Analysis and Characterization of IoT Malware Command and Control Communication

Đorđe D. Jovanović and Pavle V. Vuletić

Abstract — **The emergence of Mirai botnet in 2016 took worldwide research teams by surprise, proving that a large number of low-performance IoT devices could be hacked and used for illegal purposes, causing extremely voluminous DDoS attacks. Therefore, a thorough inspection of the current state of IoT botnets is essential. In this paper, we analyze the dynamic behavior and command and control channels of two classes of IoT botnets, Mirai and Gafgyt. Based on collected information, a comparative analysis and key phases of botnet communication is provided. Such an analysis will serve as a basis for smart botnet detection mechanisms.**

Keywords — **Botnets, CnC communication, IoT.**

I. Introduction

BOTNETS represent a network of computers (bots) which are under the control of a malicious hacker - botmaster. Botmasters use the machines under their control for various types of malicious activities, such as: performing DDoS (Distributed Denial of Service) attacks, spreading ransomware, stealing personal information, unwanted digital currency mining, etc. [1] A botmaster communicates with infected computers via the Command and Control (CnC or C2) channel. By exploring command and control dynamic behavior patterns and creating a system that can efficiently discover botnet communication, it is possible to stop the bots in the earlier stages of the attack lifecycle. This is to prevent them from becoming activated by the botmaster and involved in the attack, and also to mitigate threats to the information security. Preventing botnet creation in a target network will not stop the attacks from another network against the target. However, widespread use of similar systems would increase the overall information security of the devices connected to the internet. The CnC channel is a single point of failure for the botnet and its detection, and mitigation would fully disable the control of the bots. CnC channels evolved in time, as well as their detection avoidance techniques, which became more and more sophisticated. Examples being: DNS (Domain Name Server) fluxing and DGA (Domain Generation Algorithm), mentioned in the next Section.

Paper received April 30, 2020; revised August 8, 2020; accepted August 18, 2020. Date of publication December 25, 2020. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Grozdan Petrović.

This paper is revised and expanded version of the paper presented at the 27th Telecommunications Forum TELFOR 2019 [6].

Đorđe Jovanović is a PhD student at the School of Electrical Engineering, University of Belgrade, Bul. kralja Aleksandra 73, 11120 Belgrade, Serbia (phone: -; e-mail: jd185001p@student.etf.bg.ac.rs).

Pavle V. Vuletić, is with the School of Electrical Engineering, University of Belgrade, Bul. kralja Aleksandra 73, 11120 Belgrade, Serbia (phone: +381-11-381465, e-mail: pavle.vuletic@etf.bg.ac.rs).

The Mirai botnet appeared in 2016. Its power was demonstrated by performing DDoS attacks against famous sites, such as Krebs on Security and OVH, as well as DNS providers Dyn and Lonestar Cell [2]. Infected devices included IP cameras and home routers, and their number surpassed 200.000. Since the IoT (Internet of Things) devices oft-times do not possess even the basic security features, they represent a weak spot for any computer network. This is the primary reason why researching IoT botnets, as well as their CnC communication, is paramount.

Although the peak of the first Mirai infection was in 2016, the malware which is based on the Mirai code still exists and is very active in building new botnet infrastructures. Recently, new botnets have appeared, built using a variant of Mirai with the addition of some recent exploits in the networking equipment [3]. Furthermore, new botnet malware appears every day. It is a common practice for the attackers, instead of writing the malware code every time from scratch, to reuse the code of a previous malware, make small changes, or even mix the features of multiple malware. For example, the recently discovered Dark Nexus botnet generating malware is built on top of Mirai and QBot code [4][5]. It is exactly this common practice, where hackers reuse malware code components for building new malware, which gives a full justification for the research on the dynamic behavior of existing malware. Analyzing the characteristic behavioral patterns can lead to the discovery of methods to mitigate the proliferation of many different versions of botnet generating malware through their detection.

The paper is organized as follows: in the Related Work section, an overview of existing approaches in botnet CnC communication analysis is given. This section is concluded with a possibility of extending existing approaches. In the Methodology section, a detailed description of system architecture, software tools used, and the CnC botnet communication analysis process is given. In the Results section, botnet communication patterns and their statistical parameters are analyzed. In the Conclusion section, using the results from previous section, several conclusions are drawn. This paper is the expanded version of a paper which was presented at the 2019 TELFOR conference [6].

II. Related Work

There are two most common approaches to the botnet CnC detection: packet header analysis and deep packet analysis.

Packet header analysis concerns the analysis of packet headers, and drawing conclusions about the nature of network traffic from them. One approach is to use certain

elements of the header, such as source and destination IP, in order to distinguish malicious traffic from normal traffic. Packets can be collected into conversations, flows or quadruplets defined as (source IP, destination IP, source port, destination port), and then analyzed and classified into malicious and normal conversations based on different approaches described below [3],[7],[8]. The other approach displayed in the research literature is to perform an n-gram analysis on domain name, collecting its statistics in order to detect anomalies[9],[10].

Deep packet traffic analysis involves analyzing packet payloads, searching for specific malware signatures and analyzing specific protocol traffic, such as DNS traffic, to detect anomalous behavior [11],[12]. Two main techniques that are used by botnets in order to hide CnC communication are Domain Generating Algorithms (DGA for short) and DNS-fluxing [9],[10],[13],[14]. DGA involves using multiple domains, and attaching them to the same IP address. DNS-fluxing involves using DNS records with short timeout (often less than 600 seconds), and changing the IP address of the CnC server each time the DNS record expires.

Most of the literature today implements machine learning or deep learning neural networks to distinguish normal from malicious network traffic. As an aid to this analysis, a technique such as whitelist of URLs (Uniform Resource Locators) and virus analysis tools are used. The data is gathered either from a well-known dataset, such as CTU-13[1] [3],[8] (Czech Technical University-13), or collected independently. Although this approach has advantages, such as identifying the key statistical traits of botnet traffic, it requires a reliable sample for it to work. CTU-13 dataset consists of old samples of malware communication, which are not relevant today.

In contrast to these approaches, our aim is to explore the dynamic behavior of the CnC channel of recent IoT malware families on a level which is finer grained than the packet header analysis, without going into deep packet inspection of packet payloads. Such an approach could provide additional parameters for the anomaly detection techniques, thus making them more reliable. The goal of this paper is to show the CnC behavioral patterns as seen by the typical network defense systems (intrusion detection and prevention systems) and to explore the potential of finding new methods for botnet activity detection and mitigation.

III. Methodology

In order to analyze IoT botnet behavior, we used two RaspberryPi 2B devices and two RaspberryPi 3 devices with Raspbian OS. We infected them with various malicious applications, which were downloaded in the period between June 15th and July 15th 2019. Devices mentioned were used because they are based on ARM architecture, since most of the IoT malware can be compiled for ARM. Two IoT malware families which are the most common today were analyzed: Mirai and Gafgyt. Eight Mirai variations were analyzed (ab.arm7, armv7l, kalon.arm, okane.arm, pandora.arm7, zehir.arm7, ntpdd.arm8, r4z0r.arm7), for which three DDoS attacks are recorded. Nine Gafgyt (cc9arm6, soul.arm6, eagle.arm, eagle.arm7 Demon.arm4, TacoBellGodYo.arm4, frag.arm, yakuza.arm6, assailant, eagle.arm7, eagle.arm7) applications are analyzed, for which five DDoS attacks are recorded. The analysis was performed on real devices in order to avoid malware polymorphism – the situation in which malware changes the behavior when it detects the execution in sandboxes or virtual machines [15]. RaspberryPi devices were connected to a router, which connected them to the internet. Network topology is shown in Fig. 1.

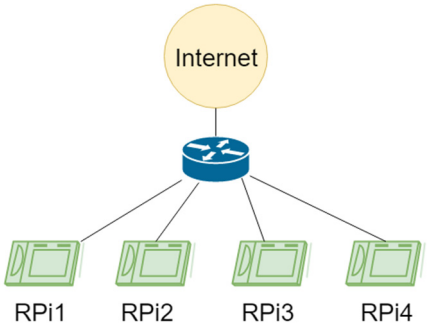

Fig. 1. Network topology.

All devices had public IP addresses and the only rule set on the devices was filtering the traffic to the local network in order to protect the rest of it from malware infection.

All the infected executables were collected from the URLHaus[2] database. This database collects URLs containing suspicious or malware files, and assigns tags to them. We also compared the dynamic malware behavior with the results of Cuckoo, a well-known free sandbox malware analyzer, and used static malware code analysis in order to fully capture malware properties. Packets were recorded using `tcpdump` on the Raspberry Pi devices. In order to filter out only botnet communication, in parallel with recording the malicious traffic, we captured and analyzed the traffic of non-infected machines for comparison. All non-local IP addresses in packet traces were manually checked for ownership and location using IP and DNS lookup tools.

In addition to performing botnet communication analysis manually, we performed a time series analysis on the collected data. Using the contents of the `pcap` files and extracting the message length in bytes, a time series with bin size of 1s was created, thereby counting the number of bytes passed during each second of network flow. Network flow is defined by the quadruplet consisting of: source IP address, destination IP address, source port and destination port. The unicast communication within the network is most often bidirectional. In such cases there will always be two network flows which belong to one conversation between the two devices. These two network flows will have the

[1] https://www.stratosphereips.org/datasets-ctu13

[2] https://urlhaus.abuse.ch/browse/

same but swapped values in the pairs (source IP address, destination IP address) and (source port, destination port). We will call such flow pairs related flows. Using the information from time series, the following parameters were extracted: the Pearson correlation coefficient between the two related flows, number of bytes per second in each flow and the difference of unidirectional flow throughput between the bins in two related flows. Also, for all the parameters, the following statistical measures were calculated for botnet and normal traffic, separately: minimum, maximum, average, and standard deviation.

The Pearson correlation coefficient is used as a measure of similarity between two related flows. Values of over 0.7 are considered to indicate strong correlation.

Unidirectional flow throughput is observed in order to compare how often, in comparison to normal traffic, do botnets communicate, as well as to provide an estimate on network load on the bot machine.

Difference of unidirectional flow throughput is used to indicate symmetric behavior between two directions in a flow. If the hosts pass a similar number of messages between them, this measure will be lower, which is expected in botnet traffic.

A. Mirai botnet

Mirai botnet family is an evolution of the BASHLITE botnet. It uses brute force methods to break in and infect a Linux-based machine. The virus spreads by scanning pseudo-random IP addresses on port 23/2323, avoiding the addresses in its blacklist (government and military institutions). Once it finds a valid IP address, it attempts to guess user credentials. If successful, the machine is infected, and Mirai proceeds with closing off other ports, lest no one else take control of the machine. Then the bot downloads the infected malware, and starts communication with the CnC server. Once the attack command is received, bot commences a DDoS attack against the victim.

B. Gafgyt botnet

Gafgyt botnet (also known as BASHLITE Lizkebab, Qbot, Torlus and LizardStresser) is a family of IoT malware, using bash vulnerabilities to infect a Linux-based machine, and issue DDoS attack commands against a victim.

IV. Results

In this section, malware applications analyzed are grouped according to their families, Mirai and Gafgyt, as well as statistical features of traffic flows belonging to botnet CnC traffic flows and normal traffic flows. Concerning the malware applications CnC communication analysis, first the general characteristics of each group are given. Then, each application is observed with respect to its particularities. The analysis is divided into three phases of communication: establishing connection with the CnC, maintaining the connection and attack command. As for the description of statistical features, a graph plotting the data points for each feature of both the botnet and normal traffic is shown. Afterwards, a deeper analysis of each feature is given.

A. Uninfected device traffic

Uninfected traffic on these devices is characterized by a large number of short TCP (Transmission Control Protocol) session attempts that begin with a SYN packet, initiated by a remote machine, and end with RST-ACK packet. This is a background random port scanning over various ports that can be observed on all machines connected on the internet. Uninfected device experienced also ICMP (Internet Control Message Protocol) ECHO requests. Similar behavior is filtered from the traces of the infected devices. This step was crucial, in order to filter out all communication that occurs without the presence of malware.

B. Mirai traffic

In the case of Mirai botnet, the connection is initiated by the infected machine using TCP handshake, during which it sent information about itself to the CnC. Malware used various TCP ports for the CnC communication (e.g. 3301, 1337, 1791) sometimes hiding its operation behind the well-known ports which pass through the firewalls (e.g. ntpdd.arm8, which used DNS port 53). There were no DNS queries for the duration of the infection. The connection is maintained either by the infected machine, or the CnC, by sending messages periodically, every 60s. The exact number of these messages varies, as well as their contents. Finally, the CnC server sends the attack command, via a PSH-ACK packet, with all the necessary parameters, to the infected machine. The bot responds with an ACK packet, after which the DDoS attack commences. For all of the variations, the establishing connection phase is the same as described earlier.

A diagram of Mirai CnC communication maintenance phase is given in Fig. 2. Data from the packet payload is enclosed in parentheses, strings are designated by quotation marks, while the hex code is designated by initial “0x”. The plus sign denotes string concatenation.

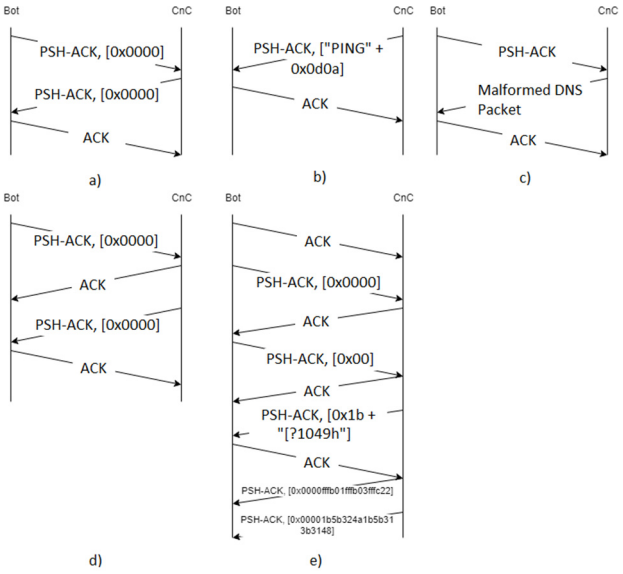

Fig. 2. Mirai CnC communication.

In Fig. 2, the subdivision is as follows: a) represents the ab.arm7, pandora.arm7 and kalon.arm malware, b) represents the armv7l malware, c) represents the ntpdd8.arm8 malware, d) represents the okane.arm and

r4z0r.arm7 malware, and e) represents the zehir.arm7 malware.

In the attack phase, the CnC first sends a packet with a command string, after which the attack begins. The command string can be of 2 types: plain-text and hex. For instance, in the case of "armv7l" malware, the CnC sends a PSH-ACK packet, with the following attack command:

```
".UDP 40.81.59.133 30142  20 32 0 10"
```

Taking a look at the command string, and comparing them to the captured traffic, UDP (User Datagram Protocol) protocol is used. Using static code analysis, a conclusion has been drawn that the other parameters, from left to right, are: target IP address, destination port, DDoS attack duration, option to use SOCK_DGRAM (denoting that a socket is used for a datagram-based protocol) or SOCK_RAW (denoting lack of processing by the network layers) for creating sockets, buffer size, and the number of attacks against a target. After the command string, the infected machine sends an ACK packet, and an attack occurs. This describes the case of the plain-text attack message.

In the case of "r4z0r" malware, the CnC sends a PSH-ACK packet, with the following attack command:

```
"00 12 00 00 00 1e 09 01 67 5f dd a7
20 01 07 02 38 30"
```

The attack string was written in hex form, and by converting the hex values into decimal and ASCII, the destination IP and port can be deciphered (0x 67 5f dd a7, which gives 103.95.221.167 for the destination IP; 0x38 30, which, by converting into ASCII gives 80 for the destination port). After the command string, the infected machine sends an ACK packet, and an attack occurs.

C. Gafgyt traffic

In the case of Gafgyt botnet, similarly to Mirai, the infected machine initiates a TCP session over various ports. Malware used various TCP ports for the CnC communication (e.g. 87, 17769, 58215). Botmaster is contacted directly over IP without using DNS resolution. After the connection has been established, the infected machine sends the CnC server a string describing the hardware and software environment in which the malware is running. The connection is maintained by the CnC server, by sending messages periodically, every 60s. The exact number of these messages varies, as well as their contents. Finally, the CnC server sends the attack command, via a PSH-ACK packet, with all the necessary parameters, to the infected machine. The bot responds with an ACK packet, after which the DDoS attack commences.

For all of the variations, the establishing connection phase is the same as described earlier.

A diagram of CnC communication maintenance phase is given in Fig. 3. Data is enclosed in parentheses, strings are designated by quotation marks, while the hex code is designated by initial "0x". A plus sign denotes string concatenation.

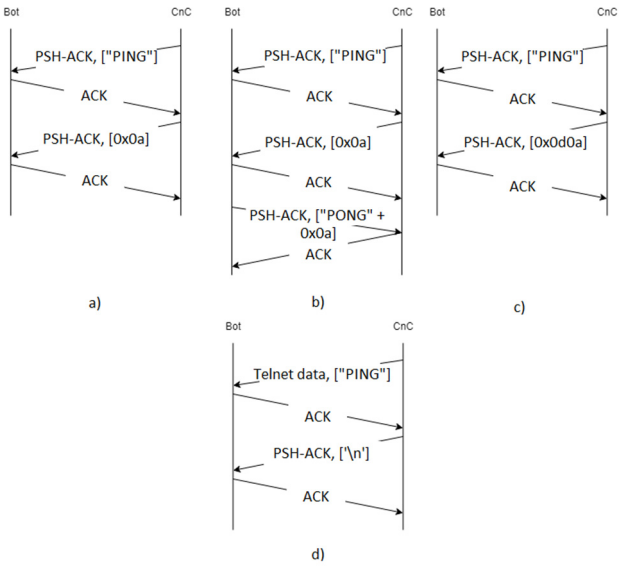

Fig. 3 Gafgyt CnC communication

In Fig. 3, the subdivision is as follows: a) represents the cc9arm6, Demon.arm4, eagle.arm and soul.arm6 malware, b) represents the eagle.arm7 and TacoBellGodYo.arm4 malware, c) represents the frag.arm malware, and d) represents the yakuza.arm6 malware.

In the attack phase, the CnC first sends a packet with a command string, after which the attack begins. The command string can be of 2 plain-text message types: plain-text and encrypted. For instance, in the case of "TacoBellGodYo" malware, the CnC sends a PSH-ACK packet, with the following attack command:

```
"!* STD 174.214.34.8 4370 300"
```

While in the instance of "eagle" malware, the attack command has the following format:

```
"!* UDP 13.83.247.215 30217 100 32 0 10"
```

Taking a look at the command string, and comparing them to the captured traffic, the first argument is the protocol used. In both of these cases, the UDP protocol was used for the attack. Using static code analysis revealed the use of the same syntax for attack description as in the case of Mirai: target IP address, destination port, DDoS attack duration, option to use SOCK_DGRAM or SOCK_RAW for creating sockets, buffer size, and the number of attacks against a target. After the command string, the infected machine sends an ACK packet, and an attack occurs. This describes the case of the plain-text attack message.

Decompiling and static malware code analysis revealed also other potential attacks like those using TCP over a predefined port, and other botnet command and control commands like the option to shut down bot or sending it on hold for some time.

D. Statistical features of botnet CnC traffic and normal traffic flows

In order to better understand the differences between CnC and normal traffic flows, we need to determine in what range does the data for each type of flow lie for each of the characteristics mentioned in previous section: the Pearson correlation coefficient, unidirectional flow throughput,

difference of unidirectional flow throughput. A plot of the statistical characteristics is given in Fig. 4. Normal flows are designated by green data points, while the botnet flows are designated by red data points. The y axis of this plot serves solely to separate normal and botnet data points.

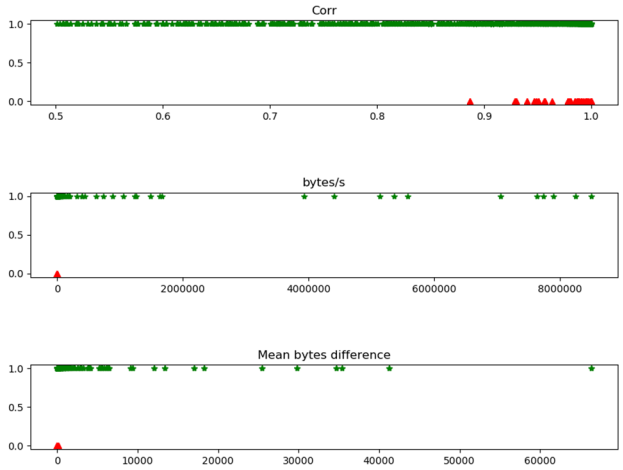

Fig. 4. Statistical characteristics plot.

As can be seen from the plot, the botnet and normal traffic flows differ significantly. Botnet traffic flows show a higher Pearson correlation coefficient, as well as low unidirectional flow throughput and difference of unidirectional flow throughput. This is due to the fact that botnet traffic occurs periodically, in bursts, and botnet and CnC exchange similar messages during the connection maintenance phase.

In order to gain a deeper insight into the statistical nature of these characteristics, we calculated the following for each one of the metrics: minimum, maximum, mean and standard deviation. This is useful for gaining a better understanding of the range and spread for each of the two types of network flows. The Pearson correlation is dimensionless, while unidirectional flow throughput (designated as UFT in subsequent tables) and difference of unidirectional flow throughput (designated as DUFT in subsequent tables are displayed in bytes/s. Botnet traffic flows' statistics and normal traffic flows' statistics are shown in Table 1.

The presented statistics of botnet and normal traffic, reveals several important properties of the botnet CnC traffic. The characteristics of botnet traffic are the following: higher values for the Pearson correlation, and lower values for the packet count and difference of unidirectional flow throughput compared to the normal traffic. Also, the standard deviations of botnet traffic are significantly smaller than those of the normal traffic. Having a low spread means that botnet flows of Mirai and Gafgyt botnets have a unique footprint, which could be used for creating filters which could ease their detection.

TABLE 1: STATISTICS FOR NORMAL FLOWS' CHARACTERISTICS AND BOTNET FLOWS' CHARACTERISTICS.

	PEARSON CORRELATION	UFT	DUFT
Normal Flows			
MINIMUM	0.501309	0.015455	0
MAXIMUM	1	8501640	66353.27
MEAN	0.950483	20379.22	225.6704
STD	0.093984	335266.9	2278.214
Botnet Flows			
MINIMUM	0.886722	0.944156	0.005734
MAXIMUM	1	318.8587	168.7144
MEAN	0.97302	10.86656	6.772756
STD	0.027745	46.37664	31.47784

V. CONCLUSION

This analysis served to detect behavior inherent to the recent versions of Mirai and Gafgyt IoT malware families, in order to aid botnet detection. There are several important conclusions of this research. First, in order to avoid deep packet inspection which uses signature-based detection, different malware versions use various strings formats in the CnC and attack commands. Second, all explored IoT malware-initiated communication with the botmaster avoiding DNS query. This means that IoT malware CnC still does not use advanced hiding techniques like DNS-fluxing or DGA, but also that this is a characteristic behavior pattern which can be used as an input into the suspicious behavior analysis. Third, all samples of both malware types showed characteristic CnC lifecycle phases and the periodic CnC maintenance pattern which cannot be observed with the flow or packet header analyses. Fourth, static code analysis revealed the strategies hackers use to avoid detection: constant minor changes of the ports, strings and IP addresses bots use to communicate. Fifth, the botnet CnC communication has a distinct behavior regarding the Pearson correlation, unidirectional flow throughput and difference of unidirectional flow throughput. However, the overall architecture of the malware remains the same for a longer period of time preserving the characteristic communication patterns. Analysis using Cuckoo sandbox was not particularly useful, as the information obtained from it about the dynamic malware behavior was too modest to be used for a more serious analysis (only the IP address of the botmaster was revealed). These results are encouraging for our further research towards creating a stateful programmable packet processing for the detection of botnet CnC, using this analysis as a starting point.

REFERENCES

[1] G. Vormayr, T. Zseby, and J. Fabini, "Botnet Communication Patterns," *IEEE Commun. Surv. Tutorials*, vol. 19, no. 4, pp. 2768–2796, 2017.

[2] M. Antonakakis *et al.*, "Understanding the Mirai Botnet This paper is included in the Proceedings of the Understanding the Mirai Botnet," *USENIX Secur.*, pp. 1093–1110, 2017.

[3] R. Chen, W. Niu, X. Zhang, Z. Zhuo, and F. Lv, "An Effective Conversation-Based Botnet Detection Method," *Math. Probl. Eng.*, vol. 2017, pp. 1–9, 2017.

[4] B. Krebs, "Zyxel Flaw Powers New Mirai IoT Botnet Strain", Krebs on Security website, https://krebsonsecurity.com/2020/03/zxyel-flaw-powers-new-mirai-iot-botnet-strain/ (accessed on April 25th 2020)

[5] R. Lakshmanan, "Dark Nexus: A New Emerging IoT Botnet Malware Spotted in the Wild", The Hacker News, https://thehackernews.com/2020/04/darknexus-iot-ddos-botnet.html (accessed on April 25th 2020)

[6] Đ. D. Jovanović and P. V. Vuletić, "Analysis and Characterization of IoT Malware Command and Control Communication,", *2019 27th Telecommunications Forum (TELFOR)*, Belgrade, Serbia, 2019, pp. 1-4.

[7] P. Narang, S. Ray, C. Hota, and V. Venkatakrishnan, "PeerShark: Detecting peer-to-peer botnets by tracking conversations," *Proc. - IEEE Symp. Secur. Priv.*, vol. 2014-Janua, pp. 108–115, 2014.

[8] S. Chowdhury *et al.*, "Botnet detection using graph-based feature clustering," *J. Big Data*, vol. 4, no. 1, 2017.

[9] V. Tong and G. Nguyen, "A method for detecting DGA botnet based on semantic and cluster analysis," *ACM Int. Conf. Proceeding Ser.*, vol. 08-09-Dece, no. December, pp. 272–277, 2016.

[10] R. Sharifnya and M. Abadi, "DFBotKiller: Domain-flux botnet detection based on the history of group activities and failures in DNS traffic," *Digit. Investig.*, vol. 12, pp. 15–26, 2015.

[11] J. Lee, J. Kwon, H. J. Shin, and H. Lee, "Tracking multiple C&C botnets by analyzing DNS traffic," *2010 6th IEEE Work. Secur. Netw. Protoc. NPSec 2010*, no. August, pp. 67–72, 2010.

[12] J. Kwon, J. Lee, H. Lee, and A. Perrig, "PsyBoG: A scalable botnet detection method for large-scale DNS traffic," *Comput. Networks*, vol. 97, pp. 48–73, 2016.

[13] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, and E. Gerhards-Padilla, "A Comprehensive Measurement Study of Domain Generating Malware," pp. 1996–2014, 2016.

[14] T. S. Wang, H. T. Lin, W. T. Cheng, and C. Y. Chen, "DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis," *Comput. Secur.*, vol. 64, pp. 1–15, 2017.

[15] J. Gardiner, M. Cova, and S. Nagaraja, "Command & Control: Understanding, Denying and Detecting," *arXiv.org*, vol. cs.CR, no. February, p. 1136, 2014.