An Empirical Analysis of Malware Blacklists

Marc Kührer and Thorsten Holz
Chair for Systems Security
Ruhr-University Bochum, Germany

Abstract

Besides all the advantages and reliefs the Internet brought us over the years, there are also a lot of suspicious and malicious activities taking place. Attackers are constantly developing new techniques to compromise computer systems. Furthermore, there are many malicious servers on the Internet that host, for example, exploits, drive-by download toolkits, or malicious software. We want to track the network locations of these malicious servers by analyzing different kinds of blacklists that provide a listing of suspicious servers.

In this article, we present the design and implementation of our blacklist parser system that tracks 49 different blacklists. We have collected more than 2.2 million distinct blacklist entries and more than 410,000 distinct URLs in the first 80 days of running the system. Besides discussing the design, we also provide an overview of the first empirical results of analyzing the collected data. In the future, we plan to extend the system such that it provides a comprehensive overview of malicious activities on the Internet.

1 Introduction

The Internet has evolved to a system used by hundreds of million people world-wide on a daily basis. We use the Internet to search for information, to order goods, or to keep in contact with friends. Besides all the advantages and reliefs the Internet brought us, there are also a lot of abuses going on. One of the main problems we need to deal with in today's Internet are so-called *botnets*. A botnet is a network of compromised machines under the control of an attacker [3, 13] who uses this mechanism as a tool to make attacks more scalable. For example, the majority of today's e-mail spam is sent with the help of botnets [11], and botnets are frequently used to steal credentials from compromised machines [7]. The core behind a botnet is the *Command and Control* (C&C) server, which is used to distribute the attacker's commands to the infected machines. There are different types of network setups for C&C servers [3], with the majority relying on one central or a small number of central servers. Since the C&C servers play a critical role for the whole botnet, a promising approach to stop this threat is to identify the network locations of C&C servers. Once these servers are detected, we can monitor the network and block access to such servers, i.e., generate a blacklist of C&C servers which is then enforced at the network level.

Besides worrying about C&C servers, we also want to study other servers used by attackers to perform their malicious activities. For example, we want to track servers that host exploits, drive-by download toolkits, or malicious software. Nowadays, various kinds of blacklists are available, covering different types of information about past and current threats on the Internet (e.g. [1, 2, 5, 6, 9, 17]). Some of these lists solely include domain names, URLs, or IP addresses of a specific type of malicious activity, e.g., IP addresses of currently compromised server and client systems, or domains of websites providing malicious content. Other blacklists enclose much more information like timestamps or even source, type, and description for each entry. Due to the variable number of provided data, most of the available blacklists differ in their file format structure. As a result, every blacklist has to be handled differently, and parsing one specific list does not guarantee that we are capable of processing other blacklists as well.

In this paper, we present the results of an empirical analysis of different types of malware blacklists. We have developed a system to closely track 49 different blacklists and report on the results of analyzing the first 80 days of collected data. Our blacklist tracking system consists of several different modules (e.g. input normalization and DNS/HTTP/RBL query modules) which enable us to easily adopt the monitoring of different input sources. Up to now, our system has collected more than 2.2 million distinct blacklist entries and more than 410,000 distinct URLs, and we performed some initial analyses of this data set, which we present in this article. This data can be used for many different purposes, e.g., to detect infected machines on the network layer in combination with an Intrusion Detection System (IDS). In the future, we plan to extend the system such that it provides a comprehensive overview of malicious activities on the Internet.

The rest of this article is organized as follows. In Section 2, we discuss related work in this area, and in Section 3 we present an overview of the design and implementation of our blacklist tracking system. First empirical results are discussed in Section 4, and we conclude the article in Section 5 with an outlook for future work in this area.

2 Related Work

There have been several studies of different kinds of blacklists in the past. For example, Sheng et al. performed an empirical analysis of phishing blacklists [15]. They analyzed 191 fresh phishing websites that were less than 30 minutes old to conduct two tests on eight anti-phishing toolbars. The authors observed that only 20% of the phishing sites were included at hour zero, significantly reducing the effectiveness of these blacklists to actually protect users. In a similar study, Zhang et al. evaluated 200 verified phishing websites from two sources and 516 legitimate sites to test the effectiveness of ten anti-phishing tools [20]. They also found that phishing blacklists only provide a limited coverage and often do not include suspicious sites in a timely manner. In our work we focus on tracking a wide variety of blacklists. Later we also plan to perform a temporal analysis of the different data sources to study how timely they react to new malicious domains or IP addresses.

Reputation-based blacklists to block spam were analyzed by Ramachandran et al. [14] and Sinha et al. [16]. Both found that such lists have a significant number of false negatives and react slowly to new spam runs. We analyze several different kinds of blacklists and plan to also perform this kind of temporal tests.

Zhang et al. [19], Felegyhazi et al. [4], and Prakash et al. [12] studied ways to generate *predictive blacklists*, i.e., blacklists that can adopt to local settings and predict the malicious use of domains/IP addresses proactively. Our goal is to study current blacklists and correlate the information contained in different lists, but in the future we also plan to analyze ways to predict potentially malicious regions on the Internet.

3 Design and Implementation

In the following, we present the design and several implementation details of our blacklist tracking system. Since there are many different kinds of blacklists available, we have designed our system in such a way that it can be easily adopted to different input formats and sources. The software is modular and each module has a specific task, e.g., to perform a specific kind of query or to parse the input according to specific rules.

Blacklist parsing: We found that the different blacklists provide the data in a diverse set of formats and thus we had to develop a component to normalize and parse the input. Our system needs to normalize the received input, since a variety of different characters are utilized to separate blacklist entries from others. Some of the lists apply new line characters (`\n`), others employ spaces or tabs (`\t`) or even use patterns like `"entry", "entry", [...]`. Furthermore, the way to introduce comments differs in many cases. Comments are preluded by the pound (`#`) character in most of the blacklists, however, C and C++ style comments (`/* */` and `//`) also appear quite often. To handle all these different file formats without creating a specific file parser for every blacklist, we decided to implement a generic parser which is able to handle various file formats with help of identifier strings like `%domain`, `%ignore`, `%w`, and `%newline`. These identifiers are combined to a pattern that defines the format structure of a blacklist. For instance, the pattern `%newline%w%domain%ignore` describes a domain blacklist in which every domain is stated in its own line as follows:

- the first identifier `%newline` ignores potential new line characters,
- the `%w` string ignores whitespace characters like spaces and tabs,
- the `%domain` identifier parses the actual domain information from the blacklist,
- and `%ignore` simply ignores the rest of the line, in the case the blacklist includes negligible information.

While parsing the blacklist content, the format pattern is applied to every single line to identify and filter all stored domains. Currently, 28 identifiers are known to the parser to support diverse file formats and extract all valuable information from the lists. We found this list of identifiers enough to parse all blacklists we are covering for now.

To add new blacklists to the parser the download location, the comment style, and the file format pattern have to be inserted into a configuration file. Using this configuration-based handling of blacklists simplifies the maintenance compared to writing a file parser for every single list. As a result, blacklists can be added or modified within a short timeframe and without altering the source code of the parser.

System Design: Figure 1 illustrates the architecture and the different components of the implemented blacklist tracking system. The system requires access to an external SQL database to store the blacklist information. The actual downloading, parsing, and post-processing of blacklists is handled by the system itself. Once a blacklist has to be processed, the parser activates the build-in download module which attempts to receive the blacklist from the given download location stated in the configuration file. Occasionally, the download module also executes external applications, i.e., when the downloaded file has to be unzipped.

If downloading the desired blacklist was successful the file content and the format pattern from the configuration file are forwarded to the parser module. As already stated above, each line of the blacklist is then parsed with the specified file format pattern. The parsed information (IP addresses, domain names, and URLs) is then forwarded to

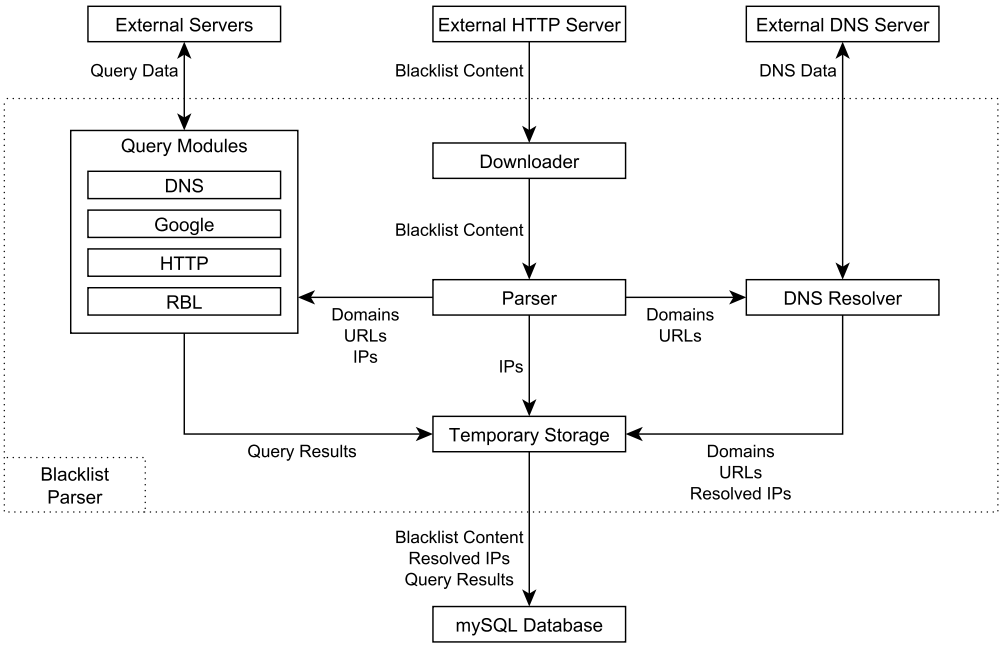

Figure 1: Architecture of the implemented blacklist parser.

further processing modules. IP addresses are redirected to a temporary storage, whereas domain names and URLs are routed to the *DNS resolver module*. The DNS resolver tries to gather A records (i.e. IP addresses) for the domains and URLs, since this information might also be important for further investigations. Afterwards, the domains and URLs, aggregated with the corresponding IP addresses, are forwarded to the temporary storage as well.

Many downloadable blacklists are freely available on the Internet. However, some organizations do not want to publish all of their collected data and only provide the possibility to query, if a specific domain name or IP address is included in their blacklist. To access as many of these databases as possible the parser supports different types of query modules. All parsed domain names, URLs, and IP addresses of downloaded blacklists are forwarded to these query modules which then contact the external databases and check whether the entries are listed. The following query modules are supported for now:

- The *DNS query modules* act similar to the *DNS resolver module* and transmit requests to a DNS server which is directly connected to a blacklist. On requests, the DNS server checks whether or not the requested entry is in that blacklist database. In the case the entry is not blacklisted, the DNS server returns the ordinary A record, thus behaves like a normal DNS server. However, if the entry is listed the DNS server replies with a predefined IP address to indicate that the requested entry is blacklisted (e.g. with 127.0.0.1).
- The *Google query module* requests blacklist information from the *Google Safebrowsing API* [6]. The database of *Google Safebrowsing* has to be downloaded and locally stored, thus acts similar to a normal blacklist, except that the blacklist content is obfuscated (i.e. entries are hashed), such that it can not directly be monitored. As a result, the parser has to pre-process (normalize and hash) each request before it can be checked against the Google blacklist.
- *RBL query modules* work similar to *DNS query modules*. Requests are sent to specialized RBL (*Realtime Blackhole List*) servers which hand back IP addresses to indicate whether the requested entries are covered by the blacklist or not.
- *HTTP query modules* attempt to access blacklist databases via the HTTP protocol. Many organizations, e.g., different vendors of anti-virus software, provide online blacklists. To obtain information from these databases the parser has to send HTTP queries to the corresponding web server and receives HTML code which indicates if the requested domains or IP addresses are blacklisted. Unfortunately, these replies include much more HTML statements than a simple listed/not-listed indication, hence HTTP queries are potentially consuming a lot of traffic. However, accessing these databases via HTTP is the only possibility when organizations do not offer direct access via protocols such as DNS or RBL.

Once all query modules finished handling the content of a downloaded blacklist, the gathered information is sent to the temporary storage and afterwards written to the external SQL database. This database provides various tables to store the blacklist content, resolved IP addresses, and query results including all relevant timestamps. With help of these timestamps, we are able to monitor the history of blacklist entries, i.e., at what time entries got removed from blacklists or re-added after a while.

4 Empirical Results

We now present some first results gathered from the first 80 days of tracking 49 different malware blacklists. Altogether, we collected 2,289,299 distinct blacklist entries and 410,864 distinct URLs up to now. Furthermore, we aggregated history information for 671,072 blacklist entries.

To provide a more detailed insight into the amount of processed data we introduce seven of the 49 currently processed blacklist databases in Table 1. The number specified in the *Active* column describes the amount of entries currently seen in the respective blacklist, while the data given in the *History* column covers information about entries which were removed from the blacklist during the initial analysis period. Furthermore, the *Revoked History* column includes entries which were removed from the blacklist but got re-added later again. As can be seen, the amount of active entries differs considerably for the exemplary blacklists. Especially DNS- and HTTP-based blacklists, such as *ClearCloud DNS*, *McAfee SiteAdvisor*, *Trend Micro Site Safety Center*, and *Web Of Trust*, cover a huge amount of blacklisted data as a side-effect of how our blacklist parser queries these lists.

Database	**Type**	**Active**	**History**	**Revoked History**
AMaDa Abuse.ch [1]	Download	1,337	48	130
ClearCloud DNS [5]	DNS Query	454,397	766	17,336
Malware Domain List (MDL) [9]	Download	39,622	27	106,640
McAfee SiteAdvisor [10]	HTTP Query	200,904	2,197	2,893
Trend Micro Site Safety Center [17]	HTTP Query	311,691	2,921	18,142
Web Of Trust [18]	HTTP Query	278,000	65	1,317
ZeuS Abuse.ch [2]	Download	613	2,471	3,389

Table 1: Detailed information on the amount of blacklisted entries in seven exemplary blacklists.

Due to the high number of blacklisted entries, it is of peculiar interest to find out how many entries are included in two or even more databases: such domains are likely malicious, since they were not only flagged by one source as suspicious, but at least two blacklists independently indicate that this particular entry is suspicious. Table 2 introduces the number of overlaps between each database. It can be seen that there are plenty of overlapping entries between *ClearCloud DNS* and *McAfee SiteAdvisor*, and *ClearCloud DNS* and *Trend Micro Site Safety Center*. However, only 148,579 distinct entries are both listed in the *McAfee SiteAdvisor* and *Trend Micro Site Safety Center*. The intersection between all three databases amounts to 137,808 matches, and intersecting these databases with *Web Of Trust* only reveals 107,638 matches. Compared to their overall sizes in Table 1, the blacklists have a huge amount of shared data, but every blacklist seems to also have other independent data sources. Note that *AMaDa* and *ZeuS* have an overlap of 0, since these two blacklists are designed in such a way that they are mutually exclusive.

Domains	AMaDa	ClearCloud DNS	MDL	Safety Center	SiteAdvisor	WOT	ZeuS
AMaDa	-	1,329	172	1,291	1,290	1,308	0
ClearCloud DNS	1,329	-	34,420	272,542	181,548	223,601	580
MDL	172	34,420	-	30,217	30,736	39,007	387
Safety Center	1,291	272,542	30,217	-	148,579	190,451	595
SiteAdvisor	1,290	181,548	30,736	148,579	-	153,780	569
Web Of Trust	1,308	223,601	39,007	190,451	153,780	-	547
ZeuS	0	580	387	595	569	547	-

Table 2: Intersection between the blacklists introduced in Table 1.

When taking a closer look at the intersection between *ClearCloud DNS* and *AMaDa*, almost all entries of *AMaDa* are included in the *ClearCloud DNS* blacklist. We found almost the same results for the other aforementioned large blacklists and the *ZeuS* tracker. Our guess is that *AMaDa* and *ZeuS* are used as data sources, or the larger blacklists employ similar methods to detect almost the same malicious activities as the *AMaDa* and *ZeuS* trackers. Similar results can be seen in the intersection of *Malware Domain List (MDL)* and *Web Of Trust*. The *Web Of Trust* database covers 39,007 out of the 39,622 MDL entries – almost 99 percent of the *Malware Domain List*.

Table 3 illustrates the overall Top 10 domains listed in all of the 49 investigated blacklists. It has to be mentioned that all popular entries are blacklisted in a large number of databases. The domain *ysbweb.com*, for example, is listed in 18 out of 49 blacklists. Eighteen blacklists do not sound impressive, but we have to keep in mind that approximately more than one third of the currently parsed blacklists cover IP addresses only instead of domains. Furthermore, some of the domain blacklists only focus on particular malicious activities. The second entry *zief.pl* belongs to a well-known botnet C&C server which has been around for several years. Several other frequently listed domains belong to botnets as well.

Entry	# Blacklists
ysbweb.com	18
zief.pl	17
xxxtoolbar.com	17
variantov.com	16
appppa1.ru	15
dnusax.com	15
afretroactive.com	15
hjwbxhqr.cn	15
brenz.pl	15
allvideo.org.uk	15

Table 3: Overall Top 10 domains and corresponding number of databases in which the domains are blacklisted.

Entry	# Linked Entries	# Linked IP addresses
ysbweb.com	35	1
zief.pl	2	2
xxxtoolbar.com	35	1
variantov.com	56	461
appppa1.ru	0	0
dnusax.com	314,243	21,737
afretroactive.com	0	0
hjwbxhqr.cn	0	0
brenz.pl	10	1
allvideo.org.uk	314,243	21,737

Table 4: Correlation of the overall Top 10 domains with other distinct blacklist entries and IP addresses.

A further remarkable fact is that many of the blacklisted entries are highly connected to other entries. Let us assume that domain *abc.com* resolves to the IP address 1.2.3.4, *def.com* to 1.2.3.4 and 5.6.7.8, and *ghi.com* to 5.6.7.8. Since the IP addresses of *abc.com* and *ghi.com* are both shared by domain *def.com*, all three domains are connected with each other, even when there is no direct link between *abc.com* and *ghi.com*. As can be seen in Table 4, some of the frequently listed domains are only linked to a few other blacklisted entries. However, there are also domains which are connected to thousands of other entries and IP addresses. The domain names *ysbweb.com* and *xxxtoolbar.com*, for example, belong to each other, since both of them resolve to the same IP address and have the same number of linked blacklist entries. The domains *dnusax.com* and *allvideo.org.uk* seem to belong to a large linked malware network using 21,737 distinct IP addresses and 314,243 distinct domains. Domain *variantov.com* might belong to a Fast-Flux network due to the large number of linked distinct IP addresses [8].

5 Conclusion and Future Work

In this article we presented the design and implementation of our blacklist parser and outlined a first empirical evaluation of malware blacklist data. In conclusion, collecting various different malware blacklists reveals innumerable information which can be used for further projects, such as the use of the collected blacklist data for the detection of malicious activities in an Intrusion Detection System (IDS). However, we have to continue processing the blacklists for a few more months to perform deeper analyses of the malware blacklist data. After that time, the collected data should become more informative and comprehensive, and we can also analyze temporal effects of the different blacklists.

For future work, the parser can be extended by other query modules in the case we find any blacklist databases which do not support the currently implemented query protocols. Furthermore, the HTTP query modules could be replaced by other protocol modules to reduce the traffic consumption on everyone's side while requesting blacklist information. Based on the data collected with the blacklist tracker, we plan to identify suspicious regions in the Internet which host a lot of malicious content. Aggregating and correlating the information stored in different blacklists enables us to obtain a more thorough overview of malicious activities on the Internet.

Acknowledgements

This work was in part supported by the German Federal Ministry of Education and Research under BMBF Grant 01BY1111 (*MoBE*).

References

[1] abuse.ch. AMaDa, 2011. `http://amada.abuse.ch`.

[2] abuse.ch. ZeuS Tracker, 2011. http://zeustracker.abuse.ch.

[3] D. Dagon, G. Gu, C. Lee, and W. Lee. A Taxonomy of Botnet Structures. In *Proceedings of the 23 Annual Computer Security Applications Conference (ACSAC'07)*, pages 325–339, December 2007.

[4] M. Felegyhazi, C. Kreibich, and V. Paxson. On the potential of proactive domain blacklisting. In *Proceedings of USENIX Workshop on Large-scale Exploits and Emergent Threats*, 2010.

[5] GFI Software. ClearCloud DNS, 2011. http://www.clearclouddns.com.

[6] Google. Google Safe Browsing API, 2011. http://code.google.com/intl/de-DE/apis/safebrowsing.

[7] T. Holz, M. Engelberth, and F. C. Freiling. Learning more about the underground economy: A case-study of keyloggers and dropzones. In *Proceedings of European Symposium on Research in Computer Security (ESORICS)*, pages 1–18, 2009.

[8] T. Holz, C. Gorecki, K. Rieck, and F. Freiling. Measuring and Detecting Fast-Flux Service Networks. In *15th Network & Distributed System Security Symposium (NDSS)*, Februar 2008.

[9] MalwareDomainList.com. Malware Domain List, 2009. http://www.malwaredomainlist.com.

[10] McAfee, Inc. SiteAdvisor, 2011. http://www.siteadvisor.com.

[11] MessageLabs. MessageLabs Intelligence: 2010 Annual Security Report. http://www.messagelabs.com/mlireport/MessageLabsIntelligence_2010_Annual_Report_FINAL.pdf, 2010.

[12] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta. Phishnet: Predictive blacklisting to detect phishing attacks. In *Proceedings of IEEE Infocom Mini-Conference*, pages 1–5, 2010.

[13] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A Multifaceted Approach to Understanding the Botnet Phenomenon. In *Proceedings of the 6th Internet Measurement Conference*, pages 41–52, 2006.

[14] A. Ramachandran, D. Dagon, and N. Feamster. Can DNS-based blacklists keep up with bots? In *Proceedings of Conference on Email and Anti-Spam (CEAS)*, 2006.

[15] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang. An empirical analysis of phishing blacklists. In *Proceedings of Conference on Email and Anti-Spam (CEAS)*, 2009.

[16] S. Sinha, M. Bailey, and F. Jahanian. Shades of grey: On the effectiveness of reputation-based blacklists. In *Proceedings of International Conference on Malicious and Unwanted Software*, pages 57–64, 2008.

[17] Trend Micro, Inc. Site Safety Center, 2011. http://global.sitesafety.trendmicro.com.

[18] WOT Services. Web Of Trust, 2011. http://www.mywot.com.

[19] J. Zhang, P. Porras, and J. Ullrich. Highly predictive blacklisting. In *Proceedings of USENIX Security Symposium*, pages 107–122, 2008.

[20] Y. Zhang, S. Egelman, L. F. Cranor, and J. Hong. Phinding phish: Evaluating anti-phishing tools. In *Proceedings of Symposium on Network and Distributed System Security (NDSS)*, 2007.