwSpamSpy Spam Domain Blacklist. http://www.joewein.net/spam/spam-bl.htm.

[2] lists.gnu.org. ftp://lists.gnu.org.

[3] SURBL. http://www.surbl.org.

[4] URIBL – Realtime URI Blacklist. http://www.uribl.com.

[5] Alexa. Alexa top 500 global sites. http://www.alexa.com/topsites, June 2011.

[6] David S. Anderson, Chris Fleizach, Stefan Savage, and Geoffrey M. Voelker. Spamscatter: Characterizing Internet Scam Hosting Infrastructure. In *Proc. of 16th USENIX Security*, 2007.

[7] Ion Androutsopoulos, John Koutsias, Konstantinos Chandrinos, Georgios Paliouras, and Constantine D. Spyropoulos. An Evaluation of Naive Bayesian Anti-Spam Filtering. In *Proc. of 1st MLNIA*, 2000.

[8] Jart Armin, James McQuaid, and Matt Jonkman. Atrivo — Cyber Crime USA. http://fserror.com/pdf/Atrivo.pdf, 2008.

[9] Behind Online Pharma. From Mumbai to Riga to New York: Our Investigative Class Follows the Trail of Illegal Pharma. http://behindonlinepharma.com, 2009.

[10] Robert Beverly and Karen Sollins. Exploiting Transport-Level Characteristics of Spam. In *Proc. of 5th CEAS*, 2008.

[11] Xavier Carreras and Luís Màrquez. Boosting Trees for Anti-Spam Email Filtering. In *Proc. of RANLP-2001*, 2001.

[12] Claude Castelluccia, Mohamed Ali Kaafar, Pere Manils, and Daniele Perito. Geolocalization of Proxied Services and its Application to Fast-Flux Hidden Servers. In *Proc. of 9th IMC*, 2009.

[13] Richard Clayton. How much did shutting down McColo help? In *Proc. of 6th CEAS*, 2009.

[14] Gordon V. Cormack and Thomas R. Lynam. On-line Supervised Spam Filter Evaluation. *ACM Trans. Inf. Syst.*, 25(3), July 2007.

[15] Dancho Danchev's Blog — Mind Streams of Information Security Knowledge. The Avalanche Botnet and the TROYAK-AS Connection. http://ddanchev.blogspot.com/2010/05/avalanche-botnet-and-troyak-as.html, 2010.

[16] Harris Drucker, Donghui Wu, and Vladimir N. Vapnik. Support Vector Machines for Spam Categorization. In *Proc. of IEEE Transactions on Neural Networks*, 1999.

[17] Federal Trade Commission. FTC Shuts Down, Freezes Assets of Vast International Spam E-Mail Network. http://ftc.gov/opa/2008/10/herbalkings.shtm, 2008.

[18] Wu-chang Feng and Ed Kaiser. kaPoW Webmail: Effective Disincentives Against Spam. In *Proc. of 7th CEAS*, 2010.

[19] Garrett Gee and Peter Kim. Doppleganger Domains. http://www.wired.com/images_blogs/threatlevel/2011/09/Doppelganger.Domains.pdf, 2011.

[20] Jan Göbel, Thorsten Holz, and Philipp Trinius. Towards Proactive Spam Filtering. In *Proc. of 6th DIMVA*, 2009.

[21] Chris Grier, Kurt Thomas, Vern Paxson, and Michael Zhang. @spam: The Underground on 140 Characters or Less. In *Proc. of 17th ACM CCS*, 2010.

[22] Guofei Gu, Junjie Zhang, and Wenke Lee. BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic. In *Proc. of 15th NDSS*, 2008.

[23] Pedro H. Calais Guerra, Dorgival Guedes, Wagner Meira Jr., Cristine Hoepers, Marcelo H. P. C. Chaves, and Klaus Steding-Jessen. Spamming Chains: A New Way of Understanding Spammer Behavior. In *Proc. of 6th CEAS*, 2009.

[24] Pedro H. Calais Guerra, Dorgival Guedes, Wagner Meira Jr., Cristine Hoepers, Marcelo H. P. C. Chaves, and Klaus Steding-Jessen. Exploring the Spam Arms Race to Characterize Spam Evolution. In *Proc. of 7th CEAS*, 2010.

[25] S. Hao, N. Feamster, A. Gray, N. Syed, and S. Krasser. Detecting Spammers with SNARE: Spatio-Temporal Network-Level Automated Reputation Engine. In *Proc. of 18th USENIX Security*, 2009.

[26] Thorsten Holz, Christian Gorecki, Konrad Rieck, and Felix C. Freiling. Measuring and Detecting Fast-Flux Service Networks. In *Proc. of 15th NDSS*, 2008.

[27] Xin Hu, Matthew Knysz, and Kang G. Shin. RB-Seeker: Auto-detection of Redirection Botnets. In *Proc. of 16th NDSS*, 2009.

[28] Geoff Hulten, Anthony Penta, Gopalakrishnan Seshadrinathan, and Manav Mishra. Trends in Spam Products and Methods. In *Proc. of 1st CEAS*, 2004.

[29] D. Irani, S. Webb, J. Giffin, and C. Pu. Evolutionary Study of Phishing. In *eCrime Researchers Summit*, pages 1–10, 2008.

[30] Andreas GK Janecek, Wilfried N. Gansterer, and K. Ashwin Kumar. Multi-Level Reputation-Based Greylisting. In *Proc. of 3rd ARES*, pages 10–17, 2008.

[31] John P. John, Alexander Moshchuk, Steven D. Gribble, and Arvind Krishnamurthy. Studying Spamming Botnets Using Botlab. In *Proc. of 6th NSDI*, 2009.

[32] Jaeyeon Jung and Emil Sit. An Empirical Study of Clustering Behavior of Spammers and Group-based Anti-Spam Strategies. In *Proc. of 4th IMC*, pages 370–375, New York, NY, USA, 2004. ACM Press.

[33] Chris Kanich, Christian Kreibich, Kirill Levchenko, Brandon Enright, Geoffrey M. Voelker, Vern Paxson, and Stefan Savage. Spamalytics: An Empirical Analysis of Spam Marketing Conversion. In *Proc. of 15th ACM CCS*, 2008.

[34] Bryan Klimt and Yiming Yang. Introducing the Enron Corpus. In *Proc. of 1st CEAS*, 2004.

[35] Maria Konte, Nick Feamster, and Jaeyeon Jung. Dynamics of Online Scam Hosting Infrastructure. In *Proc. of 10th PAM*, 2009.

[36] Krebs on Security. Body Armor for Bad Web Sites. http://krebsonsecurity.com/2010/11/body-armor-for-bad-web-sites/, 2010.

[37] Christian Kreibich, Chris Kanich, Kirill Levchenko, Brandon Enright, Geoffrey M. Voelker, Vern Paxson, and Stefan Savage. On the Spam Campaign Trail. In *Proc. of 1st USENIX LEET*, pages 1:1–1:9, Berkeley, CA, USA, 2008. USENIX Association.

[38] Christian Kreibich, Chris Kanich, Kirill Levchenko, Brandon Enright, Geoffrey M. Voelker, Vern Paxson, and Stefan Savage. Spamcraft: An Inside Look at Spam Campaign Orchestration. In *Proc. of 2nd USENIX LEET*, 2009.

[39] Martin Lee. Why My Email Went. http://www.symantec.com/connect/blogs/why-my-email-went, 2011.

[40] LegitScript and eNom. LegitScript Welcomes Agreement with eNom (DemandMedia). http://www.legitscript.com/blog/142, 2010.

[41] LegitScript and KnujOn. No Prescription Required: Bing.com Prescription Drug Ads. http://www.legitscript.com/download/BingRxReport.pdf, 2009.

[42] LegitScript and KnujOn. Yahoo! Internet Pharmacy Advertisements. http://www.legitscript.com/download/YahooRxAnalysis.pdf, 2009.

[43] LegitScript and KnujOn. Rogues and Registrars: Are some Domain Name Registrars safe havens for Internet drug rings? http://www.legitscript.com/download/Rogues-and-Registrars-Report.pdf, 2010.

[44] Barry Leiba and Jim Fenton. DomainKeys Identified Mail (DKIM): Using Digital Signatures for Domain Verification. In *Proc. of 4th CEAS*, 2007.

[45] Nektarios Leontiadis, Tyler Moore, and Nicolas Christin. Measuring and Analyzing Search-Redirection Attacks in the Illicit Online Prescription Drug Trade. In *Proc. of USENIX Security*, 2011.

[46] Kirill Levchenko, Andreas Pitsillidis, Neha Chachra, Brandon Enright, Márk Félegyházi, Chris Grier, Tristan Halvorson, Chris Kanich, Christian Kreibich, He Liu, Damon McCoy, Nicholas Weaver, Vern Paxson, Geoffrey M. Voelker, and Stefan Savage. Click Trajectories: End-to-End Analysis of the Spam Value Chain. In *Proc. of IEEE Symposium on Security and Privacy*, 2011.

[47] He Liu, Kirill Levchenko, Márk Félegyházi, Christian Kreibich, Gregor Maier, Geoffrey M. Voelker, and Stefan Savage. On the Effects of Registrar-level Intervention. In *Proc. of 4th USENIX LEET*, 2011.

[48] Brian Livingston. Web registrars may take back your domain name. http://news.cnet.com/2010-1071-281311.html, 2000.

[49] Daniel Lowd and Christopher Meek. Good Word Attacks on Statistical Spam Filters. In *Proc. of 2nd CEAS*, 2005.

[50] M86 Security Labs. Top Spam Affiliate Programs. http://www.m86security.com/labs/traceitem.asp?article=1070, 2009.

[51] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Identifying Suspicious URLs: An Application of Large-Scale Online Learning. In *Proc. of 26th ICML*, 2009.

[52] Marshal8e6 TRACELabs. Marshal8e6 Security Threats: Email and Web Threats. http://www.marshal.com/newsimages/trace/Marshal8e6_TRACE_Report_Jan2009.pdf, 2009.

[53] Justin Mason. SpamAssassin public corpus. http://spamassassin.apache.org/publiccorpus, 2003.

[54] Mohammad M Masud, Latifur Khan, and Bhavani Thuraisingham. Feature Based Techniques for Auto-Detection of Novel Email Worms. In *Proc. of 11th PACKDDD*, 2007.

[55] Damon McCoy, Andreas Pitsillidis, Grant Jordan, Nicholas Weaver, Christian Kreibich, Brian Krebs, Geoffrey M. Voelker, Stefan Savage, and Kirill Levchenko. PharmaLeaks: Understanding the Business of Online Pharmaceutical Affiliate Programs. In *Proc. of the USENIX Security Symposium*, 2012.

[56] D. Kevin McGrath and Minaxi Gupta. Behind Phishing: An Examination of Phisher Modi Operandi. In *Proc. of 1st USENIX LEET*, 2008.

[57] Robert McMillan. What will stop spam?, December 1997.

[58] Brain S. McWilliams. *Spam Kings: The Real Story Behind the High-Rolling Hucksters Pushing Porn, Pills and @*#?% Enlargements*. O'Reilly Media, September 2004.

[59] Tony A. Meyer and Brendon Whateley. SpamBayes: Effective open-source, Bayesian based, email classification system. In *Proc. of 1st CEAS*, 2004.

[60] Tyler Moore and Richard Clayton. Examining the Impact of Website Takedown on Phishing. In *Proc. of 2nd eCrime Researchers Summit*. ACM, 2007.

[61] Andy Mutton. Screengrab! http://www.screengrab.org/, 2010.

[62] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D. Joseph, Benjamin I. P. Rubinstein, Udam Saini, Charles Sutton, J. D. Tygar, and Kai Xia. Exploiting Machine Learning to Subvert Your Spam Filter. In *Proc. of 1st USENIX LEET*, 2008.

[63] Yuan Niu, Yi-Min Wang, Hao Chen, Ming Ma, and Francis Hsu. A Quantitative Study of Forum Spamming Using Context-based Analysis. In *Proc. of 14th NDSS*, 2007.

[64] Chris Nunnery, Greg Sinclair, and Brent ByungHoon Kang. Tumbling Down the Rabbit Hole: Exploring the Idiosyncrasies of Botmaster Systems in a Multi-Tier Botnet Infrastructure. In *Proc. of 3rd USENIX LEET*, 2010.

[65] ODP – Open Directory Project. http://www.dmoz.org, September 2011.

[66] Emanuele Passerini, Roberto Paleari, Lorenzo Martignoni, and Danilo Bruschi. FluXOR: Detecting and Monitoring Fast-Flux Service Networks. In *Proc. of 5th DIMVA*, 2008.

[67] Abhinav Pathak, Y. Charlie Hu, and Z. Morley Mao. Peeking into Spammer Behavior from a Unique Vantage Point. In *Proc. of 1st USENIX LEET*, 2008.

[68] Abhinav Pathak, Feng Qian, Y. Charlie Hu, Z. Morley Mao, and Supranamaya Ranjan. Botnet Spam Campaigns Can Be Long Lasting: Evidence, Implications, and Analysis. In *Proc. of 9th ACM SIGMETRICS*, 2009.

[69] Roberto Perdisci, Igino Corona, David Dagon, and Wenke Lee. Detecting Malicious Flux Service Networks through Passive Analysis of Recursive DNS Traces. In *Proc. of 25th ACSAC*, 2009.

[70] Zhiyun Qian, Zhuoqing Mao, Yinglian Xie, and Fang Yu. On Network-level Clusters for Spam Detection. In *Proc. of 17th NDSS*, 2010.

[71] Anirudh Ramachandran, David Dagon, and Nick Feamster. Can DNSBLs Keep Up with Bots? In *Proc. of 3rd CEAS*, 2006.

[72] Anirudh Ramachandran and Nick Feamster. Understanding the Network-Level Behavior of Spammers. In *Proc. of ACM SIGCOMM*, 2006.

[73] Anirudh Ramachandran, Nick Feamster, and Santosh Vempala. Filtering Spam with Behavioral Blacklisting. In *Proc. of 14th ACM CCS*, 2007.

[74] Rhyolite Corporation. Distributed checksum clearinghouse, 2000.

[75] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A Bayesian Approach to Filtering Junk E-Mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.

[76] Dmitry Samosseiko. The Partnerka — What is it, and why should you care? In *Proc. of Virus Bulletin Conference*, 2009.

[77] Fernando Sanchez, Zhenhai Duan, and Yingfei Dong. Understanding Forgery Properties of Spam Delivery Paths. In *Proc. of 7th CEAS*, 2010.

[78] Matt Sergeant. Internet Level Spam Detection and SpamAssassin 2.50. In *MIT Spam Conference*, 2003.

[79] Sushant Sinha, Michael Bailey, and Farnam Jahanian. Shades of Grey: On the effectiveness of reputation-based blacklists. In *Proc. of 3rd MALWARE*, 2008.

[80] Sushant Sinha, Michael Bailey, and Farnam Jahanian. Improving SPAM Blacklisting through Dynamic Thresholding and Speculative Aggregation. In *Proc. of 17th NDSS*, 2010.

[81] Henry Stern. A Survey of Modern Spam Tools. In *Proc. of 5th CEAS*, 2008.

[82] Kevin Stevens. The Underground Economy of the Pay-Per-Install (PPI) Business. http://www.secureworks.com/research/threats/ppi, 2009.

[83] Brett Stone-Gross, Christopher Kruegel, Kevin Almeroth, Andreas Moser, and Engin Kirda. FIRE: FInding Rogue nEtworks. In *Proc. of 25th ACSAC*, 2009.

[84] Bradley Taylor. Sender Reputation in a Large Webmail Service. In *Proc. of 3rd CEAS*, 2006.

[85] Olivier Thonnard and Marc Dacier. A Strategic Analysis of Spam Botnets Operations. In *Proc. of 8th CEAS*, 2011.

[86] 2007 TREC Public Spam Corpus. http://plg.uwaterloo.ca/~gvcormac/treccorpus07, 2007.

[87] Trustwave. Spam Statistics – Week ending Sep 2, 2012. https://www.trustwave.com/support/labs/spam_statistics.asp, September 2012.

[88] Yi-Min Wang, M. Ma, Y. Niu, and H. Chen. Spam Double-Funnel: Connecting Web Spammers with Advertisers. In *Proc. of 16th WWW*, 2007.

[89] Gary Warner. Random Pseudo-URLs Try to Confuse Anti-Spam Solutions. http://garwarner.blogspot.com/2010/09/random-pseudo-urls-try-to-confuse-anti.html, September 2010.

[90] Chun Wei, Alan Sprague, Gary Warner, and Anthony Skjellum. Identifying New Spam Domains by Hosting IPs: Improving Domain Blacklisting. In *Proc. of 7th CEAS*, 2010.

[91] Andrew G. West, Adam J. Aviv, Jian Chang, and Insup Lee. Spam Mitigation Using Spatio-temporal Reputations From Blacklist History. In *Proc of 26th. ACSAC*, 2010.

[92] John Whissell and Charles Clarke. Clustering for Semi-Supervised Spam Filtering. In *Proc. of 8th CEAS*, 2011.

[93] Colin Whittaker, Brian Ryner, and Marria Nazif. Large-Scale Automatic Classification of Phishing Pages. In *Proc. of 17th NDSS*, 2010.

[94] Meng Weng Wong. Sender Authentication: What To Do, 2004.

[95] Yinglian Xie, Fang Yu, Kannan Achan, Rina Panigrahy, Geoff Hulten, and Ivan Osipkov. Spamming Botnets: Signatures and Characteristics. In *Proc. of ACM SIGCOMM*, pages 171–182, 2008.

[96] Le Zhang, Jingbo Zhu, and Tianshun Yao. An evaluation of statistical spam filtering techniques. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(4):243–269, December 2004.

[97] Yao Zhao, Yinglian Xie, Fang Yu, Qifa Ke, Yuan Yu, Yan Chen, and Eliot Gillum. BotGraph: Large-Scale Spamming Botnet Detection. In *Proc. of 6th NSDI*, 2009.

[98] Li Zhuang, John Dunagan, Daniel R. Simon, Helen J. Wang, Ivan Osipkov, Geoff Hulten, and J.D. Tygar. Characterizing Botnets from Email Spam Records. In *Proc. of 1st USENIX LEET*, 2008.

[99] Jonathan Zittrain and Laura Frieder. Spam Works: Evidence from Stock Touts and Corresponding Market Activity. Social Science Research Network, March 2007.