A One-Year Perspective on Exposed In-memory Key-Value Stores

Tobias Fiebig
TU Berlin
tobias@inet.tu-berlin.de

Anja Feldmann
TU Berlin
anja@inet.tu-berlin.de

Matthias Petschick
TU Berlin
matthias@sect.tu-berlin.de

ABSTRACT

Today's highly-scalable low-latency Web services rely on in-memory key-value stores. While they are essential to improve Web service performance they should not be exposed to the Internet. Security problems range from data leakage to remote code execution. In this paper we use a year long data set of exposed Redis and memcached instances to highlight the magnitude (about 200K) of the problem, document new transitive attacks, and explore misconfiguration patterns. We find that the number of exposed instances is constantly on the rise and that even severe problems only lead to temporal decreases. However, by correlating misconfiguration patterns we can explain significant changes in the number of exposed systems.

CCS Concepts

•Security and privacy → Access control; Database and storage security; Information accountability and usage control; *Usability in security and privacy;* **•Networks → Network security;**

Keywords

Security; Measurement; Misconfiguration; Key-Value-Stores

1. INTRODUCTION

Key-value stores have emerged as a vital technology to support highly-scalable low-latency Web services. They were first documented in research literature by Amazon with Dynamo [1]. To offer high performance and ease of use key-value stores remove features of traditional Relational Database Management Systems (RDBMS) such as inter-object relations and type-dependent data fields. As consequence, key-value stores are excellent systems for services which need to store small values associated with an unique index key. Their intended deployment environment is a dedicated, well protected, and isolated backend network segment [1].

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

Today, due to their ease of use, high performance, and widespread availablility in almost all cloud environments almost every Web 2.0 service relies on key-value stores [2]. However, this increasing popularity has also lead to a wide spread deployment of unprotected key-value stores. This was first noticed in early 2015 [3] and affects more than 200,000 systems. Indeed, several key-value store implementations are commonly run without authentication or authorization while being Internet-wide reachable. Since then, various authors have investigated what kind of data is stored in such publicly reachable key-value stores [4, 5, 6]. The data includes personal data, credit card information, medical data, cached Web pages and byte-code, etc. Thus, in principle the data confidentiality and integrity of tens of thousands of Web applications at risk.

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

In an ideal world today no key-value stores services should be reachable via the Internet. However, we find that this is not the case. Overall, the number of unprotected instances is growing even though it may temporarily decrease as a result of public disclosure of exploits. This motivates our year-long survey of unprotected instances of two prominent key-value stores, namely *Redis* and *memcached*. Our key contributions can be summarized as follows:

- We document new *transitive* attacks on key-value stores.
- We document the magnitude of the problem of exposed key-value stores using Redis and memcached as case studies.
- We point out that even significant attack possibilities only lead to a temporary decrease in the number of exposed instances.
- We distill insights regarding individual as well as organization misconfigurations.
- We hint at how misconfiguration patterns together with reverse DNS information may be used to track offending organizations, e.g., spammers, across the Internet.

Paper structure: Section 2 introduces key-value stores and describes possible attacks. We present our dataset in Section 3 and our observations in Section 4. Section 5 discusses our case-studies. We then conclude in Section 6.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SafeConfig'16, October 24 2016, Vienna, Austria
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4566-8/16/10. . . $15.00
DOI: http://dx.doi.org/10.1145/2994475.2994480

2. Key-value stores

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

Among the prominent in-memory key-value stores are *Redis* and *Memcached*. These systems are commonly used for low-latency access to shared or cached data [7]. Similar use cases apply to other key-value stores including MongoDB and ElasticSearch [4]. To understand the security implications of exposing deployed key-value stores to the Internet we highlight what they are used for. We then outline security implications of accessing/tainting data in key-value stores.

2.1 Key-Value Stores: Use-Cases

Amazon, Facebook, Digg, and Twitter [7] commonly use key-value stores to realize their services. Moreover, key-value stores are also used by many smaller companies and startups. Common use cases are caching [8], queuing [9] and providing featured lists, e.g., lists of the N most recent items [10].

Caching: Caching is a principle mechanism for improving performance. Hereby, key-value stores are often used to store previous or intermediate results. One example is traditional database queries where the client first checks if the result is already cached in the key-value stores. If yes it uses the result if not it issues the query against the original database and stores the result in the cache. Due to their in-memory design and key-based lookup pattern key-value stores significantly accelerate performances over traditional relational databases [11, 7].

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

Lists: Key-value stores often include support for ordered lists [10] and in particular for determining the top-N of such a sorted list which is an expensive operation in traditional RDBMSs [7]. Such lists are often used to hold volatile authentication data. Examples include lists containing web-sessions, file-share links with associated tokens and expiration time, and temporary access tokens handed out to devices after login. Such tokens are needed by all involved application servers as they are used to authenticate each request. Previous work [1, 7] shows that this is one of the major use-cases for key-value stores.

Queues: Key-value stores also often support queue types which are are useful for a wide range of applications from email and messaging to workload queues. Worker processes can access those queues to obtain data for distributed compute jobs. This can involve metadata on the processed job, i.e., a global tracking of the progress on a single job or the actual data to be processed distributively. Distributed system log message management frameworks like *Logstash* rely on key-value stores to handle the log messages of thousands of systems [9]. Note, that log messages often hold sensitive information on users' actions or may have to be covertly altered by an attacker to hide the evidence of a compromise [12].

2.2 Transitive attacks

As highlighted above key-value stores are often used to store sensitive data. Thus, access to or tainting of this data leads to confidentiality, integrity, and availability problems for the key-value stores itself, see, e.g., Zahid et al. [13] for an overview. However, there are also severe consequences for the service using the key-value store. Below we outline several, so far unreported, transitive attacks on such services via their key-value stores.

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

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

Web-Session Manipulation: If, for example, an authentication token (SessionID) with the session data of an application is stored in the key-value stores [7] an attacker can impersonate users or elevate the privileges of her own session within the application's context.

Redis Remote Shell: In October 2015 Salvatore "antirez" Sanfilippo, the lead developer of *Redis*, demonstrated a *Redis* specific exploit in a blog article [14] which leads to a remote shell. Since *Redis* allows an attacker to alter its configuration the attacker can instruct it to use `~/.ssh/author ized_keys` as database. By entering a ssh-public-key enclosed by "\n" as data for an arbitrary key to that database the attacker can obtain shell access.

3. DATASET

One of the Shadowserver Foundation's projects [15] aims at reducing the set of affected systems for various protocols that are potentially vulnerable, misconfigured, or packet amplification sources. Among these potentially vulnerable systems are, e.g., Redis and memcached.

The Shadowserver foundation utilizes a distributed scanning cluster to regularly scan the Internet for affected systems. It checks if a service is reachable and for key-value stores if they reply to status requests. Due to the missing privilege separation, this indicates full access to a key-value store. Thus, this key-value store is considered affected. Hence, these scans follow the zMap-style [16] two-step approach: First the general reachability of a port is established. Then, a service specific protocol handshake is performed. The scans check for Redis on port TCP/6379 and memcached on port TCP/11211. The data is then aggregated on two levels: the Autonomous System (AS) and CC (Country Code). Thus, for each AS/CC we can estimate the number of Internet-wide reachable systems on a daily basis. This information is made publicly available each day. However, no historical summary of the key-value stores scans are provided.

In late January 2015 the Shadowserver Foundation added Redis and memcached to their scanning project [15]. Other key-value stores's were added later, e.g., MongoDB mid-February 2015 and ElasticSearch in June 2015. We, for the purpose of this paper, choose Redis and memcached as our research focus is on in-memory key-value stores. We have gathered a year-long dataset of all exposed Redis and memcached instances on the Internet, starting on the 5th of February 2015 using the Shadowserver Foundation's public data feed.

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

Table 1: Memcached: Periods with significant changes in the number of exposed instances.

As the ShadowServer datasets are already highly aggregated we augment them with additional data. For this purpose, we perform our own targeted low bandwidth (5mbit) zMap [16] SYN only scan. But rather than targeting all ASes we focus on a subset of the ASes which we highlight in our case studies, see Section 5. To identify which prefixes to scan we use the HE BGP tool [17] to identify all de-aggregated /24 prefixes an AS originates at the time of a scan.

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

4. OBSERVATIONS

To get an overview of how the number of exposed Redis and memcached instances on the Internet evolved over time, Figure 1(a) and 1(b) show for each day the number of exposed instances. Surprisingly, given the severe security implications, we still see a steady increase in the number of exposed instances in both datasets. However, there is one major difference between the two key-value stores. In case of Redis the number of exposed instances is substantially reduced during the time period from early November 2015 to the beginning of January 2016.

Taking a closer look at the Redis dataset we identify three phases, see Figure 1(a). From the start of our observation period, Feb-05-2015, till about Nov-06-2015 we observe a steady increase of exposed instances. Here, the average increase of exposed instances is 62/434 per day/week. Starting from Nov-07-2015 to Jan-03-2016 we observe a huge drop by 30,575 exposed instances. The average decrease rate is 449 instances per day. This decrease is briefly interrupted by a short increase over the holiday season at the end of 2015. However, starting around Jan-04-2015 till the end of our observation period, Apr-01-2015, the number of exposed instances is on the rise again with an average increase of 109 per day.

In the memcached dataset, see Figure 1(b), we do not see such a significant decrease in the number of exposed instances as for the Redis dataset. Here, the increase of exposed instances is roughly 170/1130 per day/week over the whole observation period. However, note that the magnitude of memcached instances is significantly larger with a maximum of almost 200,000 However, Figure 1(b) shows some strange behavior towards the end of 2015 and in March 2016. Indeed, Table 1 summarizes short time periods when the number of exposed systems changed by more than 9,000. In Section 5 we take a closer look at them.

Next, we explore how the number of exposed system is distributed across the autonomous systems. Overall, our dataset finds exposed systems in 4,705 / 7,714 ASes for Redis / memcached. This corresponds to a large fraction of total number of ASes. In addition, as one may expect, not all ASes are affected equally. Accordingly, Figure 2(a) shows for Mar-14-2016 the concentration of the number of exposed key-value stores per AS for both Redis and memcached using a cumulative distribution function (CDF) with a log-scale x-axis. To ease interpretation of the results we added a support line for the Top 10% of the ASes. For memcached they host 90% of the exposed instance while for Redis they only host 82% of exposed instances. The Top 1% contribute 60%/50% of all exposed memcached/Redis instances. For other days we find similar distributions.

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

5. CASE-STUDIES

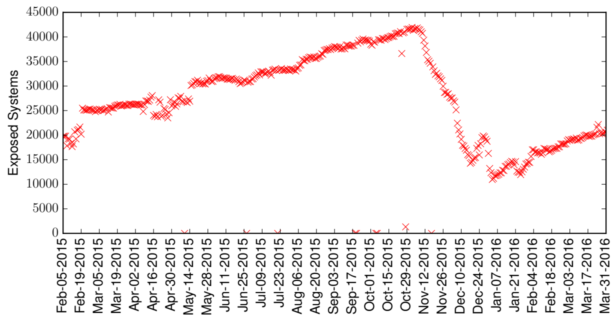
Next, we take a look at the possible causes for significant changes in the number of affected systems for both Redis and memcached.

5.1 Redis Decrease/Reincrease

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

On Oct-25-2015 Sanfilippo described how an exposed Redis server can be used to obtain remote shell access to a server [14], see Section 2. This story remained more or less dormant for the following nine days. On Nov-12-2015 various programming community-related news sites started to report on the possible exploit. Up to this time the increase of the exposed instances levels which indicate that some administrators already reacted based on the original blog post or an attack. Shortly afterwards, mainstream media picked up the issue. This corresponds to the sudden and forceful decline in the number of exposed Redis instances. Moreover, we find first evidence of the use of this exploit—stackoverflow posts [18]—on Nov-13-2015. The decline is for Redis only and did not carry over to other key-value stores, e.g., memcached. Given that this specific attack only affects Redis this may not be surprising. However, given the other attacks pointed out in Section 2, this is a severe issue which should be approached by the responsible parties.

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


(a) Redis

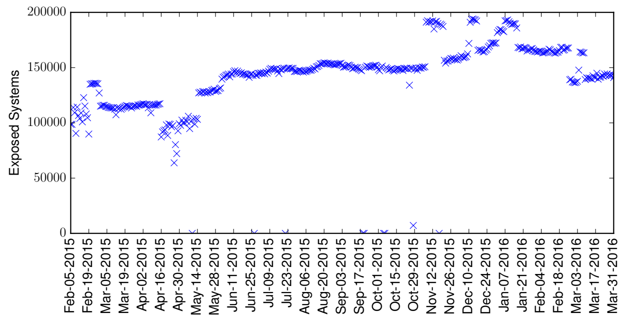
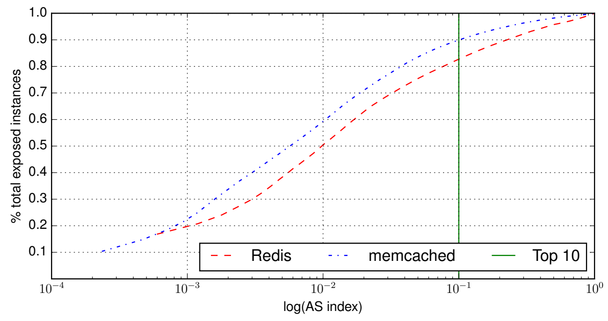

(b) Memcached

Figure 1: Redis and memcached: Number of exposed instances over time.


(a) CDF per ASN for Redis/Memcached—Mar-14-2016

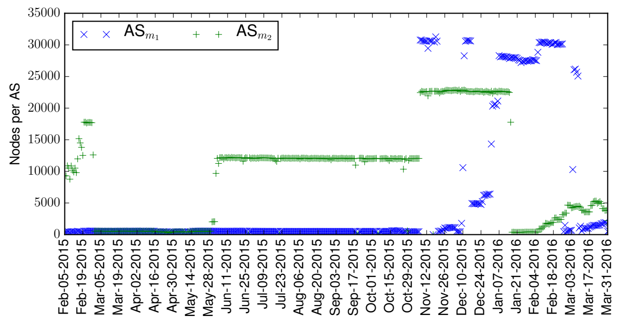
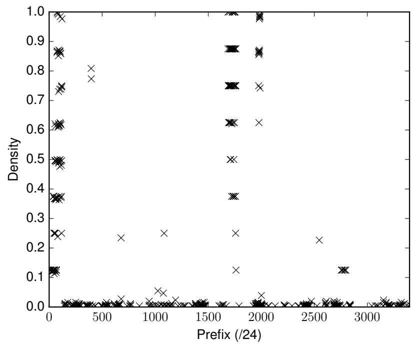
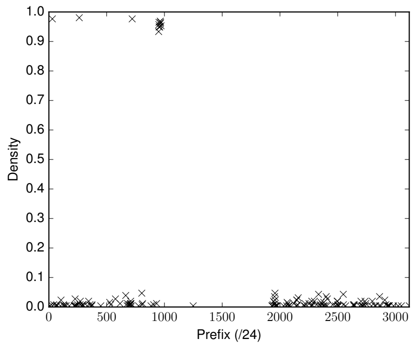

(b) Memcached: Number of exposed instance over time.

Figure 2: Redis/Memcached closeups: (a) CDF across ASN (b) Exposed instances for selected ASNs over time (Memcached only).


(a) AS_{m_1}, Feb-23-2016


(b) AS_{m_2}, Mar-16-2016

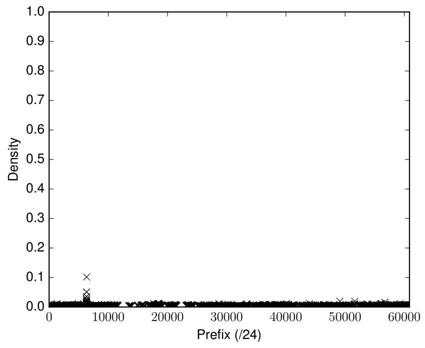

(c) AS_{m_3}, Feb-23-2016

Figure 3: Memcached: Density of exposed instances for AS_{m_1}, AS_{m_2} and AS_{m_3}.

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

5.2 Memcached Events

As pointed out the number of exposed memcached systems is varying, see Table 1. To understand its root cause we use the distribution per AS and identified two major contributing ASes: AS_{m_1} focuses on large web-hoster and also offers dedicated/virtual-private server provider; AS_{m_2} is another large hoster. In contrast to AS_{m_1} AS_{m_2} focuses mainly on co-location as well as dedicated/virtual-private servers. For comparison, we also consider AS_{m_3}: A large end-user PaaS cloud network.

To get an overview of how the number of exposed memcached instances for AS_{m_1} and AS_{m_2} changes over time Figure 2(b) shows for each day the number of exposed instances by these ASes. By comparing Figure 2(b) with Figure 1(b) we find that both ASes are responsible for most of the events of Table 1. One can see that most identified irregular events between Nov-2015 and Apr-2016 can be attributed to one of the two AS even given that AS_{m_2} partly (de)amplified the observable effects. Therefore we label each of the events with the AS that is the major contributor in Table 1.

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

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

Figure 3 shows the densities for our three ASes for Feb-23-2016 respectively Mar-16-2016. We picked Feb-23-2016 for AS_{m_1} and AS_{m_3} as there are a lot of exposed memcached instances within AS_{m_1}. We picked Mar-16-2016 for AS_{m_2} as the number of exposed memcached instances with AS_{m_2} is again around 5,000 instances[1]. AS_{m_3}, a large end-user PaaS cloud network, serves as baseline and hosts more than 4,000 exposed memcached instances. The density plot for February, Figure 3(c), and March, not shown, for AS_{m_3} shows that the exposed instances are more or less evenly distributed across the address space of the AS.

For AS_{m_1} and AS_{m_2}, see Figures 3(a) and 3(b), the densities support our hypothesis that here the exposed instances are under common administrative control. Indeed, for AS_{m_1} more than hundred adjacent /24s have significant densities. Indeed, these densities vary in multiples of 32 IPv4 addresses. It appears the allocation of addresses is done en-block on /28 granularity.

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

AS_{m_2} is responsible for the first large drop event in Feb-2015 of the exposed memcached instances. It re-introduced a large number of exposed memcached instances in the beginning of Jun-2015. This event was, however, counteracted by a simultaneous decrease in another AS and does, therefore, not significantly stick out in Figure 1(b). Its instances further increase at the same time when AS_{m_1} first exposed its large number of instances. However, in contrast to AS_{m_1}, AS_{m_2} mitigated the issue for nearly all exposed instances by mid January 2016. For AS_{m_2}, see Figure 3(b), we find 56 different /24 with a density higher than 0.9. This indicates that the systems in these subnets are under common administrative control.

Background checks showed that the corresponding network segments are most probably rented out. While AS_{m_2} already has a somewhat questionable reputation due to being regularly listed on various blacklists, these third parties are commonly known for shady practices. Indeed, all affected prefixes are on the Spamhaus blacklist for use in so-called snowshoe spamming. In snowshoe spamming, spammers reserve large prefix sets to send spam from all addresses within the network to bypass spam filtering [19]. Overall, we conclude that the system(s) behind these addresses are most likely under the control of a sub-sub-customer of AS_{m_2}.

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

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

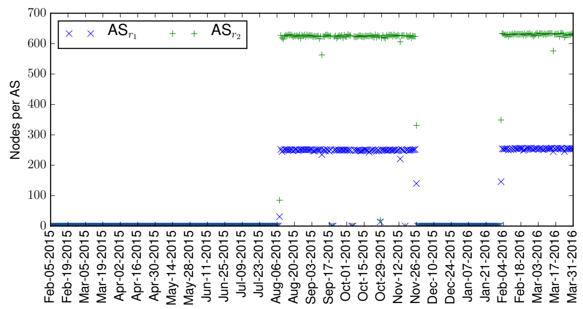


Figure 4: AS_{r_1} and AS_{r_2} from the Redis Dataset.

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

5.3 Redis anomalies

Next, we use the concept of prefixes with high-density of exposed instances per /24 to revisit the Redis data. Among the identified ASes with such prefixes are two relatively small ASes: AS_{r_1} is registered in Azerbaijan and announces one /21 and AS_{r_2} is registered in Kazakhstan which announces one /22 and one /20. Indeed, a detailed look at these ASes across time, Figure 4, shows the simultaneous sudden appearance and disappearance of exposed Redis instances.

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

6. SUMMARY

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

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

Moreover, as a significant number of these key-value stores systems are hosted in cloud environments individual as well as organization misconfigurations can drastically increase the number of exposed systems. We highlight this using exposed memcached instances in two ASes as examples. Indeed, such misconfigurations if cross-correlated with reverse DNS may even enable us to track spammers. Moreover, we show that it is possible to identify underlying organizational structures with concurrent misconfiguration patterns. We use Redis with two ASes as example to link two ASes with questionable reputation to a common operator.

Regarding future work it will be interesting to see if papers such as this one which points out the magnitude of issues actually have a longer lasting impact on reducing them. In this paper, we focused on the AS level. An analysis of the CC level may reveal further insights, e.g., on the performance of the respective country-level Computer Security Incident Response Teams (CSIRTs).

Acknowledgements

This work was supported by BMBF (Bundesministerium für Bildung und Forschung) Grant KIS1DSD032 (Project Enzevalos) and by Leibniz Price project funds of DFG/German Research Foundation (FKZ FE 570/4-1).

7. REFERENCES

[1] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's highly available key-value store," in *ACM SIGOPS Operating System Review*, vol. 41, no. 6, 2007, pp. 205–220.

[2] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, "Cloud computing-the business perspective," *Decision Support Systems*, vol. 51, no. 1, pp. 176–189, 2011.

[3] S. Srinivas and A. Nair, "Security maturity in NoSQL databases-are they secure enough to haul the modern it applications?" in *Proc. IEEE Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2015, pp. 739–744.

[4] Binary Edge. (2015, August) Data, technologies and security - part 1. http://blog.binaryedge.io/2015/08/10/data-technologies-and-security-part-1/.

[5] John Matherly. (2015, December) Memory as a service. https://blog.shodan.io/memory-as-a-service/.

[6] ——. (2015, July) It's the data, stupid! https://blog.shodan.io/its-the-data-stupid/.

[7] B. Atikoglu, Y. Xu, E. Frachtenberg, S. Jiang, and M. Paleczny, "Workload analysis of a large-scale key-value store," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 1, 2012, pp. 53–64.

[8] B. Fitzpatrick, "Distributed caching with memcached," *Linux journal*, no. 124, p. 5, 2004.

[9] J. Turnbull, *The Logstash Book*. James Turnbull, 2013.

[10] T. Macedo and F. Oliveira, *Redis cookbook*. "O'Reilly Media, Inc.", 2011.

[11] J. Ousterhout, P. Agrawal, D. Erickson, C. Kozyrakis, J. Leverich, D. Mazières, S. Mitra, A. Narayanan, G. Parulkar, M. Rosenblum *et al.*, "The case for ramclouds: scalable high-performance storage entirely in dram," *ACM SIGOPS Operating System Review*, vol. 43, no. 4, pp. 92–105, 2010.

[12] B. Schneier and J. Kelsey, "Secure audit logs to support computer forensics," *ACM Trans. on Information and System Security (TISSEC)*, vol. 2, no. 2, pp. 159–176, 1999.

[13] A. Zahid, R. Masood, and M. A. Shibli, "Security of sharded NoSQL databases: A comparative analysis," in *Proc. IEEE Conference on Information Assurance and Cyber Security (CIACS)*, 2014, pp. 1–8.

[14] S. "antirez" Sanfilippo. (2015, October) A few things about redis security. http://antirez.com/news/96.

[15] ShadowServer Foundation. (2014, March) The scannings will continue until the internet improves. http://blog.shadowserver.org/2014/03/28/the-scannings-will-continue-until-the-internet-improves/.

[16] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast internet-wide scanning and its security applications." in *Proc. Usenix Security Symp.*, 2013, pp. 605–620.

[17] Hurricane Electric. (2016, March) The hurricane electric bgp toolkit. http://bgp.he.net/.

[18] S. Upmanyu. (2015, November) Redis writing to .ssh/authorized_keys. http://stackoverflow.com/questions/33692230/redis-writing-to-ssh-authorized-keys.

[19] M. Van Eeten, J. M. Bauer, H. Asghari, S. Tabatabaie, and D. Rand, "The role of internet service providers in botnet mitigation an empirical analysis based on spam data," in *Proc. Research Conference on Communications, Information and Internet Policy formerly Telecommunications Policy Research Conference (TPRC)*, 2010.