jet INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2024, VOL. 70, NO. 4, PP. 1099–1104
Manuscript received October 3, 2024; revised October 2024. doi: 10.24425/ijet.2024.152512

Analyses of malicious software long term activity - a case study

Krzytof Cabaj, Witold Wysota, Konrad Grochowski, and Piotr Gawkowski

***Abstract*—The paper describes the approach, instruments, and their evolution over a prolonged investigation of data collected by a honeypot system. The data is focused on network activity of a cybersecurity threat, in particular, attacks and activity throughout last five years of bots belonging to Smominru botnet. Conducted analyses include, but are not limited to, IP addresses used during attacks, day by day activity and evolution of malicious executables distributed over the observation period. The presented results also contain behavioural analysis of the threat and attack sources. Moreover, the paper details the systems used for data acquisition, their modifications along the observations made and all the tools developed to achieve the results.**

***Keywords*—HoneyPots; Dionaea; malware analysis; cybersecurity; Smominru botnet**

I. Introduction

THIRTY years ago Bill Cheswick described a HoneyPot system, used to observe an attacker who was compromising computers in Computer Science Research division of AT&T Bell Laboratories [1]. Still, after all these years, HoneyPot systems provide valuable information concerning attackers' activity. This paper discloses our findings with five years of observation of worm activity targeted at Microsoft SQL Server (MSSQL). In this case, a complex infrastructure used during a massive attack has been discovered. Long observation span required that the monitoring system had to be adapted to changes in attackers' behaviour. These in turn were probably the result of evolution of the security ecosystem. This is a good example of how "arms race" between attackers and security business nowadays looks like.

The paper is structured as follows. Section II describes HoneyPot systems and defines the scope of research the paper focuses on. Next, the tools used, developed, or adapted to transform collected raw data into artefacts are presented (Section III). These artefacts and statistical data can be further analysed to trace the behaviour and development of the threat. The main part of the paper, Section IV, presents the results of the research, focusing on different aspects of malware activity, malicious payload and the origin of the attackers. The paper is summarised in Section V.

K. Cabaj, W. Wysota, K. Grochowski, P. Gawkowski are with Faculty of Electronics and Information Technology, Warsaw University of Technology, Warsaw, Poland (e-mail: {krzysztof.cabaj, witold.wysota, konrad.grochowski, piotr.gawkowski} @pw.edu.pl).

II. HoneyPot systems

HoneyPots are decoy systems that trick malicious actors into believing they are interacting with a real production-deployed computer system. When attacking software tries to exploit vulnerabilities in the system to gain access to the target, its actions are recorded.

They can be classified based on their purpose and level of interaction [2]. Starting with low-interaction that are simple and safe (from the perspective of HoneyPot owner), through medium-interaction (where emulated software is more complex), up to high-interaction ones that contain real operating system the attacker can interact with (which allows gathering more information). It has to be considered that high-interaction HoneyPots are at the same time more risky (and require more maintenance effort) than medium- and low-interaction ones. On the other hand, the latter two types of HoneyPot are less likely to be deeply exploited by sophisticated malware than the high-interaction type, because the attacker can figure out that it is a decoy and stop the operation. This leaves low-interaction solutions as a good choice for capturing worms that rely on targeting a large volume of hosts in hope of infecting as many systems as they can.

There are different HoneyPot implementations available, both free and commercial ones, that are more or less sophisticated and allow easy management and deployment of instances. Some of them were described in [3]. HoneyPots usually simulate a range of network services and can be used to collect data related to different kinds of exploits. Studies exist that describe analysis of data collected across different services. An example of such is [4], where almost 1 million events is analysed across 5 months and correlation between different events (origin and time-wise) is explored.

In this work, rather than investigating a range of services, the focus is on long-term analyses of a worm exploiting a single service — MSSQL running on Windows systems. Unlike other studies, for example, [5], [6], the research in this paper is based on a much larger set of data. From Q4 of 2019 until Q3 of 2024 over 2.31 million of attacks were registered. However, this paper considers only 45 thousand attack cases gathered in that period. The authors attribute them to the Smominru Botnet [7].

III. Toolset

All the raw research data in the analyses given further were captured by Dionae HoneyPot instance deployed by the

```
stream = [
('in',  b'\x12\x01\x00 ... \x08\x00\x00'),
('out', b'\x04\x01\x00 ... \x02\x00\x00'),
...
('in', b' ... declare @a ...;exec(@a);'),
 ...
('in', b' ... exec(@a);\x27;exec ... '),
...
]
```

Fig. 1. HoneyPot raw data extract example ("..." denotes cropped data)

authors. Due to vast amounts of registered data, their manual analysis was impossible. For this reason, some custom Python scripts were developed and used for exhaustive analysis of the gathered raw data. In the following sections the detailed description of utilised tools is presented.

A. *Dionaea HoneyPot*

Dionaea HoneyPot is a low-interaction HoneyPot and does not host real vulnerable software but can simulate different protocols, including FTP, telnet, Microsoft SMB, MySQL, and more [5], [6]. One of its features is that after a few failed attempts it allows to login to password protected services with any password. For that reason, it is possible to gather passwords used during dictionary attacks, as well as to encourage the attacker to send more commands.

For security reasons, the instance of Dionaea used in this research is deployed on a rarely used hardware architecture. Due to this fact, some libraries used by the system had to be ported to the chosen architecture. Data gathered by the HoneyPot were stored to the file system and analysed by custom Python scripts described in the following section.

B. *Python Scripts for SQL Attack Analysis*

Data collected by the HoneyPot are delivered in form of *streams*. Each of them contains all the data exchanged during a single network connection to the HoneyPot and are stored in dedicated files. These files are stored in one folder per day, each in one folder per year-month pair, and finally in one folder per year. Such grouping enables a convenient inspection of the gathered data and speeds up some disk operations on a large data sets. A file name itself encodes the data and origin (in the form of a remote IP address) of the captured activity, as well as the port and the name of the service simulated by the HoneyPot (e.g. `mssqld-1433-210.4.123.XYZ-hAsH`).

Figure 1 presents an example of the file contents of such a stream (cropped in some places due to its size). The data format uses a list of Python binary strings. This provides human readability, which allows for easy inspection of the data. Each item in a *stream* is marked either *in* or *out*. The outgoing data are the HonyPot responses, usually the same execution confirmation message repeated after each incoming data. The attack would not continue if no response were produced, but although necessary, the responses themselves provide no additional information for the analysis. The incoming data are the exact bytes sent to the HoneyPot port. They can contain both MSSQL protocol bytes and SQL commands in the form of ASCII characters. Not every item in the stream contains an SQL command, but those that do usually include more than one (separated by semicolons – see Fig. 1).

The structure of the data was one of the driving forces of the development of the custom analytic scripts. The second one was the authors' understanding of the attack, growing with each iteration of the tools and observations. The tools were initially created when binary data encoded as hexadecimal strings were observed embedded in some SQL commands. This first version was a simple prototype for filtering of interesting SQL commands, extracting and decoding those strings. Python was selected as a natural environment for prototyping data manipulation, especially when all the processing could be performed offline and execution speed was not a top priority, rather the ability to quickly add more layers to the analysis.

The extracted data were called *blobs* and the first analysis was focused on establishing the relationship between dates, IP addresses and blobs used. This led to an early observation, that even a single IP uses multiple different blobs, but most addresses in the same period used a similar set of blobs. This suggested, that different blobs might not be separate payloads, but rather part of a more complex, multi-stage attack. Scripts were extended to preserve the connection between blobs obtained from the same stream and their order in the stream itself. The next assumption was that shorter streams were not examples of a different kind of attack, but samples of an attack that for some reason ended prematurely. That assumption was used to further group the attacks, leading to discovery of *sequences* that could be used to tag the various stages of the attack evolution.

All scripts produce data in a form of human-readable textual reports (for easy human inspection) and JSON (for convenient feeding the data into following stages of the analysis). Finally, the scripts also produce obtained blobs in a binary form, so they could be used for further analysis.

IV. Results

The first attempt of an attack directed at MSSQL was recorded by HoneyPot system on 3 April 2017. This activity was observed with relatively high intensity for almost two years. Suddenly, it almost completely stopped around the middle of February 2019. During this period, more than 36,000 attacks were observed. The sudden drop of activity was rather a global trend[1]. However, on 7 October 2019 a new wave of attacks began. Figure 2 presents the activity during the analysed five years of worm activity.

The recorded and analysed 43,544 attacks are not distributed equally in time. From Q4 2019 until around Q2 2022 a slow decrease in worm activity was observed (from more than 100 to around 25 attacks per day). Then, for more than two years, the observed activity was less than 10 attacks a day. The interesting rise in activity, to around 15 attacks per day, could be observed in June and July 2024. However, after these two months, the average activity has decreased to the earlier

[1]see live data gathered by https://dshield.org/port.html?port=1433

frequency of less than 10 attacks per day. In the following sections we present the most valuable results from conducted analysis of gathered data.

A. Execs and Potatoes

Each observed attack instance begins with guessing a weak password for the Database Management System (DBMS) operated by MSSQL. Afterwards, a sequence of various SQL instructions are sent by the attacker to the database management system which in effect downloads a malicious executable and finally infects the machine. During analysis of SQL queries, two interesting code fragments were identified. Due to the usage of characteristic keywords they were nicknamed *execs* and *potatoes* (examples are presented in the Fig. 3 and 4, respectively).

In-depth analysis of *execs* code reveals that encoded string contains JavaScript program, which, when executed, would download next stage malicious executables. Twenty seven unique parts of such code were detected in the analysed period of time. The analysis of JavaScript code reveals an attempt to access *wpd.txt*, *kill.html*, *test.html*, and *v.sct* files, which are associated with Smominru botnet [8].

What is worth mentioning, the executables were observed to appear in unique sequences, which could be used to distinguish various versions of the botnet. During the conducted research, fourteen unique sequences have been discovered (see Table I). The table presents also the number of detected sequences (Full column), the number of partial sequences – which ended before the full sequence is received by the HoneyPot (Part column), and the total number of attempts (Total column). As can be seen in the table, some sequences appear very commonly, while some are rather rare. This shows the evolution of the threat – the attackers make multiple iterations of modifying the code and evaluating whether the change has increased the infection rate.

TABLE I
DETECTED *exec* SEQUENCES

ID	Sequence	Full	Part	Total
E1	1, 2, 2, 3, 4, 5, 6, 7, 8	121	397	518
E2	9, 10, 10, 9, 11, 4, 5, 6, 8	978	20	998
E3	1, 2, 2, 3, 4, 5, 6, 12, 8	15071	26	15097
E4	13, 14, 14, 15, 4, 5, 6, 16, 8	30	0	30
E5	17, 18, 18, 17, 11, 4, 5, 6, 8	10	0	10
E6	19, 20, 20, 21, 22, 23, 5, 24, 6, 25, 8	6	0	6
E7	1, 2, 2, 3, 4, 5, 6, 26, 8	2528	2	2530
E8	1, 2, 2, 3, 4, 5, 6, 27, 8	95	0	95
E9	1, 2, 2, 3, 27, 8	19138	46	19184
E10	2, 2, 3, 27, 8	18	8	26
E11	27, 8	85	6	91
E12	2, 3, 27, 8	7	0	7
E13	8	4575	0	4575
E14	3, 27, 8	9	5	14

Figure 5 presents *execs* sequence evolution, where the most prominent attacker sequences (i.e. E3, E7, E9, and E13) are clearly visible. Comparing this plot with the overall activity of the worm (Fig. 2) one can notice an important difference: after the rise of activity observed in June and July 2024, the *execs* has completely disappeared in analysed data. Careful analysis of data, shows that this element of infection was abandoned by attackers around 20 July. This event can be possibly explained by *potatoes* evolution analysis described below.

Similar analysis was applied to *potato* type of SQL code fragments. Analysis of the *CREATE ASSEMBLY* SQL command shows that it creates CLR[2] based executable. Nineteen unique executables of this type were detected during the analysed period. Once again, they were observed in unique sequences. Table II presents detected potato sequences and their cardinality.

TABLE II
DETECTED *potatoes* SEQUENCES

ID	Sequence	Full	Part	Total
P1	1	18527	0	18527
P2	2	6510	0	6510
P3	3, 4	12666	29	12695
P4	5, 6, 7	219	1	220
P5	5, 8, 6, 9	4234	80	4314
P6	10, 11, 12	129	3	132
P7	13, 14, 15	5	1	6
P8	13, 14, 16	215	0	215
P9	17, 18, 19	8	0	8

The evolution of *potatoes* has been presented in Fig. 6. The shape of the plot is similar to the one presented in Fig. 5. The plots are very similar and the only difference in behaviour can be noticed after the rise of activity in June and July 2024. In contrast to the activity of *execs* (which ceased completely after 20 July), *potatoes* appear in few variants (see P6-P8 sequences in Fig. 6). This is consistent with the observed changes in botnet's behaviour: decreasing number of different *execs* yields an increase in the number of *potato* types. The next section provides more details on the contents of *potato* payloads.

B. CLR executables analysis

The SQL command sample from Fig. 4, containing the *potato* type of encoded data, is the reason for the keyword used in this paper – it is derived from the assembly name used by the attackers themselves. The code sample suggests that *potatoes* are binary executables in a format compatible with CLR, to be injected into an MSSQL instance. This was verified and deemed true by inspecting header of the decoded files.

Table III provides basic information on the obtained CLR executables (the *potatoes* used in the attacks). The identifier used in the table matches the number in the sequence definition (e.g. sequence P3 contained executables of ID 3 and 4). This also explains why the last column – the date of the first observation of the given executable – contains groups of the same

[2]Common Language Runtime — Microsoft's execution platform for .NET languages' family (C#, F#, VB.NET)

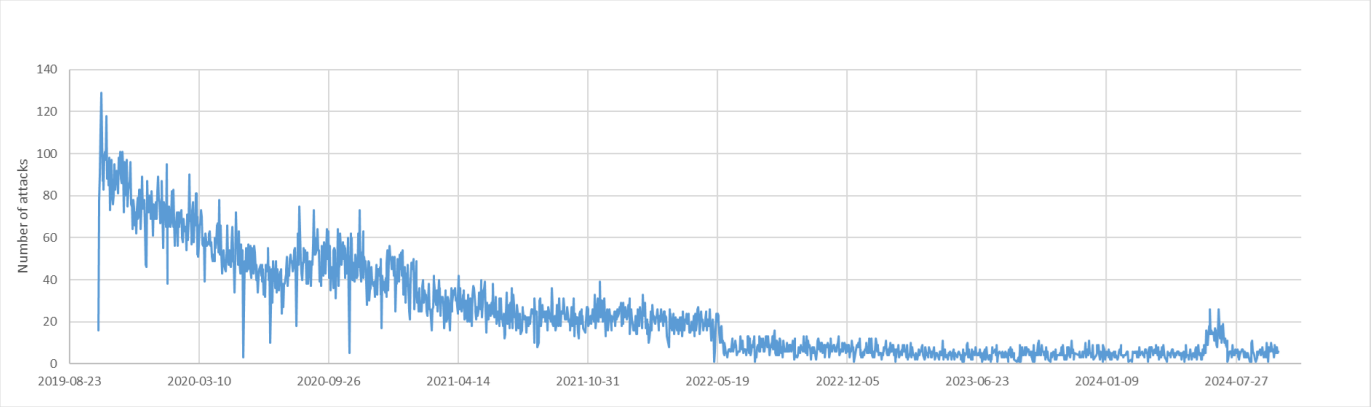

Fig. 2. Observed worm activity between 2019Q4 and 2024Q3

```
declare @a varchar(8000);set @a=0x44
45434C41524520406A733120696E743B4558
454320737O5F4F414372656174652027536
72697074436F6E74726F6C272 ... ;exec(@a);
```

Fig. 3. Malicious SQL code containing exec keyword

```
CREATE ASSEMBLY [SweetPotatoClr20]
AUTHORIZATION [dbo] FROM x4D5A90000
3000000040000000FFFF0000B80000000000
. . .
0000000 WITH PERMISSION_SET = UNSAFE;
```

Fig. 4. Malicious SQL code containing potato keyword

dates: all executables from the same sequence were observed together. The hash column contains MD5 checksum of the binary, which can be used to identify it in online resources like VirusTotal[3], without the need to propagate malicious software. All obtained samples were compared with VirusTotal database of known threats, which provides a comprehensive amount of information. Two interesting metrics were included in the table: the number of security vendors that flagged the sample as malicious (over total vendors queried) and the date of the first submission to the database. Marked entries denote samples first uploaded to VirusTotal as a result of this analysis. It is worth noticing, that 13 out of 19 samples (68%) were first analysed during this research and those seem more difficult to properly flag as malicious by security software. Dates for remaining entries can be treated as indicative data suggesting that the attacks described in this paper were observed by others in similar time frames.

The name and initial behavioural analysis provided by VirusTotal suggests, that the malicious codes might be variants of SweetPotato Windows privilege escalation tool, developed by penetration testers and described in [9]. Yet, the first recorded use of a *potato* in the observed attacks was recorded in October 2019 – six months before publication of the tool code in the GitHub repository in April 2020 (or three years if taking into the account VirusTotal submission date). This observation requires further research. Apart from the analysis of the payloads used during the attack, the research also focused on the origins of the attacks. The results of this analysis are described in the following section.

TABLE III
OBTAINED CLR EXECUTABLES

ID	Hash	Virus Total		First Observed
		Flags	Submitted	
1	2f1aecbdb7ffcb0016de8ab734c0de44	51/73	2017-05-15	2019-10-07
2	130d2b07a1c4cde8f0804df9fa9622d4	55/75	2021-03-06	2020-07-26
*3	7c253e9c5d1b5f562866b92d64499552	32/72	2024-10-07	2020-12-07
*4	908481721f2f150589337023f44f0393	32/72	2024-10-07	2020-12-07
5	b41633dc589b37d7894240f5c48ef332	53/72	2022-05-19	2022-05-25
6	77c9692543c3c370ae46f72718a11d4e	52/73	2022-04-29	2022-05-25
*7	224c244f6cab736846a3e1c199d670d5	17/71	2024-10-07	2022-05-25
8	32b4d14563d320a0912a94a54877aa54	46/72	2022-07-25	2022-06-01
*9	d6d437b32437ad26ea8fb7660d546f01	19/72	2024-10-07	2022-06-01
10	c08a4df2c1a925b79a875ed4b90fe09f	5/73	2024-08-15	2024-07-20
*11	539ebc99c7cdff52d338672dcbb11ad8	10/72	2024-10-07	2024-07-20
*12	3cd878f8d0f638bd6eebc834ffd1f43d	8/72	2024-10-07	2024-07-20
*13	9c30af755203a50ad4c800a298469928	17/72	2024-10-07	2024-08-20
*14	9a911eeff01d9910e9745a2ee388048b	11/72	2024-10-07	2024-08-20
15	62b531a9a2d935c0ee9129bb64603bca	33/73	2024-09-08	2024-08-20
*16	c246bb43e26df88e2a112fdc39e14e43	23/72	2024-10-07	2024-08-21
*17	adc9db5f9f0a7e329fbb6ed134b45ce9	17/72	2024-10-07	2024-09-29
*18	d7ee11c8c6bcc8ea6bd17636e6b3bba6	12/72	2024-10-07	2024-09-29
*19	13b8d0dd0b249bd2de9a8e0c757c5ef4	17/72	2024-10-07	2024-09-29

C. Attackers IP addresses analysis

Having recorded so many network attacks, their geographic origins were analysed. First, occurrences of given IP network addresses were counted for all attacks. Then, a simple Python script to assemble a database of mappings between IP addresses and geographical information obtained via IP-WHOIS.IO[4] was developed. That data were then further

[3] https://www.virustotal.com

[4] https://ipwhois.io/, API: http://ipwho.is

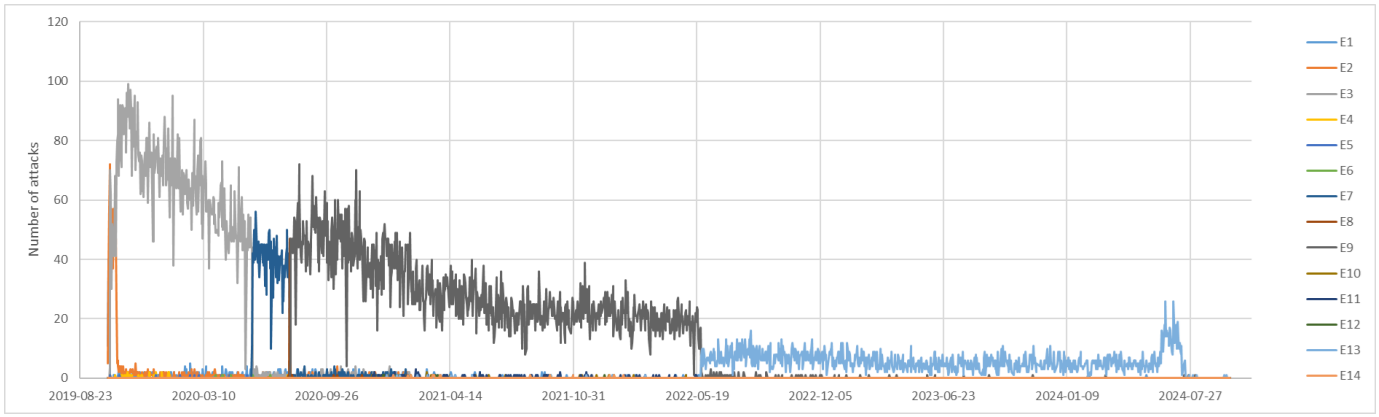

Fig. 5. Detected exec sequences from 2019Q4 to 2024Q3

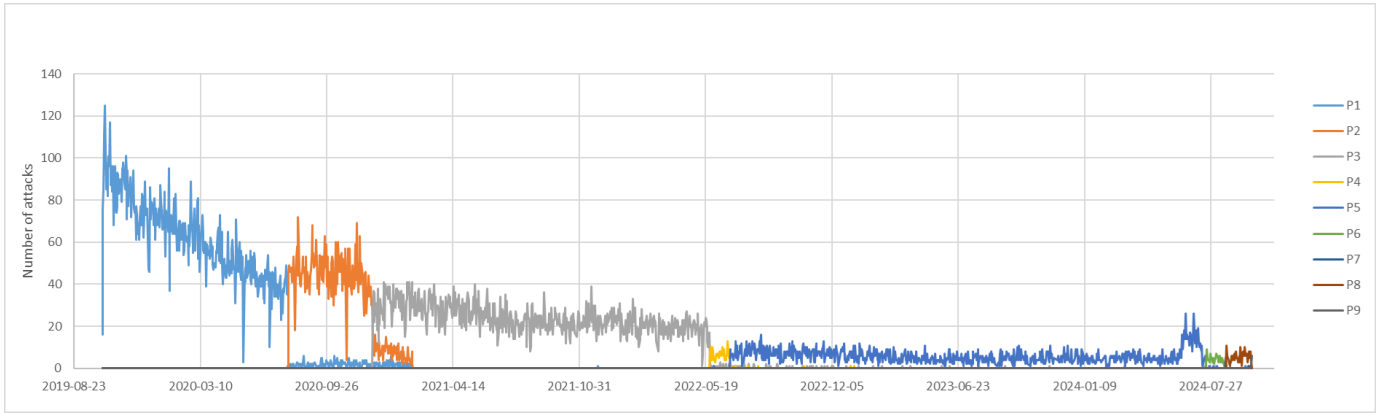

Fig. 6. Detected potatoes sequences from 2019Q4 to 2024Q3

processed by more Python code to obtain network information about a particular offender and its country of origin.

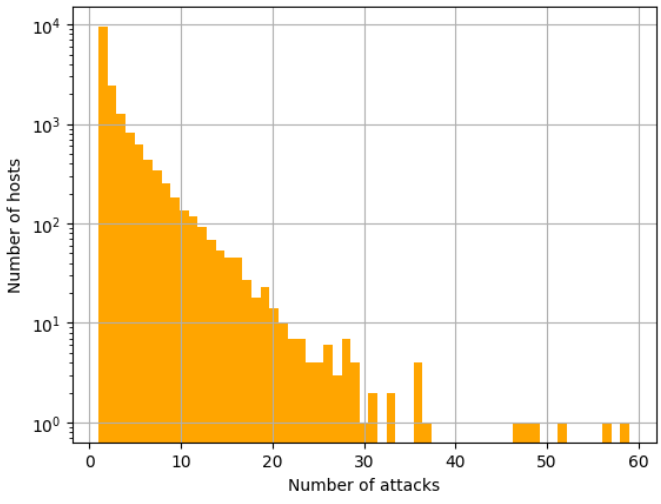

Fig. 7. Count of IP addresses conducting a given number of attacks

The majority of hosts performed only a few attacks each. In the analysed dataset, only 16 attackers are responsible for making 30 or more attacks each, contributing to the total of 659 incidents, which makes only 1.49% of all attacks. Ten of those attackers were within networks attributed to ASNs[5] for networks located in China. A histogram of the number of attacks performed per IP is presented in Fig. 7. It is clearly visible that most hosts have conducted an attack only once or twice within those 5 years. It is unclear whether those addresses that are the source of attacks more frequently are really the same attackers or rather public IPs of private network gateways. That could be a field of further research.

GeoIP analysis of the data shows that most of the attacks originated in China (slightly over 50%, not including Hong Kong) with the United States of America, Brazil, and Russia trailing far behind. The distribution of incidents across countries of origin is presented in Table IV (top 10) as well as in Fig. 8 (all). Since China is the leading area of the source of attacks, a more fine-grained information on that is shown in Fig. 9. Most incidents come from the Beijing area (20% of all incidents from China) as well as other developed regions of China on the south-east coast. It is worth noting that these results are consistent with the attack sources of Smominru botnet stated in [8].

[5] *Autonomous System Number*; Autonomous Systems are large networks with a single routing policy

TABLE IV
COUNTRY OF ORIGIN (TOP 10)

Country	Number of attacks	Percentage
China	22061	50.66
USA	2382	5.47
Brazil	1399	3.21
Russia	1321	3.03
India	1081	2.48
Hong Kong	1067	2.45
Mexico	1020	2.34
Korea	943	2.17
Taiwan	891	2.05
Indonesia	746	1.71

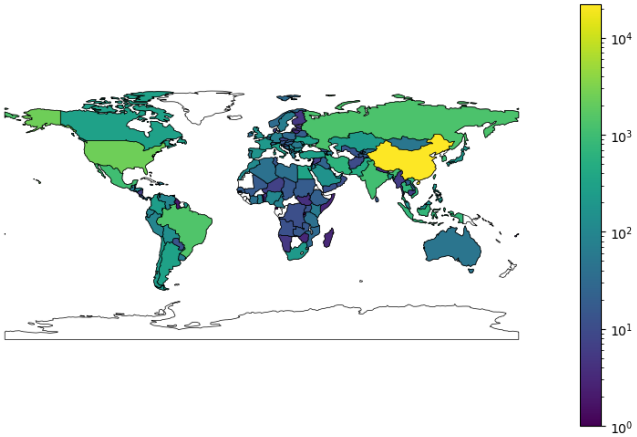

Fig. 8. Countries of origin by attack count

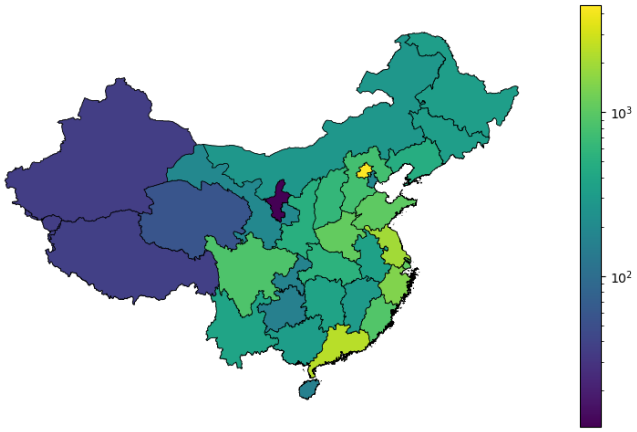

Fig. 9. Sources of attacks originating in China

V. CONCLUSIONS

The recorded five years of Smominru botnet activity is a huge set of data to dive into and explore. It can be analysed in many ways to discover different malware aspects and properties. Captured raw data includes the attack vector (in this case, the set of passwords used to get into the system). One can also get detailed scenarios of commands the malware sends to MSSQL for further system exploiting.

An interesting observation is the evolution of executables the malware tries to run on the affected system. Moreover, these executables are injected in some sequence that can be used for identification of subsequent botnet versions. It is sad that even after plenty of time many antivirus systems still do not recognise them as malicious.

Each registered attack unveils more information about the worm itself. This allows for multilayered analysis of the threats – starting from technical nuances of the exploits, up to geographical distribution of the infected machines. Examples of such various analyses are presented in this paper. Yet they do not exhaust the research possibilities offered by the already gathered data.

Further research can be focused on dynamic analysis of gathered executables to explore their evolution (e.g. system resource usage, communication patterns, file system operations). However, it is worth noting that even static low-cost data sources like HoneyPots require dynamic approach to data analysis. Continuous analysis of the malware using HoneyPots is indisputably beneficial but still challenging task.

REFERENCES

[1] B. Cheswick, "An evening with berferd in which a cracker is lured, endured, and studied," in *In Proc. Winter USENIX Conference*, 1992, pp. 163–174.

[2] I. Mokube and M. Adams, "Honeypots: concepts, approaches, and challenges," in *Proceedings of the 45th Annual ACM Southeast Conference*, ser. ACMSE '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 321–326. [Online]. Available: https://doi.org/10.1145/1233341.1233399

[3] W. Ahmad, M. A. Raza, S. Nawaz, and F. Waqas, "Detection and analysis of active attacks using honeypot," *International Journal of Computer Applications*, vol. 184, no. 50, pp. 27–31, Mar 2023. [Online]. Available: https://ijcaonline.org/archives/volume184/number50/32645-2023922624/

[4] E. Vasilomanolakis, S. Karuppayah, P. Kikiras, and M. Mühlhäuser, "A honeypot-driven cyber incident monitor: lessons learned and steps ahead," in *Proceedings of the 8th International Conference on Security of Information and Networks*, ser. SIN '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 158–164. [Online]. Available: https://doi.org/10.1145/2799979.2799999

[5] V. Sethia and A. Jeyasekar, "Malware capturing and analysis using dionaea honeypot," in *2019 International Carnahan Conference on Security Technology (ICCST)*, 2019, pp. 1–4.

[6] K. Saikawa and V. Klyuev, "Detection and classification of malicious access using a dionaea honeypot," in *2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 2, 2019, pp. 844–848.

[7] C. Cimpanu, "Smominru botnet infected over 500,000 windows machines," *Bleeping Computer*, vol. 1, 2018.

[8] "The massive propagation of the smominru botnet," https://www.akamai.com/blog/security/the-massive-propagation-of-the-smominru-botnet, accessed: 2024-10-10.

[9] C. Coburn, "Sweetpotato – service to system," Apr 2020. [Online]. Available: https://www.pentestpartners.com/security-blog/sweetpotato-service-to-system/