Characterizing Actively Distributed Malware

Henning Krause

HAW Hamburg

henning.krause@haw-hamburg.de

ABSTRACT

Malware is spread across the internet in various ways. The dataset of this work was collected by the reactive network telescope *Spoki*, which is actively approached by malware distributors. The malware is distributed to victims who have been identified using two–phase scanners. This methodology became prominent since the release of the Mirai botnet code. After engaging into interaction during the second phase by a TCP handshake, *Spoki* receives packets with a payload that contains a download URLs. Using the download URL from the payload, the samples are collected. The analysis of the data shows that the malware samples differ a lot in the matter of distribution infrastructure and behaviour. Furthermore, a first analysis of the samples indicates that a lot of samples work as droppers which are used to download further executables. This work gives an overview of the data and ideas for future work.

KEYWORDS

Malware

1 INTRODUCTION

A machine that is infected by a malicious software (malware) can be abused by the intruder in multiple ways—stealing sensitive data, causing unintended behaviour, such as launching DoS attacks and mining cryptocurrency, or even causing system failures, only to name a few. That makes malware one of the most severe threats. Most of the time, an infection is caused over the network. A common scenario is that a user visits an infected web page, which downloads and executes malware on the computer of the user. Furthermore, malware distributors are actively scanning to find vulnerable devices. This infection vector is a common approach for IoT malware.

Observing network traffic that is caused by malicious activity using a network telescope has been practised for a long time [6]. However, most network telescope only operated passively—that is collecting all network packets they receive. This works builds upon a dataset that was collected by *Spoki*, a reactive network telescope [7]. *Spoki* stands out due to its ability to react to two–phase scanners. Two–phase scanning is a method that first uses stateless TCP scans to find open ports and subsequently gathers further information with a second stateful phase. *Spoki* engages in the TCP handshake of the second phase and collects the payloads from the packets. By downloading the URLs from the payloads, the samples for our dataset could be collected. The observed spreading method differs from other distribution techniques, such as social engineering or drive by downloads [10], but is closer to the distribution of IoT malware, as it is utilized with the Mirai botnet code [3].

1.1 Problem statement

Malware must be stopped as early as possible, as a system may be hard or even impossible to recover once a malware settled in. During the life cycle of malware, the initial infection is the first step. Thus, it is important to understand what kind of malware is trying to intrude a system. Therefore, we aim to analyse the collected data in order to give a characterization of the samples. First, we characterize our collected dataset, followed by the distribution ecosystem and finally the executables themselves.

We use the collected data and extend it with information from the VirusTotal (VT) database and the MalwareBazaar database. VirusTotal is an information aggregator that presents data that is a combined output of different antivirus products [9]. Using the combined dataset, we aim to address the stated problems. The rest of the work is structured as follows. Section 2 gives an overview of related analysis methods found in the literature. In section 3 the collected data is presented and a summary of the used analysis methods is given. The analysis results are shown in section 4 followed by a conclusion and perspective on future work in section 5.

2 BACKGROUND

Analysing a dataset of malware samples or the network traffic that was generated by a malware is an approach that is often seen in literature. It is used to get an understanding of the ecosystem behind malware, the behaviour of the malware, past events and current emerging threats. The presented work is related to this work in either the analysis techniques or the goal of the work.

Pastrana and Suarez-Tangil [17] aim to characterize the crypto-mining malware ecosystem. They use public malware repositories, such as VirusShare and VirusTotal, to create a dataset that is being used to identify mining campaigns and their infrastructure by aggregating the samples based on metadata such as the addresses of the wallets which are being extracted from the binaries.

Using a dropper—a specialized or general–purpose program that is being used to download further malware binaries—to distribute malware is a central distribution technique. Kwon et al. [14] use the Worldwide Intelligence Network Environment (WINE) dataset to deduce downloader graphs for droppers to uncover relationships between the downloaders and the malware they download by associating the downloaded files with the URLs the file was downloaded from and the processes that triggered the download.

Focused on IoT malware, but still relevant in terms of analysis techniques and general malware ecosystem characteristics, Choi et al. [5] dissect a dataset that was collected by a Telnet Honeypot. By reverse engineering the samples, they identify “dropzone”

IPs—servers where malware binaries are stored for download—and "target" IPs—targets for further infections. Furthermore, Alrawi et al. [2] study the life cycle of IoT malware. They used data from network data collection points and VT to perform an extensive analysis of a set of malware samples. First, they characterized the samples using the metadata, then they performed a static analysis using multiple tools, such as the bash file tool, YARA signatures and UPX, to extract further information, e.g., IP addresses and infection vectors. Finally, a dynamic analysis is used to collect system and network calls. They show that IoT and traditional desktop malware have a similar life cycle.

Peng et al. [18] aim to explore the labelling process of VirusTotal in their work. Even though their focus is on the labelling of URLs, the question of how fast the results are available on the API and how consistent the labels are across different vendors is still of interest in a general context. They conclude that VirusTotal seems to pull (update) the labels from the vendors only when the scan API is triggered. The scan API is used when a URL is submitted, whereas the querying API provides the data from the database. Therefore, labels may be outdated and inconsistent with the vendor's API, when a URL has not been submitted for a while. Vendors only seem to share their results on demand and do not push an update to VirusTotal. This indicates that it may take some time until results show on VirusTotal. Furthermore, they showed that the performance considering the correct classification across different vendors vary.

Being related to the previous work, Kantchelian et al. [13] addressed the question of weighting the labels of the file scan API of VirusTotal and their consistency. They found that vendors are changing their labels over time as a rescan may happen that causes a change in their classification. Furthermore, the labels of a sample differ between the vendors, and it also has to be taken into account that there is a chance of a wrong label classification by individual vendors.

Focusing on DoS, Moura et al. [15] analysed the 2015 Root DNS Event, Jonker et al. [11] aimed to characterize the DoS ecosystem and Antonakakis et al. [3] looked at the Mirai Botnet. All works analyse the network traffic that was collected at collection points. The researchers focusing on the Mirai botnet furthermore analysed collected binary samples using YARA signatures and static analysis. In regard to this work, especially the analysis methods of the network traffic are interesting. Multiple cleaning, extraction and aggregation methods are used.

In their work, Invernizzi et al. [10] aim to detect malware distribution in large scale networks. They identify the traffic in multiple steps: First they filter by the MIME type of the payload, then they identify candidates who could be distributing malware and finally they aggregate the data to detect the distribution infrastructure. In this work, the detection of distribution candidates is not of interest, but the final aggregation and clustering of distributors is relevant.

3 METHODOLOGY

In this section, the individual data sources and the analysis methods are described. For each data source, a brief summary on which information is included and how the information is gathered is given.

3.1 Data Sources

Starting from the 29/3/2021 until the 19/8/2021 we were able to download a total of 8914 malware files on 104 days. The total downloads contain 241 unique executables. Due to a system outage, the collection period is divided by a gap in which no samples could be collected. Throughout the download process, different kind of data is collected. The following datasets were created which are used for the analysis.

Activity log. We define an activity as the attempt to download the data from a URL, which was extracted from a payload that we received from a scanner. In order to limit the download rate, there is a timeout for any given unique URL of 1 hour. The information about an event of a download attempt is logged in the activity log. Those include whether the download was successful, which status message we received from the connection, which URL was used for this download and at what time the download attempt was performed. If the download results in a successful downloaded executable, the hash value of the executable is added as well. Furthermore, we are logging if a download was not performed due to the timeout of a URL. For the second collection period we introduced the gathering of reverse DNS data for the used URLs. As this data is only available for half of the dataset, it is only of limited usage, but may be useful in the future.

Download log. If the download of an executable is successful, we save information about the event in the dedicated log of the specific file. Each executable is identified by its hash value. By using the hash, we can determine if we have downloaded the executable before, so there is only one sample of each unique executable stored. Irrespective of saving the file, a log entry is created for the download event. The logged information includes the name of the executable, the URL, the timestamp of the event, the server from which we downloaded the executable and the tag of the download event. The name of the executable and the server are extracted from the URL. For example, if the URL is *http://123.123.123.123/bins/evil.arm7* the server is interpreted as *123.123.123.123* and the name of file as *evil.arm7*. The tag of the event is set to the event tag which lead to the initial retrieval of the payload that included the download URL.

VirusTotal and MalwareBazaar. VirusTotal (VT) is an information aggregator which is collecting the ratings of over 70 antivirus vendors [9]. It is offering a public API which can be used to query hash values and as a result returns characteristics and general statistics about the file. When an executable is successfully downloaded for the first time we query VT. Depending on whether the executable is known to VT, we save the received data. During the finalization of our dataset we re-queried all files. Thus, we have two datasets of the VT information. One, that was created and expanded during the collection period and one that represents the status of the API on a specific date, namely the 26/8/2021. Looking at the second dataset, 189 of 241 samples were known to VT. MalwareBazaar [1] serves as a sharing platform for researchers where malware samples can be easily shared. It differs from VT in multiple ways: (I) It's database only consists of malware samples–no benign files or adware, (II) it is not a multi antivirus scanning engine and it contains confirmed malware samples only. All collected samples were

queried against MalwareBazaar's API to collect further information. In total 71 samples of our dataset were known to MalwareBazaar.

3.2 Analysis methods

Aggregating malware based on their metadata to form groups is a common approach [17]. We adapt this practice for our dataset and adjust those methods to fit our needs.

In order to create a basic understanding of our dataset, multiple characteristics can be determined. By using the activity log, it is possible to determine how often an executable is downloaded among other statistics. Additionally, the potential downloads, which means that a URL had a timeout, may be added as well. A possible flaw at this point is that the malware executable that is served by a specific URL may have changed during the timeout.

VirusTotal uses the *trid*[9] tool to determine the type of an executable file. As there are samples which are unknown to VT and we do not have the file type information as a consequence, we use the *file* standard program of Unix to determine the file types for the missing samples. In order to ensure conformity, a comparison of both tools was performed for any sample that was classified by both tools.

Each sample that is known to VT may have multiple labels. Each label can be given by a different number of vendors. We consider the label which was given by the most vendors the ground truth for the sample. Tools like *AVClass*[19] are used by researchers to automate this process [5]. In future work, the usage of such tools should be adapted as those tools additionally extract further information and take them into account.

YARA is a tool that aids researchers in identifying and classifying malware samples. It uses rules that identify a sample with textual or binary patterns. A common usage is to identify relevant samples in a dataset [3, 8, 17] or classify samples into categories[2]. The effectiveness of YARA is dependent on the rules that are used to for the identification. We used YARA to classify our samples into malware families.

4 ANALYSIS RESULTS

In this section the analysis results will be presented. First the general characteristics of our dataset are addressed, followed by the distribution ecosystem and behaviour and finally an analysis of the executables.

4.1 General characteristics

In period one 23984 activities were recorded which led to the download of 3579 executables that are compound of 139 unique samples. Period two has 27573 activities, 5335 downloads and 118 unique executables. 16 executables were seen in both periods. Figure 1 shows the number of downloads per day. Each executable in the figure has a unique colour. Throughout both collection periods an increase in the number of downloads is visible: in January the daily downloads were below 100 whereas they are above 300 in August. The share of the samples in the total of downloads of each day differs significantly between the months. However, it is clearly visible, that the downloads are dominated by several samples, which are

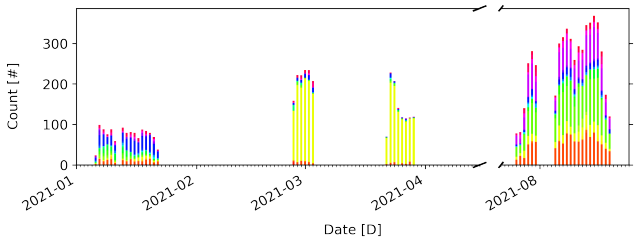


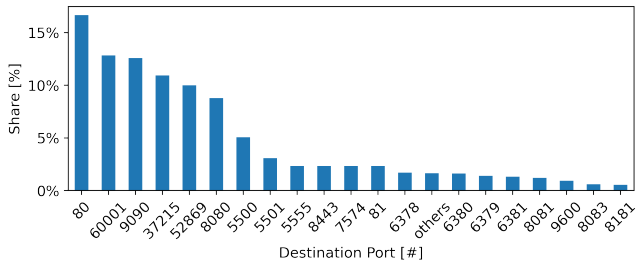
Figure 1: Number of successfully downloaded samples per day. Each unique executable was assigned an individual colour.

downloaded way more often than other executables. Especially in March and April most of the downloads are by one executable. In August, the share of this executable is lower, but it is still visible in the plot, which means that is seen comparatively often, even though several other samples are dominating this month. Due to the size of the plot not all samples of a day can be seen, but this visualization still demonstrates the composition of our dataset. On average, we receive 15.67 different unique samples per day in period one and 20.81 in period two. In period one, 93 samples (66.91%) were seen only on one collection day. This applied to 65 samples (55.08%) in period two.

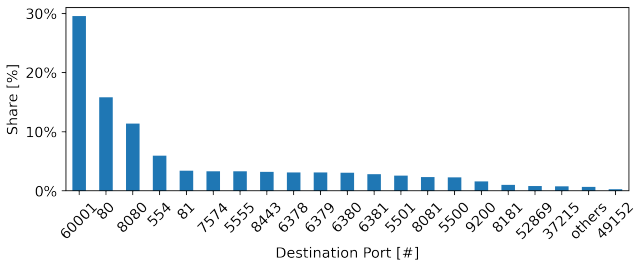
Due to the timeout of one hour for each unique URL, not all downloads are actually executed. When a URL is in timeout, the download is not attempted, but the activity that had the specific URL in its payload is still tracked. This restriction does not apply, when an executable is downloaded from different URLs. This is only the case for some executables; in both periods around three quarters were downloaded from one URL only. The other executables mostly have up to 100 different URLs. Seven extremes are present, with a maximum 2418 of URLs for one executable. The observation that most executables are always downloaded from the same URL indicates that during the timeout period a download could have been performed and would have led to the collection of a specific executable. On the other hand, there are several URLs from which we downloaded different executables. We further address this observation in section 4.2.

When a packet is received from a scanner with a downloader payload, different destination ports are used. In Figure 2 the distribution between the ports is shown. Some ports are assigned ports, e.g., port 80, while others are unregistered ports which are used for special purposes by specific devices. Those ports are sometimes known for services that can be exploited. For example, port 37215 is a known exploit vulnerability port for Huawei routers[1]. When comparing both periods as shown in Figures 2a and 2b, it strikes out that the distribution changed among the different ports. The top ports are different except for two ports. Port 9090 which had a share of over 10% in period one is not even in the top 20 in period two. Furthermore, port 60001 which was second most used port in period one with about 12.5% more than doubled its share in period two and becomes the top port with almost 30% percent of the total.

[1] https://vuldb.com/?id.114804



(a) Period one (Jan–Mar 2021).



(b) Period two (Jul–Aug 2021).

Figure 2: Top 20 ports targeted by scanners for both collection periods. The remaining ports are aggregated into "others". Note the different y–axis scales.

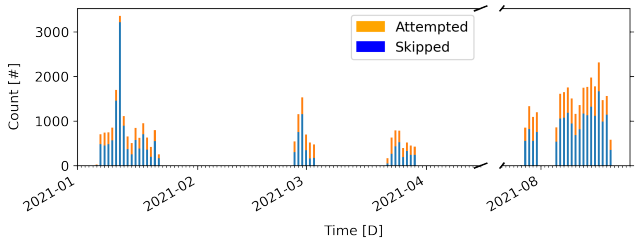


Figure 3: Number of activities per day. The download rate is limited to one download per URL per hour and repeated events are skipped.

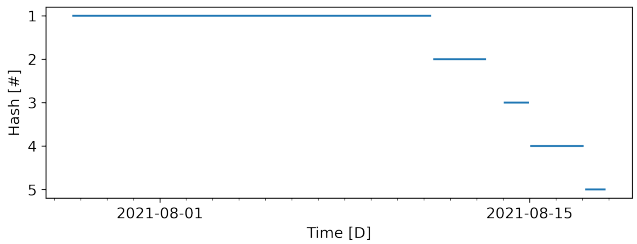


Figure 4: Availability of different executables from a URL. The availability is determined by the first and last successful download.

In Figure 3 the composition of attempted and unattempted download events in the activity log is shown. The share of unattempted downloads is significantly higher every day. In the dataset, a tiny minority of seven scanner source addresses is responsible for almost half the activities. That is not due to the fact that those source addresses send a lot of different URLs in their payload, but that the number of packets received from them is very high. The other activities are split between 10632 other scanner source addresses. Furthermore, most of the activities (81.86%) were not attempted to download. The share of attempts by the seven top source addresses is very small. This explains why the share of unattempted downloads is so high. Five of the highly active scanners do not have any download attempts at all, and the other two only have a very small amount. This further indicates that the rate they are sending their payloads is very high, as they are in the time out of one hour most of the time. Furthermore, the URL that is sent very often by the top source addresses, is also distributed by other source addresses, which leads to the fact that some do not have a single download attempt.

4.2 Distribution ecosystem

URL reusage and availability. Out of the 7961 unique URLs from which we downloaded an executable, 11 (3 in period one and 8 in period two) were recorded that were reused to distribute different unique executables. Figure 4 illustrates the availability of different executables from a URL. Each executable has a start and end date which are determined by the first and last successful download. The executables were only downloaded from this particular URL. As visible in the Figure, the periods in which the executables are available never overlap. The time spans in which the URL provides an executable varies. Some were only available for one day, whereas others were for several weeks. All executables were downloaded with the same name and have a similar size (2 KB difference).

Infrastructure. The download process involves two servers: One server that initializes a contact with our system, and at some point sends a payload which contains a download URL. This server will be referred to as the *scanner* server. The second server, which will be called the *hoster* server, hosts the download URL and thus provides the executable. Both servers can be the same server, but do not have to be. In period one in all activities, 35.81% of the servers were identical. If only servers are considered from which we successfully downloaded an executable, the share rises to 52.35%. Throughout period two, 23.3% of the activities had the same server and 31.26% of the servers with a successful download.

The total number of unique download servers (7666) is almost as high as the number of unique URLs from which we downloaded a binary (7961). This indicates that most servers only serve a single URL–which we have seen. In total we recorded 7400 servers from which we only downloaded from one URL. 245 servers offered two URLs. The maximum is at 6 URLs by one single server. There are several possible explanations for that fact. First, the reuse of a server may not be a common practice. Second, as our dataset is not continuous and the total period is only half a year long, it must be

considered that either different URLs were not tracked, or the usage duration is longer than half a year. Furthermore, the distribution methods differ among different malware families. Peer–to–peer distribution, like it was used by the Mirai botnet, propagates itself. Infected machines are scanning for other machines that are vulnerable. As more machines get infected, more scanning activity would occur–even for the same address space. As a consequence, this propagation behaviour leads to traffic from a lot of different scanner and hoster servers. Additionally, as observed before, there are URLs which are reused to distribute different executables. Thus, a server can serve various malware's by using the same URL over again.

The location of the IP addresses was determined using the *GeoLite2* database.[2] See the appendix for a visualization of the distribution of the scanner and hoster source addresses. There is a significant difference between the periods. In Period one, the scanners are dominated by the Netherlands with almost 30%, followed by India and France with about 15%. Only some more countries have a share of over 5%. In contrast, period two is mainly made up of scanners from China and the US, which have a combined share of over 80%. China has not been in the top 20 at all before and the share of the US has increased by over 30%. Furthermore, there is a difference between hoster and scanner origins. Comparing the confirmed hoster addresses (see Figure 7, which are those from which we successfully downloaded an executable, and the scanner addresses (see Figure 5) shows that there are several origin countries which have a similar share in both scanner and hoster origins. On the other hand, Germany is the top hoster origin country in period two and has only a tiny share in the scanning activity. The unconfirmed hosters have a significant share of *nan* values. This was caused by a handful of malformed URLs. For example, the URL *http://%s:%d/Mozi.7* indicates that the wild cards *s* and *d* should be replaced with a source address and port. However, this was not performed, which leads to an unresolvable origin country. The comparison of the origin countries shows that scanning activities are performed from a variety of different origins (82 countries in total) but are mainly seen from several dominating origins. Furthermore, the hosters show a different distribution than the scanners, which indicates two separate infrastructures.

4.3 Basic binary analysis

Target. Throughout the collection period, a total of 241 unique executables were collected. The samples have different file formats. In table 1 the different file types and their share in the total amount of executables is displayed. Most files are ELF executable files, which is the standard binary format for Unix [16]. Table 2 shows the target architectures of the ELF files. The majority target the MIPS architecture, which is used in the embedded processor market. Furthermore, beside shell scripts, there are also several executables which are in plain ASCII text or HTML format.

Additionally, by performing a basic decoding operation from Unicode to ASCII text format, more samples could be transformed so that they are human–readable. In total we were able to access 68 samples. Table 3 shows the classification of all readable files. The

[2]https://dev.maxmind.com/geoip/geolite2-free-geolocation-data?lang=en

File Type	Absolute	Share
ELF	166	68.89%
Bourne-Again shell script	26	10,79%
POSIX shell script	11	4,56%
HTML document	9	3,73%
ASCII text	28	11,62%
unknown	1	0,41%
Total	241	100%

Table 1: Distribution of file types among the collected unique executables.

Target	MIPS	ARM	Intel 80386	x86-64
Absolute	130	32	2	2
Share	78,31%	19,29%	1,2%	1,2%

Table 2: Distribution of the architecture of the ELF files.

File Type	VT	Not VT	Total
ASCII text	12	16	28
Bourne-Again shell script	9	17	26
POSIX shell script	9	2	11
HTML document	2	0	2
unknown	1	0	1
Total	33	35	68

Table 3: Characterization by file type and knowledge of VT of all malware samples that could be decoded to ASCII format.

files are distinguished by the fact if they are known to VT or not and what file type they have. More than half of the files are unknown to VT. For all the files that are known to VT, the assigned threat categories are distributed as follows: 2 miners, 10 trojans and 15 downloaders. The other files that have been submitted to VT before are labelled undetected or type unsupported only, which indicates that those files, even though known to VT, were not classified by any vendors yet. Interestingly, many files labelled as trojans have almost as many labels as downloaders. This is conform with a manual inspection, which shows that these files usually perform downloads as well. Furthermore, our brief inspection showed that most files that are unknown to VT show a similar composition as the downloaders that are known.

The executables which are in ASCII text format could be read and inspected. Most of those samples show the structure of a shell script: navigating directories, downloading and executing further files. For example:

```
1 wget <url>/arm;
2 curl -O <url>/arm;
3 chmod 777 arm;
4 ./arm exploit;
5 rm -rf arm
```

Listing 1: Source code of a downloader script

The executables which have been analysed and submitted to VirusTotal before are labelled by VirusTotal. 237 threat categories changed comparing the initial and final dataset. Even though that does not necessarily mean that the most popular category changed, we use the most up–to–date labels. Each executable may have multiple different labels as assigned by the vendors. In total, the samples are labelled by VT as follows: 164 trojans, 15 downloaders and 2 miners. Sometimes multiple labels almost have the same amount of votes by the vendors. Furthermore, the labels are known to be inconsistent and sometimes inaccurate [13]. Thus, the labels must be regarded with caution.

All executables that were unknown to VT when the initial query was performed were still unknown at the time of the requery. The labels of the known executables changed on the other hand. On average the executables gained 1.7 labels as malicious and lost 2.1 labels as undetected. This shows that the known executables were analysed by more vendors who shared their data with VirusTotal.

Malware family classification

MalwareBazaar query results include a signature of a malware file, which is defined as the malware family. However, this attribute is not always available. In total, 71 samples of our dataset were known by MalwareBazaar. 35 files samples have a signature: 34 belong to the Mirai and one to the Gafgyt malware family. In order to further classify the samples, we apply YARA rules on our dataset and analyse the file names.

https://github.com/InQuest/awesome-yara

YARA. YARA is a tool that helps malware researches to identify and classify malware samples. File are classified based on patterns defined in a rule.

Rules are composed of two parts: strings definition and condition. The defined strings are used in the condition which determines the circumstances under which a file satisfies the rule.

Strings may be hexadecimal strings, text strings or regular expressions. Hexadecimal strings allow wild cards, jumps and alternatives which makes them a powerful raw byte sequence matching mechanism. Text strings and regular expressions on the other hand are used to match legible text.

In order to use YARA, rules must be obtained. Writing rules requires expertise and a dataset of samples upon which the rules are based. Furthermore, there are a number of publicly available rules sets which offer a good starting point to use YARA. For this work, the following rule sets were used:

YARA-Rules repository This project is a public rule set that was created by a group of IT Security Researchers to centrally collect rules. It is maintained by the community and often used in literature.[3]

Signature–Base This is the rule set that is used by the scanners *LOKI* and *THOR lite* It is the free version of a commercial, more extensive rule repository.[4]

Malpedia's YARA-signator rules Malpedia's rules are automatically generated based on a set of malware samples of Malpedia's malware database [4].[5]

ReversingLabs YARA Rules The rules of this project are written by threat analysts with a focus on precise rules and the aim to provide zero false-positive detections.[6]

Open-Source-YARA-rules This repository is a very large collection of free rules that the creator found across the Internet.[7]

InQuest YARA rules A rule repository that is maintained by network and cloud cybersecurity company.[8]

All rule sets were applied to the executables we collected. In total, 57.26% (138 unique executables) matched at least one rule. However, only three of the listed datasets contain rules that matched a sample: *Yara-Rules repository*, *Open-Source-YARA-rules* and *Signature–Base*.

The matched executable are 95 ELF files, 6 Bourne Again shell scripts, 10 POSIX shell scripts, 9 HTML files and 18 plain ASCII text files. On average, 17 rules matched per executable. However, most of the rules that did match are very generic rules. For example, rules such as *math_entropy_4* or *contains_base64* are not conclusive in terms of malware classification. More precise rules, such as *MAL_ELF_LNX_Mirai_Oct10_2*, only make up the minority of matches. Comparing the different rule sets, especially the large rule collections have a lot of matches. The *YARA–Rules repository*, which is often used in literature, resulted in many generic matches and no specific malware identifications. An investigation showed that most of the rules of this repository are rather old (3–4 years) which could be the reason for the bad performance on our very up–to–date dataset. The rule set of *Signature–Base* showed several specific matches. 13 samples matched *Mirai* rules. Two different variants were identified. Another rule which identifies files that abuse a vulnerability in the Traffic Management User Interface (TMUI)[9] matched five samples. Furthermore, there are 69 matches for UPX compressed files, which is a common obfuscation technique, and 36 matches for *Mozilla/5.0* hard coded user agent strings, which has been seen in a *Command and Control* (C2) malware setup.[10]

File names. VT provides all names that are associated with a file. Furthermore, a meaningful name is chosen, which is defined as "the most interesting name out of all file's names" [9]. Some file names are obviously inconclusive, such as *bin.sh*, *copy.sh* or when the name is just the hash of the file.

Considering all the file names that are provided by VT, two–thirds match the names under which we gathered the executable. A very dominant example are the *Mozi* executables, which have three

[3] https://github.com/Yara-Rules/rules

[4] https://github.com/Neo23x0/signature-base
[5] https://github.com/malpedia/signator-rules
[6] https://github.com/reversinglabs/reversinglabs-yara-rules
[7] https://github.com/mikesxrs/Open-Source-YARA-rules
[8] https://github.com/InQuest/yara-rules
[9] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-5902
[10] https://us-cert.cisa.gov/ncas/analysis-reports/ar21-048b

different extensions: *Mozi.m*, *Mozi.a* and *Mozi.7*. In our dataset, several unique executables were downloaded under different names which are different combinations of the *Mozi* names. If all *Mozi* names are considered a match regardless of their extension when comparing our names to the VT names, 78% of the names match. If further all inconclusive names of VT are ignored, 93% match. Therefore, we use our collected names for the classification.

Classifying our dataset by name results in 70 different groups. Our set of names also includes inconclusive names such as *arm7*, *mips* or *infect*. This fact looks as if our parsing of the name failed. However, some characteristics can still be derived. Out of 241 unique samples, 112 belong to the *Mozi* family. Furthermore, if our downloads are filtered by all *Mozi* name variants, 90.6% (8083 of 8914) of the performed downloads can be mapped to the *Mozi* executables. On the other hand, the share of *Mozi* is 15.55%, if the time outed downloads are considered as well.

5 CONCLUSION AND FUTURE WORK

This work analysed the data that is collected at the reactive network telescope *Spoki*. We gave an overview about the data and showed basic characteristics. A focus of the work is the distribution of the individual executables that were downloaded. Even though our dataset is not continuous, it is already clear that unique executables may have peak periods, in which they are seen very often. In order to model and follow the development of the distribution, a more stable dataset is required. Furthermore, it is shown that even though a lot of different samples are collected, prominent samples usually dominate the dataset. Especially the *Mozi* malware family was seen very often during the collection period. It is an IoT P2P botnet malware[11]. A first classification of the collected executables gave an idea of the different types of files that are collected. Most prominent are ELF executables and shell scripts.

The analysis in this work gave a first idea of the possibilities that our data offers. In order to enhance the work and offer more valuable insights, future work should consider the following aspects:

As for data collection, the following points offer enhancement possibilities. Even though the current dataset is not continuous, the change in the distribution behaviour of samples can already be estimated. Some samples, which are very present in one month, do have a significantly lower impact in other months. In order to deduce further steps, the data collection must be stabilized. Furthermore, we have seen malware executables that are used to download further files from a remote server. This relates to the work of Kwon et al. [14], which studies the distribution networks of malware. In future work, the subsequent downloads shall be performed in order to acquire further samples, derive the relationships of malware samples and gain further information about the malware's life cycle. Additionally, the collection of further metadata may be useful. For example, if the availability of an executable is of interest, periodically re-querying the URL(s) could show how long the executable is available from a URL or if the URL is reused for distributing a different executable.

The analysis of the samples could be improved with the following aspects. Depending on the file type, different analysis methods can be adopted. For legible files, applying the URL extraction techniques which we used in the payloads can help to discover further URLs. For other files, such as ELF executables, a static analysis tool can help to parse and extract further information from those binaries. We are currently distinguishing the malware samples using their hash value, which is calculated using the SHA256 algorithm. This method does not consider that reobfuscation and repackaging is an established practice. Thus, there is the possibility that the actual set of different malwares is smaller than we assume. To address the classification problem, machine learning approaches are becoming popular [12, 20] in which neural networks are trained to identify malware samples. Furthermore, an established practice is the use of the YARA tool. The possibilities of YARA were shown in this work by a first classification using different rule sets. In order to further improve the performance, a solid rule set must be created. This does not necessarily mean writing custom rules, but rather collecting publicity available rules. As a starting point, the *awesome–YARA repository*[12] provides an extensive list of free YARA rule repositories. First, the performance of the available rules should be evaluated. Then a subset of rules should be created based on the relevance of the rules for the samples that we are collecting. Furthermore, a regular update strategy must be determined, as rule repositories are often updated and may provide further useful rules.

In summary, this work showed first characteristics and facts about the collected data and unveils improvement possibilities and analysis approaches for future work.

REFERENCES

[1] abuse.ch. [n.d.]. MalwareBazaar. https://bazaar.abuse.ch/. Accessed: 2021-10-29.

[2] Omar Alrawi, Charles Lever, Kevin Valakuzhy, Ryan Court, Kevin Snow, Fabian Monrose, and Manos Antonakakis. 2021. The Circle Of Life: A Large-Scale Study of The IoT Malware Lifecycle. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Virtual Event, 3505–3522. https://www.usenix.org/conference/usenixsecurity21/presentation/alrawi-circle

[3] Manos Antonakakis, Tim April, Michael Bailey, Matthew Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. 2017. Understanding the Mirai Botnet. In *Proceedings of the 26th USENIX Conference on Security Symposium* (Vancouver, BC, Canada) *(SEC'17)*. USENIX Association, USA, 1093–1110.

[4] Felix Bilstein and Daniel Plohmann. 2019. YARA-signator: Automated generation of code-based YARA rules. *The Journal on Cybercrime & Digital Investigations* 5, 1, 1–13. https://doi.org/10.18464/cybin.v5i1.24

[5] Jinchun Choi, Afsah Anwar, Hisham Alasmary, Jeffrey Spaulding, DaeHun Nyang, and Aziz Mohaisen. 2019. IoT Malware Ecosystem in the Wild: A Glimpse into Analysis and Exposures. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing* (Arlington, Virginia) *(SEC '19)*. Association for Computing Machinery, New York, NY, USA, 413–418. https://doi.org/10.1145/3318216.3363379

[6] Uli Harder, Matt W. Johnson, Jeremy T. Bradley, and William J. Knottenbelt. 2006. Observing Internet Worm and Virus Attacks with a Small Network Telescope. *Electronic Notes in Theoretical Computer Science* 151, 3 (2006), 47–59. https://doi.org/10.1016/j.entcs.2006.03.011 Proceedings of the Second International Workshop on the Practical Application of Stochastic Modeling (PASM 2005).

[7] Raphael Hiesgen, Marcin Nawrocki, Alistair King, Alberto Dainotti, Thomas C. Schmidt, and Matthias Wählisch. 2022. Spoki: Unveiling a New Wave of Scanners through a Reactive Network Telescope. In *Proc. of 31st USENIX Security Symposium*. USENIX Association, Berkeley, CA, USA.

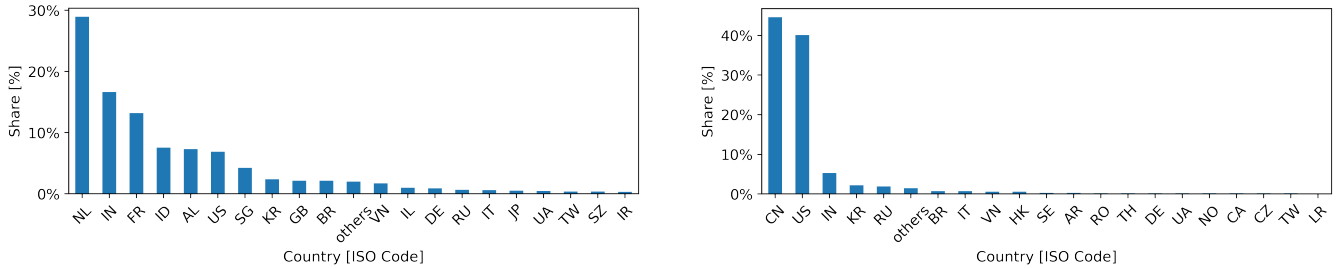
[8] Danny Yuxing Huang, Maxwell Matthaios Aliapoulios, Vector Guo Li, Luca Invernizzi, Elie Bursztein, Kylie McRoberts, Jonathan Levin, Kirill Levchenko, Alex C. Snoeren, and Damon McCoy. 2018. Tracking Ransomware End-to-end. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, San Francisco, CA, USA, 618–631. https://doi.org/10.1109/SP.2018.00047

[11] https://malpedia.caad.fkie.fraunhofer.de/details/elf.mozi

[12] https://github.com/InQuest/awesome-yara

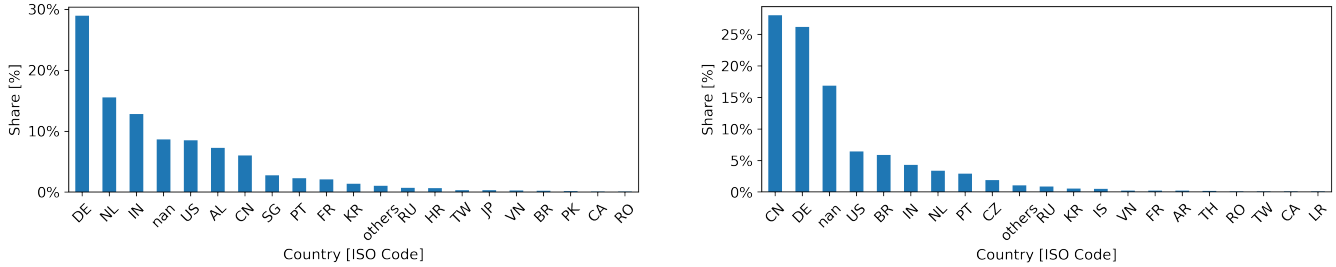
[9] Google Inc. [n.d.]. VirusTotal. https://www.virustotal.com. Accessed: 2021-09-19.
[10] Luca Invernizzi, Stanislav Miskovic, Ruben Torres, Christopher Kruegel, Sabyasachi Saha, Giovanni Vigna, Sung-Ju Lee, and Marco Mellia. 2014. Nazca: Detecting Malware Distribution in Large-Scale Networks.. In *21st Annual Network and Distributed System Security Symposium, NDSS*, Vol. 14. The Internet Society, San Diego, CA, USA, 23–26. https://www.ndss-symposium.org/ndss2014/
[11] Mattijs Jonker, Alistair King, Johannes Krupp, Christian Rossow, Anna Sperotto, and Alberto Dainotti. 2017. Millions of Targets under Attack: A Macroscopic Characterization of the DoS Ecosystem. In *Proceedings of the 2017 Internet Measurement Conference* (London, United Kingdom) *(IMC '17)*. Association for Computing Machinery, New York, NY, USA, 100–113. https://doi.org/10.1145/3131365.3131383
[12] Mahmoud Kalash, Mrigank Rochan, Noman Mohammed, Neil D. B. Bruce, Yang Wang, and Farkhund Iqbal. 2018. Malware Classification with Deep Convolutional Neural Networks. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, Paris, France, 1–5. https://doi.org/10.1109/NTMS.2018.8328749
[13] Alex Kantchelian, Michael Carl Tschantz, Sadia Afroz, Brad Miller, Vaishaal Shankar, Rekha Bachwani, Anthony D. Joseph, and J. D. Tygar. 2015. Better Malware Ground Truth: Techniques for Weighting Anti-Virus Vendor Labels. In *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security* (Denver, Colorado, USA) *(AISec '15)*. Association for Computing Machinery, New York, NY, USA, 45–56. https://doi.org/10.1145/2808769.2808780
[14] Bum Jun Kwon, Jayanta Mondal, Jiyong Jang, Leyla Bilge, and Tudor Dumitraş. 2015. The Dropper Effect: Insights into Malware Distribution with Downloader Graph Analytics. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (Denver, Colorado, USA) *(CCS '15)*. Association for Computing Machinery, New York, NY, USA, 1118–1129. https://doi.org/10.1145/2810103.2813724
[15] Giovane C.M. Moura, Ricardo de O. Schmidt, John Heidemann, Wouter B. de Vries, Moritz Muller, Lan Wei, and Cristian Hesselman. 2016. Anycast vs. DDoS: Evaluating the November 2015 Root DNS Event. In *Proceedings of the 2016 Internet Measurement Conference* (Santa Monica, California, USA) *(IMC '16)*. Association for Computing Machinery, New York, NY, USA, 255–270. https://doi.org/10.1145/2987443.2987446
[16] R. O'Neill. 2016. *Learning Linux Binary Analysis*. Packt Publishing, Birmingham, United Kingdom.
[17] Sergio Pastrana and Guillermo Suarez-Tangil. 2019. A First Look at the Crypto-Mining Malware Ecosystem: A Decade of Unrestricted Wealth. In *Proceedings of the Internet Measurement Conference* (Amsterdam, Netherlands) *(IMC '19)*. Association for Computing Machinery, New York, NY, USA, 73–86. https://doi.org/10.1145/3355369.3355576
[18] Peng Peng, Limin Yang, Linhai Song, and Gang Wang. 2019. Opening the Blackbox of VirusTotal: Analyzing Online Phishing Scan Engines. In *Proceedings of the Internet Measurement Conference* (Amsterdam, Netherlands) *(IMC '19)*. Association for Computing Machinery, New York, NY, USA, 478–485. https://doi.org/10.1145/3355369.3355585
[19] Marcos Sebastián, Richard Rivera, Platon Kotzias, and Juan Caballero. 2016. AVclass: A Tool for Massive Malware Labeling. In *Research in Attacks, Intrusions, and Defenses*, Fabian Monrose, Marc Dacier, Gregory Blanc, and Joaquin Garcia-Alfaro (Eds.). Springer International Publishing, Cham, 230–253.
[20] Danish Vasan, Mamoun Alazab, Sobia Wassan, Hamad Naeem, Babak Safaei, and Qin Zheng. 2020. IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. *Computer Networks* 171 (2020), 107138. https://doi.org/10.1016/j.comnet.2020.107138

APPENDIX



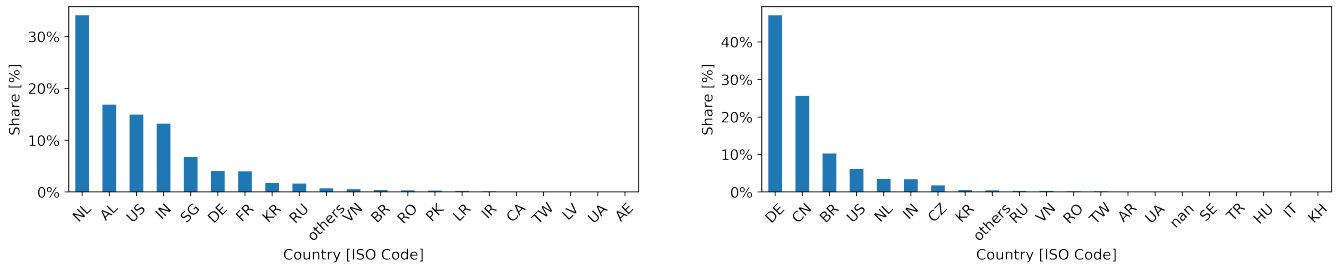
(a) Period one (Jan–Mar 2021). (b) Period two (Jul–Aug 2021).

Figure 5: Top 20 origin countries of the scanner source addresses of all recorded activities. The remaining countries are aggregated into "others". Note the different y–axis scales.



(a) Period one (Jan–Mar 2021). (b) Period two (Jul–Aug 2021).

Figure 6: Top 20 origin countries of all hoster source addresses of all recorded activities. The remaining countries are aggregated into "others". Note the different y–axis scales.



(a) Period one (Jan–Mar 2021). (b) Period two (Jul–Aug 2021).

Figure 7: Top 20 origin countries of the hoster source addresses from which a successful download could be performed of all recorded activities. The remaining countries are aggregated into "others". Note the different y–axis scales.