Journal of Systems Architecture 116 (2021) 102143

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

The evolution of IoT Malwares, from 2008 to 2019: Survey, taxonomy, process simulator and perspectives

Benjamin Vignau [a,*,1], Raphaël Khoury [a,1], Sylvain Hallé [a,1], Abdelwahab Hamou-Lhadj [b,2]

[a] Université du Québec à Chicoutimi, Canada
[b] Université Concordia, Montréal, Canada

ARTICLE INFO

Keywords:
IoT security
Evolution of IoT malware
IoT botnet
VPNFilter
IoTReaper
Hide'n Seek
Echobot

ABSTRACT

The past decade has seen a rapidly growing interest in IoT-connected devices. But as is usually the case with computer systems and networks, malicious individuals soon realized that these objects could be exploited for criminal purposes. The problem is particularly salient since the firmware used in many Internet connected devices was developed without taking into consideration the expertise and best security practices gained over the past several years by programmers in other areas. Consequently, multiple attacks on IoT devices took place over the last decade, culminating in the largest ever recorded DDoS attack, the Mirai botnet, which took advantage of weaknesses in the security of the IoT. In this survey, we seek to shed light on the evolution of the IoT malware. We compare the characteristic features of 28 of the most widespread IoT malware programs of the last decade and propose a novel methodology for classifying malware based on its behavioral features. Our study also highlights the common practice of feature reuse across multiple malware programs.

1. Introduction

The advent of the *Internet of Things* (IoT), has made our world more connected. A study from the European Digital Economy Think Tank IDATE suggests that there are already billions of connected devices in the world [1]. These are Internet-connected devices that can interact with the physical world (referred to collectively as IoT), such as surveillance cameras, temperature and humidity sensors, automatic gate or smart heating. IoT devices are now widely used in fields as varied as medicine [2] or agriculture [3], contexts where a security vulnerability can have far-reaching implications. They generate a lot of data and are thus a key component of big data analytic. However, previous research has shown that their security and authentication functionalities are not always adequate [4]. Consumers increasingly feel the need to own and use connected smart objects, such as smart phones, smart watches and even connected intimate devices [5]. However, they also want these objects to be affordable, easy to configure and easy to use. This has led to the so-called "Plug-in-and-do-not-care" mentality [6]: users demand objects that require little to no installation, and no maintenance.

Moreover, industries seek to minimize production costs by providing default configurations, which may include poor or easily obtainable default credentials, and generally poor security features [6,7]. These weaknesses have lead to the appearance of large-scale botnets such as Mirai [7], that take advantage of security weaknesses to achieve various nefarious goals. Furthermore, while it is easy to observe that botnets are evolving, gradually becomes more sophisticated and more effective, reliable information about the functionalities of IoT botnets is often unavailable. When information is available, it is often provided in a heterogeneous manner making comparison between bots difficult.

The rapid evolution of IoT botnets and paucity of data that describe them accurately hinders the development of effective countermeasures. The present study aims to make headway in addressing these issues by providing a systematic study of IoT malware, organized in a thorough taxonomy that highlights the past evolution of IoT malware from 2008 to 2019. We further propose a model that will allow researchers to reason about and predict the future evolution of IoT malware. Our goal is to provide data and a taxonomy to help the scientific community to better understand the evolution of IoT botnets. We argue that a better understanding of the evolution of IoT botnets will help in the development of methods and tools to prevent and counteract IoT botnets. More specifically, this study makes the following contributions.

- We present a systematic literature review of IoT malware, devised following the guidelines provided by Wohlin [8]. We included every botnet that was active between 2008 and 2019, four of which (JenX, TheMoon2, Hide'n Seek and Echobot) have

* Corresponding author.
E-mail address: benjamin.vignau1@uqac.ca (B. Vignau).
[1] Laboratoire d'Informatique Fondamentale (LIF).
[2] Software Research and Technology Lab (SRT Lab).

https://doi.org/10.1016/j.sysarc.2021.102143
Received 17 October 2020; Received in revised form 7 April 2021; Accepted 8 April 2021
Available online 20 April 2021

never been the topic of academic research at the time of data collection (December 2019). We did not include botnets that appeared in 2020, because at the time of doing this study, too little information about them was available.
- Drawing upon the varied goals and features of the botnets from this time period, we propose a new taxonomy of IoT malware with 77 taxa. Guided by this taxonomy, we centralize and organize the current state of knowledge about these IoT botnets.
- We study the evolution of IoT botnets, and draw conclusions that may help researchers address the challenges raised by the proliferation of this type of malware.
- We introduce two novel graphical representations that provide striking visual representations of the evolution of IoT malware.
- We devise a new simulation model that allows researchers to predict the spread of future malware, thus anticipating the future evolution of IoT malware.

The remainder of this paper is organized as follows: Section 2 presents related works. In Section 3, we state the research questions that this paper seeks to answer and expose our methodology. In Section 4, we present a novel taxonomy of IoT botnets. Section 5 relies upon this taxonomy to describe and categorize 28 IoT botnets families. A further discussion of the evolution of IoT botnets is given in Section 6. Section 7 discusses broader conclusions that can be gleaned from the preceding analysis. In Section 8, we describe a new model that can be used to predict the future evolution of malware. Experimental results in the use of this model are given in Section 9. Section 10 describes limitations of our study and threats to validity as well as avenues for future research. Concluding remarks are given in Section 11.

All of our data and algorithms, as well as high resolution versions of the figures in this paper, are available on the author's GitHub repository.[3] This study is an extension from our previous study [9].

2. Related work

In order to best understand the evolution of IoT malware and their associated botnets we first arm ourselves with a expressive taxonomy that classifies the relevant malware features. To this end, we began by studying four existing taxonomies (De Dono et al. [10] Hachem et al. [11], Dagon et al. [12], Rawat et al. [13]), which we then extended to create a novel taxonomy that suits our needs.

We define three criteria to analyze and compare these taxonomies. Firstly, we consider the number of levels in each taxonomy, secondly, the total number of taxa and finally, the number of specific taxa related to the IoT. Not all of the taxonomies we examined focus specifically on IoT botnets, but they were still useful in designing and organizing our own.

The taxonomy proposed by Dagon et al. [12] focuses on botnets network structure, according to its usefulness to the attacker. They define specific responses for each network type. The taxonomy is based on four criteria: (1) efficiency of attack, (2) available bandwidth, (3) efficiency of communication and (4) robustness of the network. They also define metrics that quantify these criteria, namely the size of the network (number of victims), the available bandwidth at any given time, the amount of time required to transmit data to every zombie and the availability of redundancy mechanisms.

They analyze several types of networks including random Erdös–Rény networks, peer to peer networks and the Watts–Strogatz network. Most of these networks exhibit topologies that have never been observed in networks botnets. However, the idea of using the number of victims to measure the general efficiency of a botnet is intuitive and practical. We thus adopted it as a measure of the impact of features.

[3] https://github.com/bvignau.

Hashem et al.'s taxonomy [11] dates back to 2011, and draws upon the life cycle of botnets to classify their features. The taxonomy is organized in two levels. The top level contains four elements that mirror the four phases of a botnet's life cycle, namely "propagation and infection", "command and control system", "application" and "network resilience". The second level captures any feature observed in only some malware and botnets.

The "propagation and infection" criteria groups features used to infect victims, such as spam mails, software vulnerability exploitation, corrupted link sending, corrupted files sharing in P2P networks or use of another botnet. The latter feature corresponds to the rental of a botnet to propagate other malwares. Notice that only the exploitation of vulnerabilities or the use of another botnet can be useful to create an IoT botnet since IoT connected objects such as connected cameras or routers do not open mail or click on link.

In the "command and control" taxa, the authors distinguish two mains models: centralized and decentralized. The former is more efficient in its communications (distribution of orders, updates etc.) but presents a single point of failure which the second one does not exhibit. They also describe the topology and the communication protocols used. All of these features can be found in IoT botnets.

The "application" taxon captures the goal of the botnet. This taxon groups different attack types such as DDos, spying, spam distribution or malicious activities hosting.

The final taxon includes any other additional feature of a botnet. Such features can render the efficient, more discrete or more resilient. Features included in this taxon include update systems, obfuscation mechanism etc.

This taxonomy is descriptive with respect to the features of botnets. The taxonomy is based on empirical data and can be easily extended. This taxonomy also uses two hierarchical levels, 42 taxa, and includes most of the botnets observed between 1999 and 2009. Only two IoT botnets appeared before the end of 2009 and were not included in this study, so this taxonomy does not have included any taxon that describes features specific to IoT botnets. However, we drew upon the general organization of the taxonomy presented in Section 4 which also mirrors the life cycle of a botnet.

The taxonomy proposed by De Dono et al. [10] focuses on IoT botnets whose purpose is to create large-scale DDoS attacks. This taxonomy uses 4 different levels, with 44 taxa. Unlike the previous taxonomies, the top-level does not depend on the life cycle of a botnet, but rather describes families of features. This level contains families that represent the types of DDoS attacks performed, the IP sources used, the attack rate etc. It also includes features that describe the architecture of the botnet, as well as others that describe the method used to search for potential victims. This taxonomy is helpful because it describes state-of-the-art features such as network scan. Research shows that three main techniques can be used: a predefined list, random scan and permutation scan.

The taxonomy is based on the features observed in 13 IoT botnets. However, all of these features can be found in other, older non-IoT botnets. None of the features described in this taxonomy are specific to IoT botnets. Moreover, this taxonomy is exclusively focused on malware that performs DDoS attacks. As a consequence, several behaviors common in IoT botnet, such as spying or incomes generation, are not described.

In this taxonomy, two bots are linked if they share part of their code. The authors thus performed initial work in showing the relationships and borrowings that occur between different bots. This initial work does not indicate which part of the code, or which features are shared. In our own taxonomy, we extend this idea by showing specifically which features are reused from one bot to the next.

The final taxonomy we have studied is the one devised by Rawat et al. [13]. Their taxonomy focuses on decentralized botnets. They present a taxonomy that classifies botnets as well as the techniques used to detect them. We focused our analysis on the part of the taxonomy

Table 1
Table 1: Comparison between taxonomies.

Taxonomy	# levels	# taxa	# IoT taxa
De Dono et al. [10]	4	44	0
Hachem et al. [11]	2	42	0
Dagon et al. [12]	2	7	0
Rawat et al. [13]	2	5	0
Our taxonomy	3	77	8

related to botnets. This part contains five taxa, split in two families. The first family corresponds to the network structure (P2P, hybrid etc.) while the second groups communication protocols.

Drawing upon these four taxonomies and on the features we observed in 28 IoT botnets, we were able to create our own taxonomy with three hierarchical levels and 77 taxa, including 8 taxa that capture features that are specific to IoT botnets, and not present in other types of malware. Our taxonomy contains 4 families based on the life cycle of a botnet. In our taxonomy we update, deepen and extend the taxonomy of [11], particularly with respect to the goals and efficiency features of IoT botnets. However our taxonomy does not deepen the DDoS taxonomy of De Dono et al. [10], thus our contribution is focused on the gathering of all actual and distinctive features of a IoT botnets based on features observed in IoT botnets families. A summary comparison of the five taxonomies is given in Table 1.

3. Methodology

3.1. Definition

Hackers use a dedicated malware to infect and control several computers. Those computers are called "bots" or "zombies". In our study, we focused on botnet created by IoT devices, such as DVR, connected camera, routers etc. The botnet threat has become increasingly important during the last decade, as malicious adversaries have utilized them to commit a variety of Internet crimes such as DDoS attacks, identity theft, email spamming and click fraud [14].

In keeping with the well established analogy between malware and biological viruses, this paper aims to study the evolution of IoT botnet over the last ten years.

A key originality of our research is that we characterize the evolution of IoT malware and botnet by the features they contain. Here we define a feature as follows :

Definition 1. A **feature** is a distinctive behavior implemented by a piece of malware or by it's corresponding botnet, regardless the implementation. It is a distinguish characteristics of the malware program, or of the corresponding botnet.

For example, two botnets using Syn Flood to cause a DDoS attack are both seen as having the Syn Flood feature, even if one implements this feature in python while the other implements the feature in C.

3.2. Research questions

The central question that this study aims to answer is: "How is malware targeting IoT evolving?". This general question can be divided in five sub-questions that lend themselves to quantifiable answers.

From our definition, we see that a malware is defined by the features it implements, while a botnet is defined by its goal, its size (number of victims), and its corresponding malware. We also add speed of propagation and a description of the damages that produce the botnet.

From this discussion we create five research questions. For a given malware M_x or botnet B_x:

- RQ1: What features define malware M_x?
- RQ2: What are the of goals the botnet B_x?
- RQ3: How many victims has the botnet B_x infected?
- RQ4: How much time does botnet B_x require to infect its victims?
- RQ5: What are the damages caused by botnet B_x?

These questions all lend themselves to quantifiable answers. To answer the first question, we first list every behavioral feature of all IoT malware (Section 4). This allows us to subsequently assign to each malware the set of features that it implements.

We then repeat this exercise with the various goals of botnets, which allows us to answer the second question. In order to answer the final three questions, we dig into academic papers, press accounts and technical reports and extract the required data.

Finally, from the above answers, we create novel graphical representations that visually highlight the evolution of IoT malware, and compile statistics that inform on this evolution.

3.3. Dataset

To answer these questions, we first gather as much data as possible about every IoT malware that was active during the last 10 years. We compiled a list of 28 recent IoT malware families (listed in Section 5), on which this study will focus.

For each of the 28 IoT malware that we identified, we performed the following two queries on google.scholar: (botnet $\vee$ $MalwareName$) and (malware $\vee$ $MalwareName$), where MalwareName is the name of each malware. We also performed the same queries on Web of Science, but most of the queries either yielded no results, or only returned papers already present in the results returned by Google Scholar.

We then saved the top 10 results for each query, for a total of 560 results. We excluded two papers written in Korean and Chinese languages. We then employed a python script to detect duplicate papers. At the term of this process, we obtained a corpus of 256 different papers. For each of these papers, we recorded the associated queries and the malware concerned by the paper.

3.4. Corpus creation and exclusion criterion

Several papers appeared in the results of multiples queries. We assigned each paper a score, equal to the number of queries that returned it, with the intuition that papers with the higher scores are more likely to provide at least partial answers to multiples RQs. We found that papers with the highest scores often provided syntheses and analysis of data from multiple sources. To gather a maximum amount of information for each malware, we also studied the sources of these papers.

To this end, we sorted the papers in our corpus in decreasing order of score. Then, in concordance with the Wohlin guidelines [8], which instruct on the process of performing an effective systematic literature review, we performed a backward snowball in order to gather more relevant papers.

Following these guidelines, we first excluded irrelevant papers, i.e. papers that do not address the topic of our study, namely the features and behavior of IoT botnets. Moreover, we also excluded papers that did not describe at least one feature related to a given botnet or malware, or did not include data about this malware that was relevant for our study. For example, a paper might describe a novel method to detect botnets in a network, and use a malware such as Mirai to generate data for the algorithm, without clearly describing a feature of this malware. We also excluded papers that only provided general information about a malware, i.e. papers that only mention in passage the name of the malware, or mention that it performs a DDoS attack without specifying the type of DDoS attack.

At the end of this phase, 31 papers remained. These papers include 5 papers that lacked the types of specific information we sought (on the features and behaviors of IoT botnet) but were kept in the corpus because they contained numerous sources about several IoT botnets

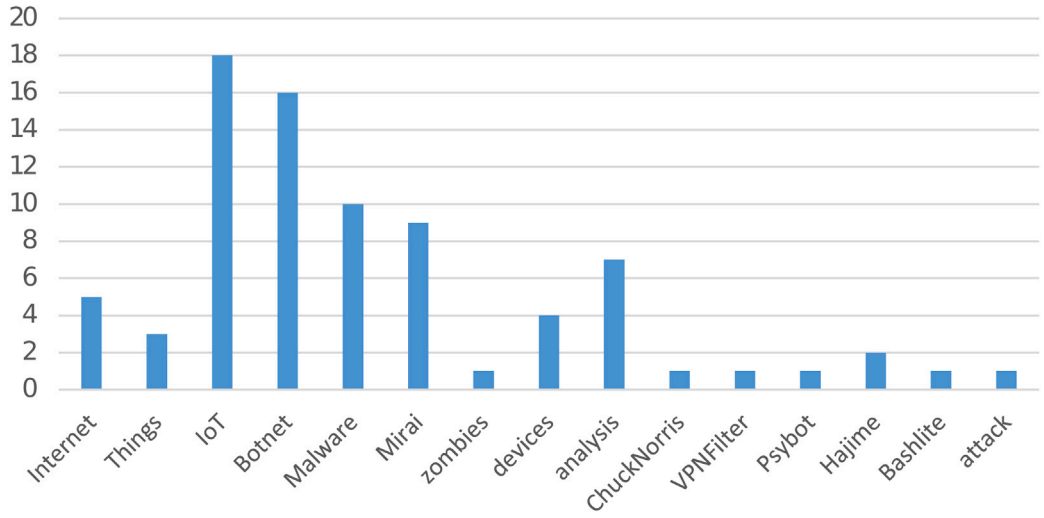


Fig. 1. Number of occurrences of each keyword.

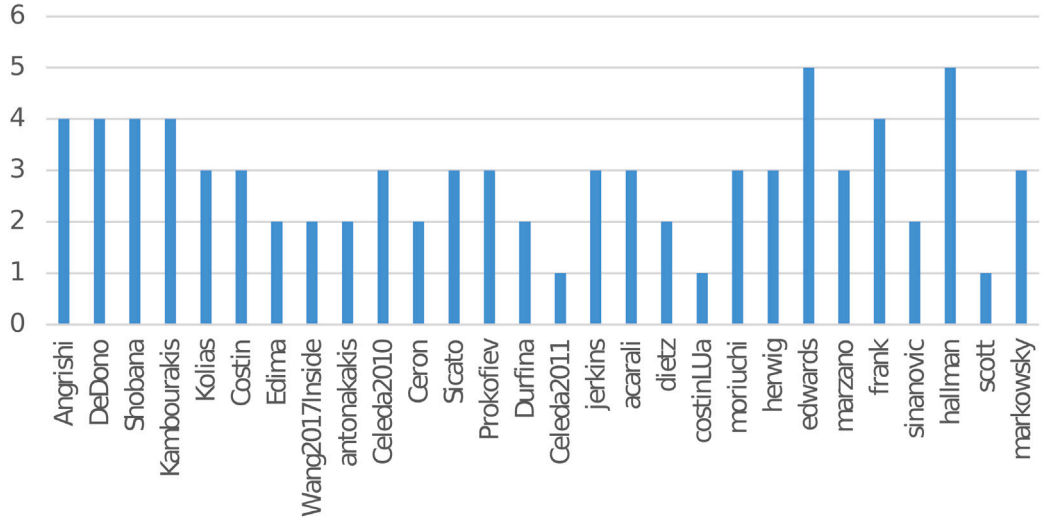


Fig. 2. Number of keywords for the first 28 sources.

that we judged worthwhile to extract. We removed these 5 papers from the corpus after the snowball.

We then proceeded with source extraction (backward snowball), for each of these papers. To aid in this tedious phase, we developed python scripts that extract references from the papers. The tool uses the `refextract` library.[4]

The snowball process yielded 1545 references. We again deleted duplicate references and filtered the remaining papers by excluding any document whose title did not include at least one of the following keywords: "corrupted, internet, thing, corruption, infection, infected, iot, malware, botnet, mirai, zombies, army, analysis, attack, comparison, connected, devices, hydra, psybot, chuck norris, carna, aidra, tsunami, kaiten, darlloz, qbot, bashlite, wifacth, mirai, hajime, amnesia, reaper, brickerbot, satori, jenx, moon, vpnfilter, echobot". This filter allowed us to narrow the scope of the snowballing process, and keep only the most relevant papers. The list of keywords was created by examining the titles of the 28 papers obtained in the initial phase of the corpus creation process. The title of each paper contains at least one keyword, at most five keywords and the mean number of keywords is 2.85. Figs. 1 and 2 provide a detailed breakdown of the frequency of occurrence of the keywords in the title of the papers in our corpus.

This process yielded 413 new sources. We then read the title and abstract of each reference and removed irrelevant paper based on the exclusion criteria presented above. Then, for the remaining papers, we additionally read introduction and conclusion. At the end of this process, we added 36 new references to our corpus, 5 new papers and 31 web articles. We again extracted the sources of the 5 new papers, but we did not find new relevant sources to add in our corpus, leading us to end the snowball process. Finally, we performed a Google search in order to obtain sources for recent botnets for which descriptive academic papers are not yet available, such as JenX or Echobot. This process added 31 web sources to our corpus.

At the end of the entire process, our corpus consisted of 31 academic papers and 62 web sources. All python scripts are available on the author's git repository.[5]

[4] https://github.com/inspirehep/refextract.

[5] https://github.com/bvignau/CorpusCreator.

Table 2
Goal features.

Category 1 : Goal					
DDoS (1.1)	PDoS (1.2)	Spying (1.3)		Money generation (1.4)	Attack vector (1.5)
SYN Flood (1.1.1)	Firewall DoS (1.2.1)	Industrial spying (1.3.1)	General Spying (1.3.2)	Bitecoin (1.4.1)	Spam (1.5.1)
UDP Flood (1.1.2)	Memory DoS (1.2.2)	Local network mapping (1.3.1.1)	Data exfiltration (1.3.2.1)	Dogecoin (1.4.2)	Backdoor as a Service (1.5.2)
ICMP Flood (1.1.3)		SCADA Monitoring (1.3.1.2)	Data obfuscation (1.3.2.2)	Litecoin (1.4.3)	
ACK-PUSH (1.1.4)		Reverse VPN (1.3.1.3)	MitM attacks (1.3.2.3)	Monero (1.4.4)	
HTTP Flood (1.1.5)			DNS Spoof (1.3.2.4)	Ad Fraud (1.4.5)	
TCP XMAS Flood (1.1.6)			Other devices exploitation (1.3.2.5)		
DNS Amplification (1.1.7)					
DNS Water Torture (1.1.8)					
GRE ETH Flood (1.1.9)					
GRE IP Flood (1.1.10)					
VSE Flood (1.1.11)					
TS3 Flood (1.1.12)					

4. Taxonomy

From the detailed descriptions of malware in our sources, we were able to distinguish 77 features. In order to gain a better understanding of these features, we organized them in a taxonomy consisting of four main categories: Goal, Infection, Organization and Efficiency. These categories are further divided in families (and some also contain sub-families), which in turn contain features. The first level of our taxonomy, called "Category" is presented in Fig. 3. The second and third levels are termed "Family" and "sub-family" respectively. The leafs of our taxonomy are the features, also called taxa. For ease of identification, we provide a hierarchical numbering scheme for features. For example, the Goal category has the number one, the DDoS family has the number one in the Goal category, and the Syn flood feature is the first feature in the DDoS family, so we assign this feature the number 1.1.1. Each digit corresponds to a hierarchical level of the taxonomy.

The remainder of this section details our taxonomy and each feature used by the IoT botnets in our study.

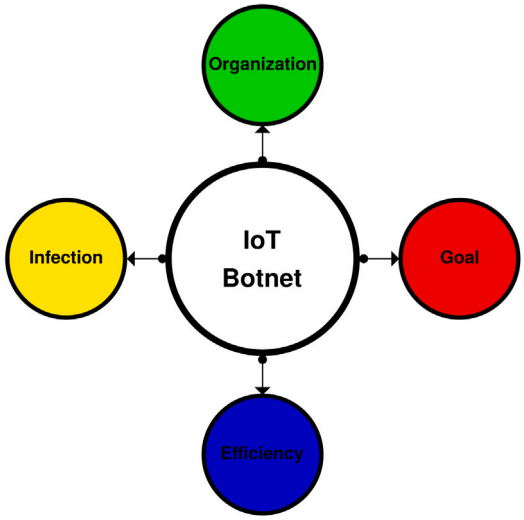


Fig. 3. Taxonomy: first level of our taxonomy.

4.1. Goal (1)

The Goal (1) Category contains the features implemented by a botnet that capture its ultimate objective. This category contains five families: DDoS, PDoS, Spying, Income generation and Attack vector distribution. The taxonomy of this category is given in Fig. 4, and a recapitulative table is given in Table 5 (see Table 2).

DDoS (1.1)

This family includes all features related to Distributed Denial of Services attacks (DDoS). This includes: Syn Flood (1.1.1), UDP Flood (1.1.2), ICMP (1.1.3), ACK-PUSH (1.1.4), HTTP (1.1.5), TCP XMAS (1.1.6), GRE ETH (1.1.9, GRE IP (1.1.10), VSE (1.1.11) and TS3 (1.1.12). These types of attacks aim to send as many packets as possible to the intended victim, thus overwhelming his resources. The different features mostly differ in the type protocol used to accomplish this task. For example, the UDP Flood proceeds using UDP packets. Most of these types of attacks are well known in the security community. We mention the VSE flood, which uses the Valve Source Enginee protocol to attack game servers; the TS3 protocol is a VoIP protocol used by the TeamSpeak application and the TCP XMAS Flood, in which an attacker sends TCP packets in which every flag is set to 1.

The DDoS family also includes the DNS Amplification attack (1.1.7) in which each attacker request a different small DNS query, using the IP address of the victim. The DNS server will then send multiple responses to the victim, leading it to exhaust its bandwidth. The DNS water torture (1.1.8) aims to exhaust the authoritative DNS of the target. This is done by sending multiple random subdomain query to the DNS. Each of these attacks is described by De Dono et al. [10].

PDoS (1.2)

PDoS stand for Permanent Denial of Services or Physical Denial of Services. In this case, the aim is to physically disable an object by overwriting its memory (1.2.2) or by changing the rules of its firewall to drop all connection (1.2.1).

Spying (1.3)

This family includes two sub families: Industrial Spying and General Spying. The former groups all features that are specially crafted to spy on industrial plants or large corporations. The latter includes more

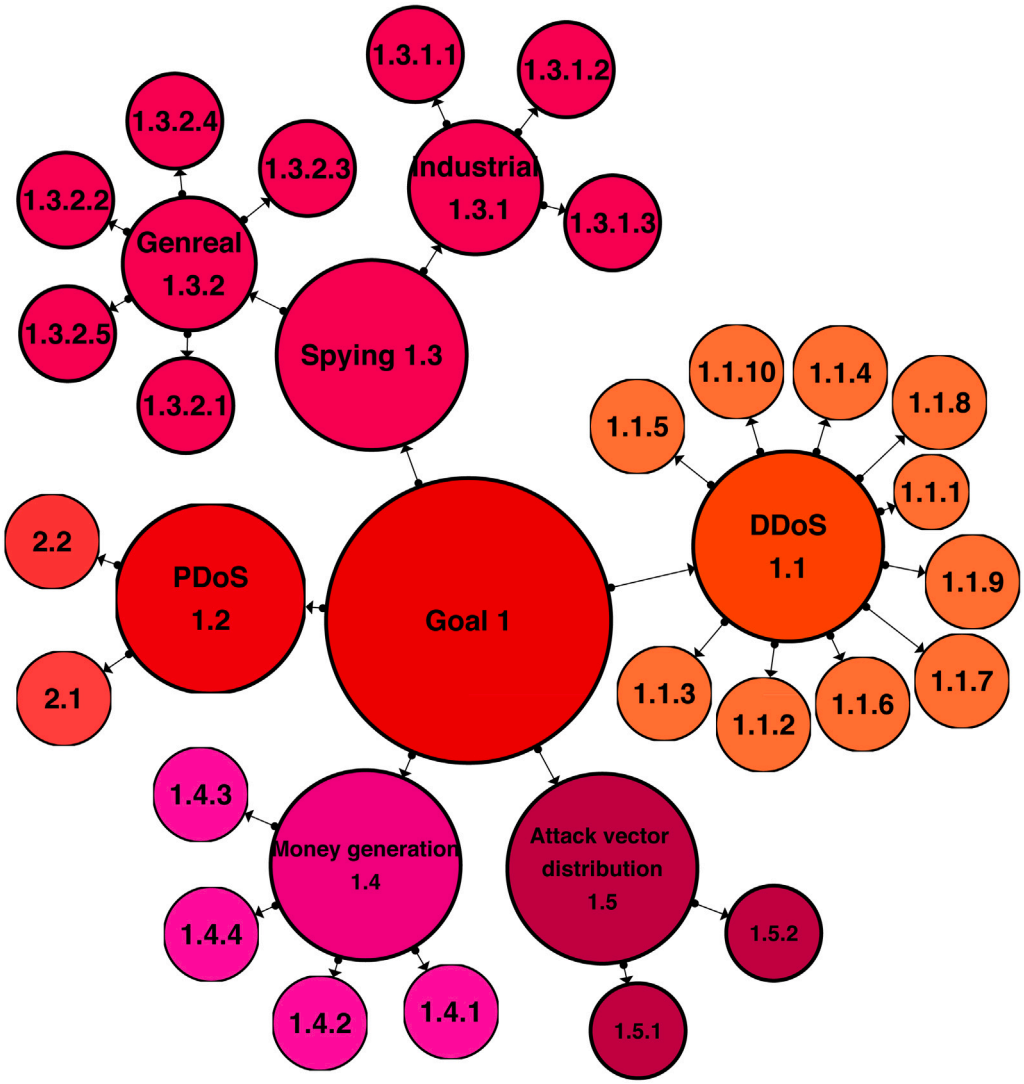


Fig. 4. Taxonomy: Goal category.

general spying features, that target everyone, from common users to large multinationals. In the Industrial spying subfamily (1.3.1) we include features that map all local subnets (1.3.1.1), features that monitor industrial controller such as SCADA (1.3.1.2) and features that create a reverse VPN, which can be used by attackers to gain access to the local networks of the company. In the general spying (1.3.2) sub family, we have included features such as data exfiltration (1.3.2.1) and data obfuscation (1.3.2.2) used to steal data from users. This sub family also includes the man-in-the-middle attack (1.3.2.3) and DNS spoofing (1.3.2.4) which redirect users to malicious fake web sites in order to steal their credential. Finally it includes features that exploit other computers in the local network (1.3.2.5).

Income generation (1.4)

This family groups all features that are used to generate income directly using the botnet. It includes cryptomining features such as Bitcoin (1.4.1) DogeCoin (1.4.2), LiteCoin (1.4.3), Monero (1.4.6) as well as Ad fraud (1.4.5). In an Ad fraud scheme, bots navigate to a predefined website and click on ads in order to generate revenue to the botmaster. The most advanced form of profit generation is called Botnet as a Service (BaaS) (1.4.7). In this scheme, botmasters rent out the computational power of the infected devices in order to provide numerous services such as large scale anonymous proxy networks, credential brute-force etc.

Attack vector distribution (1.5)

As mentioned above, some botnet are rented out to spread other malware or other botnet. For example Sirmer and Streda [15] monitored the Necurs botnet and observed that this botnet was spreading another botnet. The taxa 1.5 groups features that are used by botnets to propagate other malware. This can be done by sending spam (1.5.1) or by backdooring infected objects and then using the backdoor to propagate other malware rapidly. We name this latter feature Backdoor as a service (BaaS) (1.5.2).

4.2. Infection (2)

This category groups all features related in the infection process. This category contains three families: Exploitation methods (2.1), Victim's architecture (2.2) and Scanning methods (2.3) (see Fig. 5).

Exploitation methods (2.1)

This family groups every exploitation technique used by malware to infect IoT objects. This includes the dictionary attack (2.1.1) in which the bot attempts to brute force the victim's credential, often on telnet or SSH services. The aim is to gain access to the object, and download the malware on the victim's system. Another brute force technique, was observed by Antonakakis et al. [7] during their analysis of the Mirai botnet, called the weighted dictionary attack (2.1.2). In

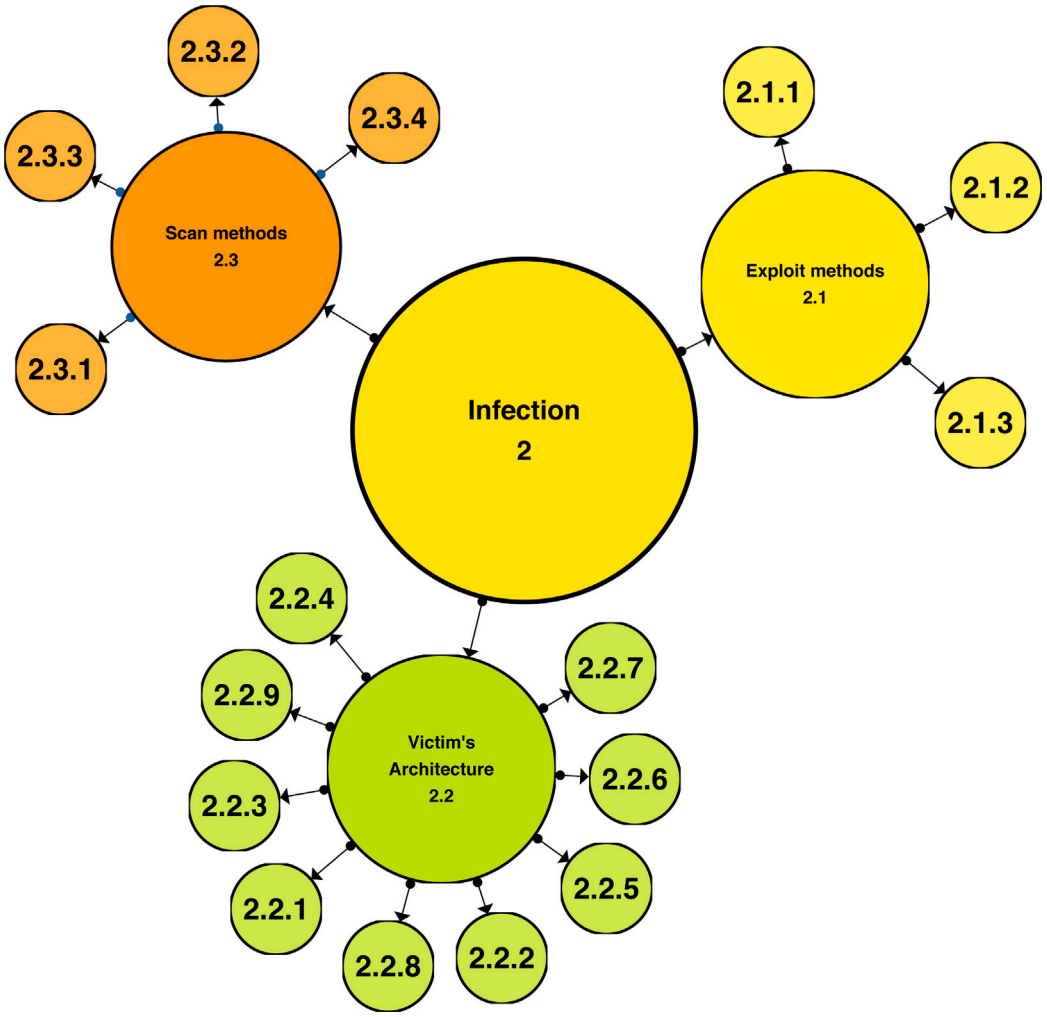


Fig. 5. Taxonomy: Infection category.

a weighted dictionary attack, the malware possesses a preprogrammed list of common credentials. At the time of infection, it randomly chooses a subset of these credentials, and tests them against the victim. This methods improves the scanning speed, but reduces the probability of gaining access to an object. The final feature is the exploitation of one or more CVE (2.1.3). In this case, the botnet implements an exploit targeting specific devices or software implemented by IoT. Generally it is far quicker to scan for a CVE than to brute force credentials. However, this exploitation method is also more difficult to implement (see Table 3).

Victim's architecture (2.2)

This family lists the different types of device architectures that each malware is capable of infecting. The IoT world is highly heterogeneous and most malware consist of a simple binary program, designed to target a specific architecture. In order to be able to infect an object, botnets need to install a binary program compatible with the processor architecture of the intended victim. Some malware includes multiple payloads, and are thus capable of infecting different types of devices. We identified nine types of architectures that have been the target of at least one malware in our sample set: MIPS (2.2.1), MIPSEL (2.2.2), ARM (2.2.3), x86/64 (2.2.4), BASH (2.2.5), PowerPC (2.2.6), Motorola 6800 (2.2.7), Sparc (2.2.8) and Hitachi SH (2.2.9). The BASH (2.2.5) feature represent malware that are fully coded in bash script, and are able to infect all devices where shell scripting is available (such as busybox devices), regardless of the architecture of their processor.

Table 3
Infection features.

Category 2 : Infection		
Exploitation (2.1)	Architecture (2.2)	Scan (2.3)
Dictionary Attack (2.1.1)	MIPS (2.2.1)	Hitlist (2.3.1)
Weighted Dictionary Attack (2.1.2)	MIPSEL (2.2.2)	Sequential Scan (2.3.2)
CVE Exploit (2.1.3)	ARM (2.2.3)	Stateless Scan (2.3.3)
	x86/64 (2.2.4)	
	BASH (2.2.5)	
	Power PC (2.2.6)	
	Motorola 6800 (2.2.7)	
	Sparc (2.2.8)	
	Hitachi SH (2.2.9)	

Scanning methods (2.3)

This family contains the different scanning strategies used by botnets to detect potential victims. The simplest is the hit-list (2.3.1) which consists in a preset a list of IP addresses that each bot scans. This strategy can be useful if an attacker already knows the IP addresses

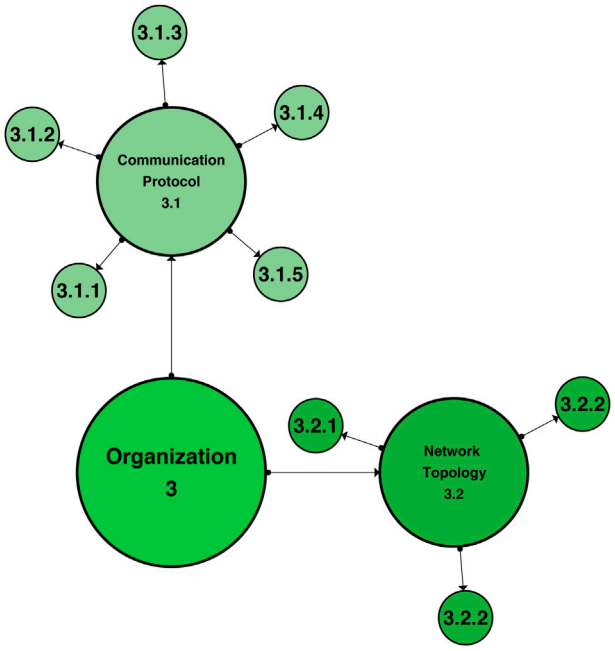


Fig. 6. Taxonomy: Part 3 organization.

of potential victims. The second one, trivial to implement, is the sequential scan method (2.3.2), where each bot scans the entire universe of IP addresses in sequential order (e.g. 1.1.1.1, 1.1.1.2, 1.1.1.3 etc.). Next, another relatively simple method to implement is the random scan (2.2.3) in which IP address are randomly generated and attacked. Finally a few bots implement the random stateless scan (2.2.4). When employing this scanning strategy, the IP addresses are also randomly generated. However, the botnet does not wait a response from the victim before proceeding with the scanning process. When a victim replies, the bot simply sends a reset packet and then restarts the entire communication in order to try to exploit the victim.

4.3. Organization (3)

This category groups all features that are used to control a botnet. This category contains two families: one to describe the communication protocol used by botnets, and the second to describe the network topology of botnets (see Fig. 6).

Communication protocol (3.1)

This family lists the protocols used by bots to communicate with C2 server or with other bots. The first method used is the IRC protocol (3.1.1). In this case, botmaster sets up an IRC server and bots log-in through dedicated channels. The botmaster subsequently sends commands through this channel, which the bot will parse and execute. The second protocol feature we observed is used by P2P botnets. They use a modified version of the BitTorrent and μTorrent protocols to communicate. We called this feature *BitTorrent derived communication* (3.1.2). We also observed botnets that create their own communication protocol using binary commands, we named this feature *Custom protocol* (3.1.3). Finally, some recent botnets use HTTP (3.1.4) and HTTPS (3.1.5) protocols to communicate with the botmaster (see Table 4).

Network topology (3.2)

In our study we observed two main forms of network typology: centralized (3.2.1) and decentralized (3.2.2) topologies. In the first case, each bot communicates with a central server (also known as command and control (C2) server). This server maintains a list of all infected devices and is used by the botmaster to manage the botnet, send attacks commands etc.

Table 4
Organization features.

Category 3 : Organization	
Communication protocol (3.1)	Network topology (3.2)
IRC (3.1.1)	Centralized (3.2.1)
BitTorrent derivative (3.1.2)	Decentralized (3.2.2)
Custom protocol (3.1.3)	Hybrid (3.2.3)
HTTP (3.1.4)	
HTTPS (3.1.5)	

In a decentralized topology, there is no C2 server; the botnet forms a P2P network and the botmaster uses the communication protocol to send orders that will spread through the network.

Some bots also use a hybrid topology (3.2.3), in which most of the bots are in a P2P network, but a portion are controlled directly by the botmaster through a C2.

4.4. Efficiency (4)

The final category contains the features that a botnet uses to improve its efficiency and as well as a few other features that did not fit neatly into any other family. There are five families (see Fig. 7).

Intelligence (4.1)

Features included in this family implement behavior that allows a botnet to adapt to different situations. The first such feature is code modularity (4.1.1), which allows the botmaster to quickly add features to the botnet, by way of an update mechanism. The second feature is the detection of the architecture of the victim (4.1.2). This feature allows a botnet to send only the appropriate payload to the victim, instead of dropping and attempting to run multiple malicious binary.

Persistence (4.2)

Most of the malware studied only reside in the RAM, therefore, a simple reboot of the device is sufficient to clean the infection. In this family we detail features that allow botmasters to increase the lifetime of their malware on the victim's system.

The daemonization feature (4.2.1) consists in creating a daemon on a Linux system that can execute (and sometimes download) the malicious binary after a reboot. Some botnets kill the watchdog (4.2.2), thus preventing automated reboot. Others add an entry in the CRON table, to execute the malware regularly. A few malware act as Windows viruses and infect another program (4.2.4) or a file (4.2.5). In these cases, the malicious payload is executed with the infected program or when the infected file is opened.

Prevention (4.3)

Several IoT botnets use the same technique to exploit IoT: brute forcing telnet or SSH credentials. Moreover the source code of some botnets has been leaked, leading to the proliferation of multiple instances of the same botnet. Since the number of vulnerable IoT is limited, these botnets must compete to control available targets. The Prevention family (4.3) contains features implemented by botmasters to help their malware to effectively compete for control of bots.

One of the best technique is to detect and delete other malware on the infected device (4.3.1). However this technique works only to protect against older botnet, and furthermore the attacker must be capable of detecting these specific botnets. Another method consists in closing ports through the addition of firewall rules or by stopping

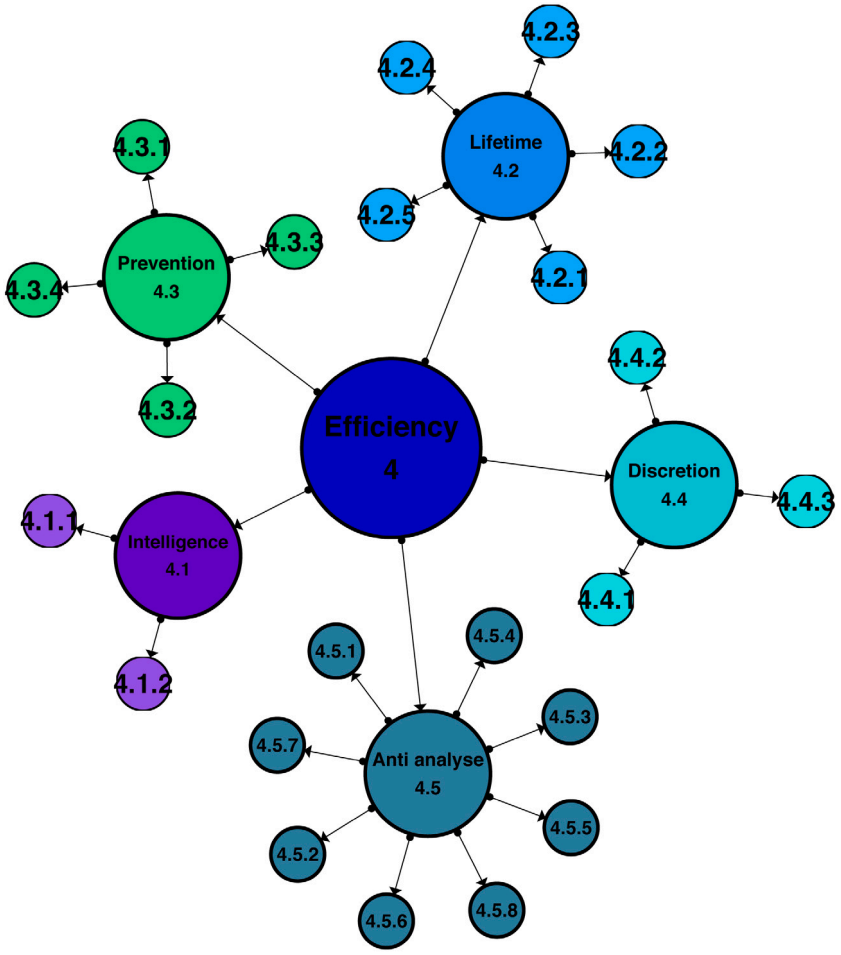


Fig. 7. Taxonomy: Efficiency family.

Table 5
Efficiency features.

Category 4 : Efficiency				
Intelligence (4.1)	Persistence (4.2)	Prevention (4.3)	Discretion (4.4)	Anti Analyze (4.5)
Code modularity (4.1.1) Victim's architecture detection (4.1.2)	Daemonisation (4.2.1) Anti reboot (4.2.2) CRON execution (4.2.3) Program infection (4.2.4) File infection (4.2.5)	Other botnet removal (4.3.1) Closing ports (4.3.2) Password modification (4.3.3) PATCH (4.3.4)	Binary removal (4.4.1) Process name stealing (4.4.2) Random name (4.4.3)	detection of virtualized environment (4.5.1) Bad ELF information (4.5.2) Corruption UPX header (4.5.3) Anti debug (4.5.4) Anti execution (4.5.5) Malicious code execution delayed (4.5.6) DGA Algorithm (4.5.7) Obfuscation of important information (4.5.7)

associated services (4.3.2). If multiple IoT are exploited via telnet or SSH, closing the associated port, or stopping the associated service, will prevent further exploitation of the device.

The final two features consist in changing the root password (4.3.3) and patching the devices against known vulnerabilities (4.3.4), in order to avoid any further attacks.

Discretion (4.4)

Most corporations track the activity botnet and their corresponding malwares on their networks. Consequently, attackers have developed features to make their malware more discrete, and thus try to avoid detection. The first such feature is to delete the binary after the onset of the execution (4.4.1). With this feature, malware only resides in the RAM and cannot be detected through an inspection of the file system.

The two others feature are related. The first is to kill a process and steal its name (e.g. telnet) (4.4.2) and the second is to give the malware process a randomly generated name (4.4.3). This feature makes it much harder to detect the malware using an automated process, such as an anti-virus.

Anti-analysis (4.5)

Malware's creators also seek to slow down the analysis of their malware by whitehat teams. To this end, they often add features that disrupt analyses and reverse engineering methods.

Some malware is able to detect virtualized environments (such as QEMU or VmWare) (4.5.1) and modify its behavior accordingly in order to avoid detection. This feature is especially common in Windows malware. An older technique is to place an erroneous ELF descriptor in the binary (4.5.2), leading the analysis tools to crash. However this is a dated trick and may not work with modern analysis tools. An often used technique is to insert a malformed UPX header (4.5.3) to confuse analysts that use tools such as "strings".[6] This method is also not very effective, since researchers can examine the header, recognize the binary and unpack it.

A more efficient technique is to use an anti-debug mechanism (4.5.4) to prevent the use of tool such as `gdb`, or to use an anti-execution mechanism (4.5.5). The latter is implemented by checking the presence of a string or a file in the victim file system. Likewise, some hackers delay the execution of the malicious code in their binary (4.5.6), which can fool automated dynamic analyses.

In order to delay the discovery of the C2 server some malware obfuscate important information such as the IP address of the C2 server. This technique is also quite inefficient because a dynamic analysis will reveal this address to researchers.

The next feature is called *Domain Generation Algorithm* (DGA) (4.5.8). When using this feature, each bot generates a predefined number (usually about 100) of random domain name each day, and tries to connect to each of them. The botmaster registers one of these domain names and redirects traffic from it to the C2. If the C2 is stopped for any reason, the botmaster can simply register a new domain name. The number of domain names generated is often too big to allow law enforcement to block them all, and allows the C2's IP address to be updated daily. As a consequence, DGA is a very effective feature to slow down researchers and law enforcement.

A final feature captures botnets that create P2P networks that are hardened against the analysis of their network by researchers. We call this feature "Anti Network Scan" (4.5.9). In this case, the botmaster develops a custom P2P communication protocol that allows him spread information to the entire network, while leaving each peer limited in its knowledge of the network. This feature has been observed only in the Hide'n Seek botnet.

5. Botnet descriptions and features

In this section we provide a brief description and context for each botnet included in our study. We list the relevant sources and detail the features of each botnet, as described in the academic literature or determined by ourselves through an analysis of the malware's code, when available. Information about some botnets was unavailable. Consequently, the fact that a feature is not assigned to a given bot does not conclusively mean that the malware does not have this feature. For example, we were unable to determine organization of the Carna botnet. This botnet was quite ephemeral and studies about it were mostly based on information provided by the author. Botnets are listed in order of appearance.

Hydra (1) – 2008

Description – Hydra is not properly a malware but rather a framework to create DDoS botnets using IoT objects. It was designed to be extensible and to facilitate the creation of malware by third parties. The source code was released in 2008.

Features – According to the sources listed below, Hydra allowed the creation of IRC botnets, using brute force dictionary attack on telnet and the D-Link bypass exploit [16] to gain control of its victims. Binaries where found for MIPS and MIPSEL architectures and these perform Syn and UDP flood attacks. To scan IP addresses, the botmaster must provide a preset list of IP addresses. (Sources: [10,17–22].)

Psybot (2) – 2009

Description – Psyb0t is the first malware discovered in the wild that targets home routers. It was discovered in early 2009 and was killed by its botmaster. The botnet infected between 80 000 and 100 000 devices.

Features – Psyb0t is an IRC centralized botnet, and proceeds through brute-forcing SSH and telnet credentials using a dictionary. This dictionary contains about 6000 usernames and 13 000 passwords. The botnet also uses the D-link bypass exploit. Once a device is infected, Psyb0t closes the telnet port and begins a sequential scan of the IP space. Moreover, the botnet is able to launch DDos attacks using UDP, ICMP and Syn flood features. (Sources: [6,10,17–21,23,24].)

Chuck Norris (3) – 2009

Description – A few month after the death of Psyb0t, another botnet was discovered. It was named "Chuck Norris" because the source code contained the phrase: "*In nomine di Chuck Norris*" (In the name of Chuck Norris). Researchers think that the authors of Chuck Norris and Psyb0t are the same due to the numerous similarities between their binary files.

Features – Chuck Norris is an IRC centralized botnet. It proceeds thought brute-forcing SSH and telnet credentials using a dictionary. The botnet was also able to use the D-link bypass exploit. Once a device is infected, Chuck Norris closes the telnet port and begins scanning the IP space with a random scan. Moreover, the botnet is able to launch DDos attacks using UDP, Ack-Push and Syn flood features. The malware is able to change the DNS settings in order to redirect some websites (such as google.com or facebook.com) to malicious copies and perform a man-in-the-middle attacks. (Sources: [6,10,17–21].)

Tsunami (4) – 2010

Description – Researchers believe that this malware is an evolution of Chuck Norris, due to the presence of numerous commons strings in their binaries. Moreover some of the IP addresses of their C2s are the same for both malware. This malware was named Tsunami (or Kaiten) because it uses an open source DDoS protocol named Tsunami/Kaiten. (Sources: [10,17,18,21,25].)

Features – This botnet mostly uses the same features as Chuck Norris. Its purpose is to provide DDoS services and it is also an IRC centralized botnet.

Aidra (4) – 2012

Description – Aidra is an open source tool used to scan and exploit IoT on the Internet. Due to its open source nature, multiple variants have been published. Some research teams recorded more than 700 different binary variants of this botnet family.

Features – Like its predecessors, Aidra's goal is to provide DDoS services. To this end, it scans for open or weakly securely secured SSH and TELNET services. Binaries were found corresponding to six different processor architectures, including MIPS, MIPSEL, x86/64, ARM, PPC and SuperH. Another interesting behavior of this malware is to search for older malware such as Chuck Norris and delete them. (Sources: [10,17,18,26,27].)

Carna (5) – 2012

Description – Unlike the previously studied botnet, the Carna botnet was ephemeral and was not aimed at performing DDoS attacks. This botnet is akin to a scientific experiment to map the Internet's IPv4 use [28]. The authors published a white paper that describes their experiment and details theirs results. They claimed to have infected 1.6 millions IoT with Carna. The authors further stated that "A lot of devices and services we have seen during our research should never be

[6] https://linux.die.net/man/1/strings.

connected to the public Internet at all. As a rule of thumb, if you believe that 'nobody would connect that to the Internet, really nobody', there are at least 1000 people who did. Whenever you think 'that should not be on the Internet but will probably be found a few times' it is there a few hundred thousand times. Like half a million printers, or a million webcams, or devices that have root as a root password" . This comment illustrates the importance of choosing strong passwords for IoT connected devices.

Features – As is the case for its predecessors, Carna exploits weak credentials on SSH and Telnet to infect routers, cameras and other IoT. It then implements a sequential distributed scan, in which each bot will scan a small part of the IPv4 space sequentially. Moreover, the authors designed the scanning process in such a way as to minimize power consumption on the infected IoT. Each bot scans up to 10 IP address per second, allowing authors to scan the entire IPv4 address space in one day with only 4000 bots. At the end of the experiment, the botnet naturally died. (Sources: [17,18,28,29].)

Bashlite (6) – 2014

Description – Bashlite is also known as: BASHLITE, Gafgyt, Lizkebab, Torlus, LizardStresser, Qbot, Bash0day and Bashdoor, is a very large scale botnet that appeared in 2014. Later, in 2015 its source code was released, allowing anyone to create his own instance. Due to its public release the botnet was massively used by hackers and "script kiddies". Researchers believe that this botnet family infected over one million IoT. It was also responsible for large scale DDoS attacks (around 400 GBps).

Features – Binaries of the Bashlite family have been found for all CPU architectures used in IoT including ARM, MIPS, SuperSH etc. The botnet exhibits classical DDoS features: Syn and UDP flood. It uses a random scan and a performs dictionary attacks on telnet ports to infect its victims. It is also an IRC centralized botnet. (Sources: [10,17–19,26,30–37].)

Darlloz (8) – 2014

Description – Initially, the Darlloz botnet targeted exclusively Linux servers. However, it quickly evolved to target IoT, and infected around 31 000 such devices. The botnet was not able to perform DDoS attacks but was able to mine cryptocurrencies.

Features – This botnet targets numerous IoT architectures including ARM, MIPS, MIPSEL, PPC and x86. It uses both brute force attacks on telnet credentials as well as the exploitation of a CVE from PHP servers to obtain root access on the infected device. The malware uses a cryptominer module for Dogecoin and LiteCoin. Moreover, the malware searches for Aidra instance and deletes them. (Sources: [17,18,26,31,32,38].)

Spike (9) – 2014

Description – In 2014 researchers observed a new DDoS botnet originating from Asia, and initially targeted Linux OS. They then found a toolkit for this malware, named Spike, and derived from Hydra (1). This malware's toolkit contains a binary for the C2 server and multiple payload builders, allowing third parties to easily create a Spike botnet. Several different binaries were eventually found and several different names have been given to this malware including Dofloo, Mr Black, Wrkatk, Sotdas or AES.DDoS. All of these malware are so closely related that researchers believe they belong to the same family. Later the Spike malware evolved to also infect Windows computers. Researchers assume that the botnet originated from Asia, because the C2 servers interface is written in Mandarin Chinese.

Features – Spike is an IRC centralized botnet, aimed at performing DDoS attacks. One interesting feature of this malware is its ability to infect the /etc.rc. local file and survive a device reboot. Like its predecessors, the malware was compiled to attack multiple IoT architectures and was able to perform different types of DDoS attacks. (Sources: [10,18,39].)

TheMoon (10) – 2014

Description – Very few sources are available to study this malware. It seems to be a P2P botnet discovered in February 2014 by a researcher from SANS Internet Storm Center (ISC). This malware targets IoT and especially Linksys and Asus routers.

Features – TheMoon is a P2P botnet and uses a command execution vulnerability to exploit its victims. Unfortunately, the malware is little documented. We could only ascertain one specific feature: the malware uses a backdooring mechanism to ensure that other IoT botnet will not infect the same victim. (Sources: [18,40].)

Wifatch (11) – 2014

Description – The Wifatch botnet was a whitehat botnet. It was not aimed at performing DDoS attacks but rather at protecting IoT from other botnets. The authors explained their action in a GitLab post.[7] The binaries were made available but not the source code in order to avoid malicious uses.

Features – Wifatch was a P2P botnet exploiting weak credentials on telnet and SSH protocols. It was also compiled for multiple architectures. The worm was able to search for and delete all preceding malware, and write a message into the logs to warn the device owner and ask them to change the IoT's password. (Sources: [17–19,31,38,41,42].)

XOR DDOS (12) – 2014

Description – The XOR DDoS botnet is a small Chinese botnet, targeting Linux systems (IoT and servers). It was responsible for DDoS attacks against several Chinese's institutions and companies.

Features – The XOR DDoS botnet infects its victims through a brute force attack against SSH service credential. Once an adequate login is found, the botnet upload a bash script that analyzes the architecture of the victims. Then it drops a Trojan that infects several files and adds an entry to the CRON tab that downloads and installs the malware every three minutes. Moreover it adds an entry in the /etc/init.d/ directory in order to daemonize itself. The botnet was able to use several TCP Flood features and a DNS amplification attack. (Sources: [10,43–47].)

Elknot (13) – 2015

Description – As is the case for TheMoon (10) botnet, we have only limited sources that describe this botnet family. It seems that it had multiples names including: Spike, Dofloo, Mr Black, Sotdas and AES.DDoS. Researchers argue that all malware in this family are offspring from the Hydra DDoS tool, but that these malware developed novel features not present in the comparatively simple Hydra toolkit.

Features – Botnets from this family seem to have multiple DDoS features including Syn, UDP, ICMP, HTTP flood. They also have a DNS amplification feature. The Elknot malware is mainly compiled for MIPS and ARM architectures. This malware also has the ability to modify the CRON table in order to create a persistence mechanism, as described above. (Sources: [10,48].)

Remaiten (14) – 2016

Description – This botnet is a hybrid of the Tsunami and Bashlite botnets. It uses the DDoS protocol of the former, and the credentials dictionary of the latter. The Remaiten botnet was discovered in early 2016.

Features – This botnet is an IRC centralized botnet, aimed at performing DDoS attacks. It exhibits a modular design that allows it to spread other malware. Once it infects its target, it kills all other malware found on the victim. One interesting feature of Remaiten is its ability to detect the processor architecture of its victims, and transmit only the corresponding payload. Previous botnets upload multiple payloads onto their victim's system, and then sequentially attempted to execute each one until the correct payload is found. (Sources: [10,17–19,31,38,49].)

[7] https://gitlab.com/rav7teif/linux.wifatch/.

Hajime (15) – 2016

Description – This botnet was discovered by Rapidity Networks in early October 2016. It exhibits a 3-stages life cycle and many advanced features. Its life cycle is describe by Edwards and Profetis [50] in 2016. This life cycle consists in three phases, the first one is a recognition phase in which the botnet identifies a victim. The second stage is an infection phase in which a first binary, specific to the victim's architecture, is downloaded to the victim's system. In the final stage, the binary starts P2P communications and then downloads and executes other modules. Hajime does not appear to present any specifically malicious behavior and may be a type of whitehat Botnet, like Wifatch and Carna. Others speculate that it may be a botnet used to spread other malware. Hajime means "Beginning" in Japanese. The botnet was discovered a few days before Mirai. The malware has an update mechanism, which has allowed multiple evolution of the botnet to be spread.

Features – Hajime is a P2P botnet, and communicates using a protocol derived from BitTorrent and μTorrent. It exhibits multiple advanced features, including an update mechanism and several obfuscation and discretion features. It searches for and deletes other malware on the infected device, closes ports, changes its process name to TELNET, and makes use of exploits. After several updates, Hajime adopted the same credentials dictionary as Mirai, updated with the addition of two entries. It also uses an exploit for ARRIS modems called 'password of the day'. Moreover, Hajime tries different credentials depending on the banner of the object being attacked. At the end of 2018 Hajime was also able to exploit 6 CVEs and had been updated with the inclusion of a persistence mechanism. (Sources: [10,18,19,30,31,38,41,50–54].)

Mirai (16) – 2016

Description – Mirai is the most well known and the most heavily studied IoT botnet. This botnet is so famous mainly because of the large scale DDoS attacks it launched and the release of its source code on GitHub a few days later. It performed a DDoS attack against the French firm OVH that consumed more than one terabyte per second of bandwidth and a DDoS attack against the DYN DNS that degraded network services for millions of Americans on the East coast. Indeed, services such as Amazon, Twitter, etc. were unavailable for several hours for millions of customers.

After the release of the source code, several duplicates appeared, and in few month many offspring evolved from Mirai adding or modifying features. While Mirai infected its victims through the use of a credentials list, some variants make use of known exploits. For example, about a month after the appearance of Mirai, a new version was developed to exploit a flaw in around 900 000 Germans routers. Germans ISPs quickly reacted and patched their routers and the botnet disappeared. Most information about this malware originates with the work of Antonakakis et al. [7]. In our study, we excluded some papers that were only based on this source without adding new content about the Mirai botnet.

Features – The Mirai botnet family exhibits several new features that allowed it to create some of the largest DDoS attacks seen to date. It made use of advanced features such as DNS Water Torture, DNS amplification, as well as of much older features such as Syn or UDP Flood. As was the case for other botnets, binaries were found to attack several types of CPU architectures. But Mirai introduced two majors features: the weighted dictionary attacks (2.1.2) and the stateless scan (2.3.4). This botnet also uses the DGA algorithm (4.5.7) to hide its C2 servers. (Sources: [7,10,17–19,25,26,30–34,38,41,51,55–69].)

New Aidra (17) – 2016

Description – This botnet was discovered few days after Mirai, New Aidra (also known as Linux.IRCTelnet) is a fusion between the Aidra botnet, Kaiten, Bashlite scan and Mirai. This is also a DDoS botnet. New Aidra was able to attacks IPv6 IoT.

Features – Like its predecessors, New Aidra was compiled for almost all CPU architectures. The botnet was able to use every type of TCP-based and UDP flood DDoS attacks. It is an IRC centralized botnet. (Sources: [10,18,70].)

LuaBot (18) – 2016

Description – This botnet is the first one written in LUA and which use its own interpreter. It was discovered by the Malware Must Die team. It is a malware created to perform DDos Attacks.

Features – This malware is highly modular, the botmaster only has to send a LUA script to add new functionality. It also uses its own JS v7 engine to bypass the Cloudflare anti-DDoS attack. The Malware also uses HTTPS to communicate with the C2 server. (Sources: [10,30,51,71,72].)

Amnesia (19) – 2017

Description – This botnet was discovered by the Palo Alto unit 42 in 2017. Amnesia mainly targets TVT Digital DVR by exploiting a vulnerability present in all products crafted by this company. It was also designed to perform DDoS attacks.

Features – Like many other IoT botnets, Amnesia is an IRC centralized botnet, able to perform multiple DDoS attacks such as UDP flood or HTTP flood. Researchers found that the DDoS features used by Amnesia are very similar to those of Mirai. A new feature introduced by Amnesia is the ability to detect the presence of a virtualized environment and destroy it (by using a "rm -rf /*" command for example). This is a common feature in Windows and Android malware used to disturb malware analysis. (Sources: [18,25,31,38,73,74].)

BrickerBot (20) – 2017

Description – This botnet was quite original in its behavior and goals. Unlike of other botnets, BrickerBot was not self replicating in infected hosts. The botnet simply destroyed its victims. Researchers from Radware observed 3 different versions of this botnet that attempted to destroy their honeypots.

Features – This botnet is fully coded in BASH, and therefore is able to attack any Linux IoT that has a BASH interpreter, regardless of their CPU architecture. The script uploaded on the victim's system only serves to destroy the object. It writes random bytes on the memory, wipes all mounted volumes and finally deletes the entire file system. In later versions, the bot also launches small fork bombs. Finally most nodes launch their attack and scan using TOR nodes. As a consequence, researchers only observed the final payload, aimed at destroying IoT. We thus have only limited information about the network topology and the organization of this botnet. (Sources: [18,30,31,38,41,75–78].)

Reaper (21) – 2017

Description – IoT Reaper, also named IoT Troop, is one of the most dangerous IoT botnet. It was discovered in late 2017. It is also a DDoS botnet, and some researchers believe that this botnet is responsible for a 300 GBps DDoS attack against the financial sector in January 2018.

Features – As was the case for LUABot, Reaper is a modular botnet coded in LUA. But unlike other IoT botnets, Reaper does not rely upon credential attacks to exploit IoTs, opting instead to exploit nine different CVEs. A study in October 2017 could not find any DDoS capabilities, but later versions of the botnet seem to have acquired this feature. Also the botnet did not use an aggressive scanning technique, opting instead to remain discrete and avoid detection. (Sources: [38,41,79–84].)

Persirai (22) – 2017

Description – This botnet is an offspring of Mirai, but was sufficiently different from its progenitor that researchers found it appropriate to attribute it its own name. Persian characters were found in the binaries and named the botnet was accordingly named Persirai. It seems that this botnet was able to infect around 120 000 devices.

Features – Persirai mostly reuses Mirai's code, but additionally exploits a CVE targeting connected cameras, rather than performing a dictionary attack. It is able to launch DDoS attacks and was compiled for several architectures. (Sources: [51,85,86].)

Satori (23) – 2017

Description – Satori is also a variant of the Mirai Botnet observed in late 2017. It has infected around 170 000 IoT devices to launch DDoS attacks.

Features – Like Persirai, Satori is an offspring of Mirai that reuses much of the latter's code and features. Satori mainly targets Realtek and Huawei Home Gateway routers and uses multiple exploits in order to infect its victims. One variant of the Satori botnet targets Claymore's software instances in order to steal their Ethereums. Subsequently, an update of the Satori botnet added an additional exploit and researchers named this new version Masuta or PureMasuta. (Sources: [38,41,87–89].)

JenX (24) – 2018

Description – JenX was discovered in February 2018. It reuses some of the leaked code from Satori and BrickerBot. Its goal is to create a DDoS service targeting the game GTA online.

Features – JenX developed a new feature which allows it to DDoS game servers using the Valve Source Engine Query and the TS3 protocol. Moreover in order to scan and infect its victims, it uses a centralized server. None of the zombies participate in the scanning process. The goal of such an architecture is to remain discrete. (Sources: [90,91].)

TheMoon v2 (25) – 2018

Description – This botnet is a major evolution from TheMoon botnet (10). Researchers continued to monitor it for several years. The botnet activity was fairly small but in 2018 they observed the introduction of new payloads and of new features and exploits.

Features – TheMoon v2 botnet is highly modular allowing the botmaster to execute any kind of code on its victims' systems. This new version exploits 6 CVEs to infect home routers and connected cameras. The botnet is able to perform Ad fraud by navigating to a specified URL and click on ads. Moreover researchers observed the feature of offering to rent the botnet (called a Botnet as a Service or Baas scheme). TheMoon is able to use each victim as a proxy or utilize them to brute-force credentials. (Sources: [92–94].)

VPNFilter (26) – 2018

Description – The VPNfilter botnet was discovered by Cisco's Talos Unit. It is a highly complex botnet and is considered an Advanced Persistent Threat (APT). Some parts of its code reuses code from the BlackEnergy Malware [95]. The botnet was used to attack the Ukraine in 2018. The FBI stated that the botnet was crafted and controlled by Russia.

VPNFilter introduced several new features, mainly used for industrial spying. It was not designed to create large scale DDoS attacks but rather as weapon that could be use for cyberwar. In 2018, the FBI and Ukrainian authorities stated that they stopped an attack on a chlorine plant that used VPNFilter. Researchers believe that VPNFilter infected approximately 500,000 IoT in the world.

Features – Like many IoT botnets, VPNFilter is able to infect multiples CPU architectures. But its most interesting feature is its ability to monitor SCADA system (1.3.1.2), map all the local networks of a company (1.3.1.1) and create a reverse VPN (1.3.1.3) that allows botmasters to remotely connect and attack the internal network of a company. Moreover VPNFilter can also exploit and infect other IoTs in the local network. It is able to implement man-in-the-Middle attacks by redirecting all HTTPS traffic to HTTP and spy the communication. Researchers think that VPNFilter is able to detect a Windows exe file in clear traffic and to modify it on the fly to add viruses. VPNFilter is also able to encrypt data and exfiltrate it. It uses https and TOR to communicate with botmasters.

The VPNFilter malware is a multi-stage, modular worm with multiple capabilities to support both intelligence-collection and destructive cyberattack operations. The goal of the first stage is to establish a persistent foundation inside the victim's system. Stage 1 downloads and executes stage 2's code after each reboot. To download this second stage code, VPNFilter uses a very special kind of DGA Algorithm based on the metadata of photos uploaded on special websites. The stage 2 of VPNFilter is the main part of the malware, and contains most of the spying and vandalism features. Moreover the stage 2 downloads other modules that perform a subsequent stage of attacks. This final stage implements every attacking feature described above. Further details about the complex life cycle of VPNFilter are available on the Talos Blog [96]. (Sources: [41,96–102].)

Hide'n seek (27) – 2018

Description – The Hide'n Seek botnet was first observed in January 2018. Later, in October, several antivirus companies reported that it had been substantially updated. As was the case for Hajime, this botnet is a P2P botnet, not designed to create DDoS attacks. However, malicious activities have been observed in Hide'n seek botnets, notably spying behaviors.

Features – Hide'n Seek uses a custom P2P protocol to establish communication between peers. This protocol includes tricks that limit the analysis of the network by researchers. Each peer has a maximal number of peers in its list, in general 512 (customized to the peer's available memory), that is updated continuously. A full update of the list takes about 18 h. Researchers had to create a dedicated crawler to speed up the scan of the network, but they could not scan the quickly-evolving network fast enough to estimate its size with any certainty. Finally all communications and updates are signed with an ECDSA key, in order to prevent poisoning attacks. This allows only the botmaster to update each node, which will in turn diffuse the update thanks to the ECDSA signature. Moreover Hide'n Seek has binaries for every CPU architecture used in IoT, and identifies its victim's architecture in order to send only the corresponding payload. The botnet uses both a dictionary attack against telnet and SSH, as well as exploits CVEs to infect its victims. Hide'n Seek reuses code from both Mirai and Reaper: the scanner is taken from Mirai and some exploits are taken from Reaper. When it was first discovered and analyzed, Hide'n Seek did not have the ability to persist after a reboot of its victim. However, in October 2018 three majors features where added: first, Hide'n Seek acquired persistence thought a daemonization of the botnet. Second, the malware obtained the ability to use the ABD protocol to infect Android devices. Finally, it also acquired the ability to install the CoinMiner software and force its botnet to mine Monero cryptocurrency. Sources: [103–108]

EchoBot (28) – 2019

Description – Echobot is the final malware we analyzed in this study. It is the most recent we found when we searched for IoT botnets in late 2019. Echobot was first discovered by Palo Alto unit 42 in the beginning of June 2019. They observed a new variant of Mirai using multiple exploits to infect IoTs. A week later, Akamai observed that the botnet added 26 news exploits to its arsenal. In August 2019 Echobot was using around 59 exploits and in late December, it was observed using more than 70 exploits. Some of these exploits are very old and were already disclosed as early as 2003. This botnet targets every type of IoT directly connected to Internet.

Features – As stated above, this botnet is derived from Mirai. We could not find any sources that describe the attacks performed by Echobot; only the CVEs it uses to exploit IoTs. Because it is a variant of Mirai, we assume that Echobot exhibits every feature that Mirai already has, such as DDoS attacks and random scan. Moreover this botnet uses both dictionary attacks and exploits in order to infect a maximum number of devices. Echobot was compiled for all CPU architectures, uses exploits for cameras, DVRs, routers, visio-conferencing devices etc. It possesses some particularly interesting exploits. One was an OracleWebLogic exploit, that allows Echobot to also infect both Windows and Linux servers that use Oracle WebLogic software. Another exploit

targets the VMware NSX SD-WAN and allows unauthenticated command injection. The most recent exploit targets Mitsubishi smartRTU. These devices are industrial controllers used to manage SCADA systems. They are commonly used in the oil and gas industry. Such devices were not initially designed to be connected to the Internet and consequently commonly exhibit several vulnerabilities.

Echobot is also highly modular. Interestingly, researchers observed that some exploits were added to the botnet, and deleted the very next week. It is assumed that botmasters tried several exploits and only preserved the most efficient ones. This botnet seems to be evolving quickly and will likely become a serious threat in the near future. (Sources [109–115].)

Tables 12–14 provide a summary of the sources for each bot and a summary of the features of each bot, respectively.

6. Results

Armed with list of features for each botnets, we devised several graphs that visually highlight the evolution of botnets. The first such graph is the *Feature Propagation Multigraph*. We devised this graph in our previous study [9] in order to highlight the way features are borrowed between malware. Because of the considerable size of these graphs, we placed high resolution versions of each graph, as well as the algorithms needed for generation and GraphML codes in a GitHub repository.[8] We strongly encourage readers to consult this GitHub repository and examine high-resolution versions of the figures in the paper. The GraphML codes can be used in conjunction with an app such as YED[9] to experiment with different layouts and glean a better understanding of the figures.

6.1. Feature propagation multigraph

In the Feature Propagation Multigraph (FPM), each malware is represented with a black circle with a radius proportional to the number of features it has. We then add a colored circle if it is the first malware that uses a feature. Each feature is associated with a different color. Finally we draw an edge of this color from this malware, to all other malware that implement the same feature. We draw a different graph for each of the four families of features defined in Section 4. Figs. 22 to 25 present the feature propagation multigraphs for the goal family, the infection family, the organization family, and the efficiency family, respectively.

Unfortunately, the complexity of these graphs increases with the number of features. Still, we can easily observe that some features are created quite early, by the first botnets (Hydra, PsybOt etc.) and are shared across multiple malware. Two interesting advantages of our representation are the ability to quickly zero-in on the most complex malware (with the largest black circles) and the most innovative malware (with the highest number of colored circle). Moreover, we can easily understand which features are shared and reused over the time. Furthermore this representation allows us to infer sub-families of malware. For example, in Fig. 24 we can easily distinguish a family of botnet that uses a centralized architecture and a family of botnet that uses a P2P architecture. Likewise, we can delineate the family of botnet using a custom protocol to communicate by following the cyan edges in the same figure.

[8] https://github.com/bvignau/Softawre-Phylogenic-classification.git.
[9] https://https://www.yworks.com/yed-live.

6.2. Feature usage over time

Next, we study the usage of features over time. We use a variation of the Gantt representation, in which a feature is analogous to a task beginning on January 1 of the year of its first use by a botnet and ending on January the 1 of the year of its last recorded use by a botnet. The date of January the 1 is arbitrary, we seek to indicate the years during which each feature was in usage. We wish to emphasize the fact that certain features are used by botnet for a span of several years, while others are introduced by a botnet and never used again afterwards. For those features, the task starts on January the 1 and ends on June the 1. This visual representation allows to quickly identify which features persist over time and which ones die out. As was the case for the Feature Propagation Multigraph, we draw a separate Gantt diagram for each feature family. Fig. 9a to 6.2 present the feature propagation multigraphs for the goal family, the infection family, the organization family, and the efficiency family, respectively.

An analysis of Fig. 9a to 6.2 shows that botnets created between 2008 and 2014 used relatively fewer functionalities compared to botnets created afterwards. This is shown by the fact that there are more features that "start" after 2014 in the Gantt diagrams. However, these features are then subsequently used continuously for several years. The FPMs show that the features that were used in this time period are also widely spread across all botnets. For example, DDoS attacks with Syn Flood, or targeting MIPS architectures were both introduced as early as 2008, and remain the widely used until the end of our study in 2019. Looking at the infection FPM and Gantt diagram also shows us that most of the essential features that allow a botnet to target different CPU architectures and infect a large number of victims were created early and rapidly gain widespread adoption. Later features focused on developing novel attack strategies. It remains to be seen if these features will gain widespread adoption, or remain localized to a small number of targeted botnets.

We also find that some specific features are not frequently used. For example the ping flood DDoS attack is used by only 3 botnets, while the cryptomining features are used by only two botnets. Industrial spying features are also rarely used. Most of the spying features were introduced by VPNFilter in 2018 and used only by this malware, with most other botnets using DDoS features.

On the other hand, in the goal features FPM, we observe that Mirai and VPNFilter were the most innovative malware, but we can also see that 6 other botnets, all of which appeared between 2016 and 2019, introduced at least one new functionality. During this same period, only 1 Organization feature and 4 Efficiency features were introduced. Finally, the Echobot botnet is the more recent botnet in our sample, with the highest number of features(29). However, it does not introduce any new features. This is partly due to a design choice of our taxonomy: we do not represent each vulnerability as a distinct feature. Had we done so, Echobot could have been seen as a highly innovative bot, since it was the first to make use of several CVEs. Still, it is interesting to note that aside from the exploitation of previously used vulnerabilities, each of the 29 features used by Echobot was previously introduced by other bots.

Finally we computed the total number of features and the total number of exploits use by each botnet. Results are given in Fig. 10. As can be seen, both numbers tend to increase over the time, showing an increase in the complexity of botnet over the time. However, this observation does not hold for every botnet, and also observe some recent botnets with comparatively few features. Indeed, most of the offspring of Mirai have fewer feature than it does. We also observe that starting in 2018, the average number of features in botnets does not increase much, but the use of exploits shoots up as recent botnets tend to include as many exploits as possible (see Fig. 8).

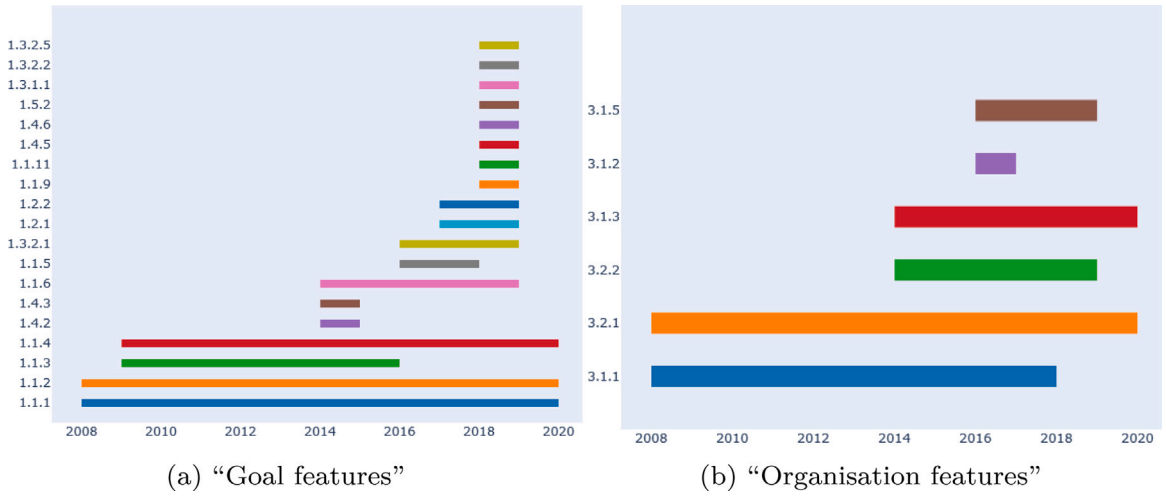


Fig. 8. Use of "Goal features" and "Organization features".

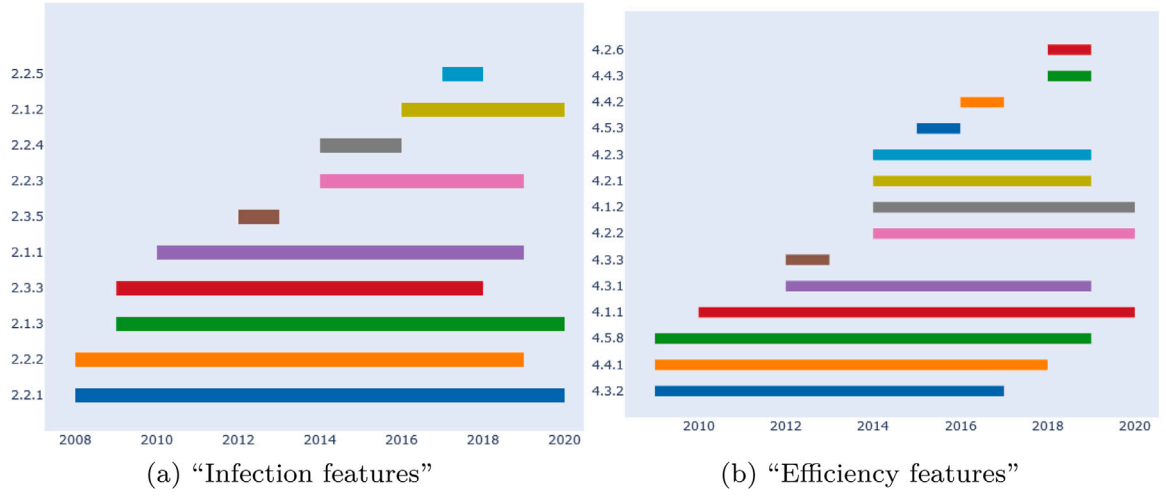


Fig. 9. Use of "Infection features" and "Efficiency features".

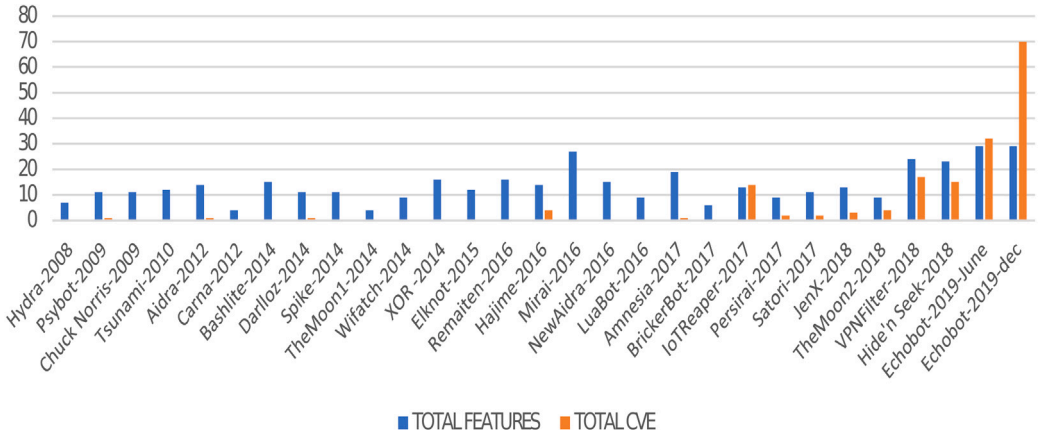


Fig. 10. Total features and exploits for each botnet.

6.3. Vulnerabilities usage over time

In this section we examine the use of distinct vulnerabilities over time. Most of these vulnerabilities are referenced as CVE but some of them are not. These vulnerabilities allow the attacker to gain root access to the IoT devices, or to execute arbitrary code on the vulnerable device. In Fig. 26, we list every vulnerability used by botnets in our sample and draw the corresponding FPM. In the case of Echobot we only include the 32 vulnerabilities observed in the initial version of the botnet. As stated in Section 5, this botnet was frequently updated to add and remove exploits and by the end December 2019, various versions included more than 70 different exploits.

Table 6
Botnet exploits.

Botnet	Year	# Exploits	Oldest	Newest
Psybot (2)	2009	1	2009	2009
Aidra (5)	2012	1	2009	2009
Darlloz (8)	2014	1	2012	2012
Hajime (15)	2016	4	2015	2018
Amnesia (19)	2017	1	2013	2013
IoT Reaper (21)	2017	14	2009	2017
Persirai (22)	2017	2	2017	2017
Satori (23)	2017	2	2014	2014
JenX (24)	2018	3	2014	2017
TheMoon 2 (25)	2018	4	2018	2018
VPN Filter (26)	2018	17	2009	2018
Hide'n Seek (27)	2018	15	2009	2018
EchoBot (June) (28)	2019	32	2009	2019

Table 6 charts the use of vulnerabilities over time. It lists every botnet that exploits vulnerabilities, and for each, the year of release of the botnet, the number of vulnerability it uses, the year of its oldest exploited vulnerability and the year of its most recent one. From this table, we draw a Gantt diagram, presented in Fig. 11, that includes every vulnerability that is used by at least two different bots. From this figure we can see than only one vulnerability (the D-Link bypass exploit) used in 2009 is still in use again 10 years after. Most of the exploits are used by only one botnet (they do not appears in the graph) or two. Yet even this low level of vulnerability reuse is somewhat disconcerting. In a security-conscious world, in which disclosed vulnerability are promptly patched, one would expect almost no vulnerability reuse to occur, since any vulnerabilities would rapidly become obsolete. In that sense, any vulnerability reuse is a sad commentary on the state of IoT security.

To glean a better understanding of the propagation of exploits, Fig. 26 presents a Feature Propagation Multigraph using every vulnerability as a feature. We can observe that the D-Link bypass exploit is the exploit that is most widely spread. Moreover IoTReaper, VPNFilter and Echobot are the most innovative malware, since they each introduce multiple new exploits. Also we can see that Hide'n Seek uses most of the exploits introduced by IoTReaper. Out of the 31 exploits used by Echobot only 4 were previously introduced by other botnet. Every one of the vulnerability exploits used by malware in our sample set was already publicly disclosed at the time of its adoption by malware developers.

7. Interpretation

All of these results allow us to highlight the presence of an evolutionary phenomenon comparable to the evolution of species. Indeed, new features are constantly created. Some of these features will gain acceptance and spread to other bots while other will fail to catch on, a process not unlike natural selection.

This evolution can be seen in each of the feature families we studied. Initially, the infection process consisted in comparatively simple brute force attacks, before slowly moving to the more targeted exploitation of publicly available vulnerabilities. The choice of exploits is the aspect of malware that evolves the fastest. The Echobot botnet is a showcase representation of this phenomenon: each update adds new exploits and deletes obsolete ones.

We also observe that botnets gradually developed payloads that allowed them to infect multiple different CPU architectures simultaneously. Every feature used to target a new CPU architecture spread quickly to all botnets. We believe that this phenomenon is due to the fact that such features are vital for a botnet as they improve its efficiency by increasing the number of potential targets. Moreover we observe that botnets gradually increased their use of efficiency features over the years. These features can be hard to develop but are quite useful to a botnet. We also observe that early bots scanned the Internet in order to search for victims using simple sequential scans, in which each bot will try to infect every IP addresses in increasing order. This feature was quickly replaced by the much more efficient random scan. In the next section we examine these scanning strategies in more detail.

Changes in the goals of botnets also reveal an evolutionary pattern. Most of the botnet studied are designed to create large-scale DDoS attacks. Each new botnet reused most of DDoS attack types that its predecessors used. A few botnets were used to mine cryptocurrencies, probably due to the low computing power of IoT devices. Since 2018 botnets introduced new goals: Botnet as a Service (BaaS), industrial spying and ad fraud. The industrial spying features introduced by VPNFilter show us that IoT botnets can be a useful cyberweapon for nation states. We believe that this feature will remain predominantly be used by state actors, due to its high difficulty of implementation. However, the BaaS and ad fraud feature may grow more present in coming years, since these features allow botmasters to quickly monetize their botnets. Moreover such features are much more discrete than DDoS attacks and thus do not attract the attention of the authorities.

The description of every botnet given in Section 5 allow us to answer the RQ1 (What features define malware M_x) and RQ2 (What are the goals of the botnet B_x). The answer to RQ2 consists in the features from the *Goal family* of a botnet, and the answer to RQ1 is found in every other feature. Tables 13 and 14 in annex provide the list of every features for each botnet. The same data is also available in a CSV file in our GitHub repository.

These results provide answers to RQ1 and RQ2 and lead us to formulate three hypotheses that explain why a feature will be reused by future botnets:

- A feature will be widely adopted and reused if it aids the botnet to achieve a defined goal.
- A feature will be widely adopted and reused if it improves the efficiency of the botnet.
- However, an otherwise useful feature may not be widely adopted and spread if its implementation exhibits a high degree of complexity and necessitates advanced technical skills.

Of course, while these hypotheses generally explain the patterns of feature propagation that we have observed, other factors may also be at play in some cases.

To answer RQ3 RQ4 and RQ5, we examined our sources to determine the number of IoT each botnet infected, the timeframe (duration of time) that the botnet required to infect this number of IoTs and the peak size of the DDoS attacks they performed (measured in the amount of bandwidth the attack consumed). The size of several botnets is shown in Fig. 12 and the peak DDoS sizes are given in Fig. 13.

We chose to use the peak DDoS size because most botnets perform DDoS attacks and it is the most germane information we could find to quantify the damages caused by a botnet. Comparing the direct economic costs of each botnet might have been a more accurate metric, but unfortunately, we were unable to obtain sufficient information to estimate these costs. Botnets that aim at goals other than DDoS (e.g. cryptomining) are naturally excluded from this analysis.

Reliable data about timeframes was not available for every botnet. The most accurate information we had are for Mirai and Carna botnets. Data for the Carna botnet is taken from the botmaster report [28]. This report states that the Carna botnet was able to infect about 420 000 devices in the course of one night in December 2012. Considering that each bot uses limited resources in order to not disturb the infected IoT, the propagation could have been faster. For other botnets, we were able to obtain data from the network telescope [116] or the black hole monitoring system [117], but these techniques only record the number of scanning bots, not the total number of bots. In general, a botmaster will not affect the totality of the bots at his disposal to the discovery of new potential victims. Indeed, the majority of the bots will be used to achieve the immediate goal of the botmaster (DDoS etc.). In general, the botmaster will recruit most of his victims during an initial recruitment phase, and then affect most of his resources to achieving

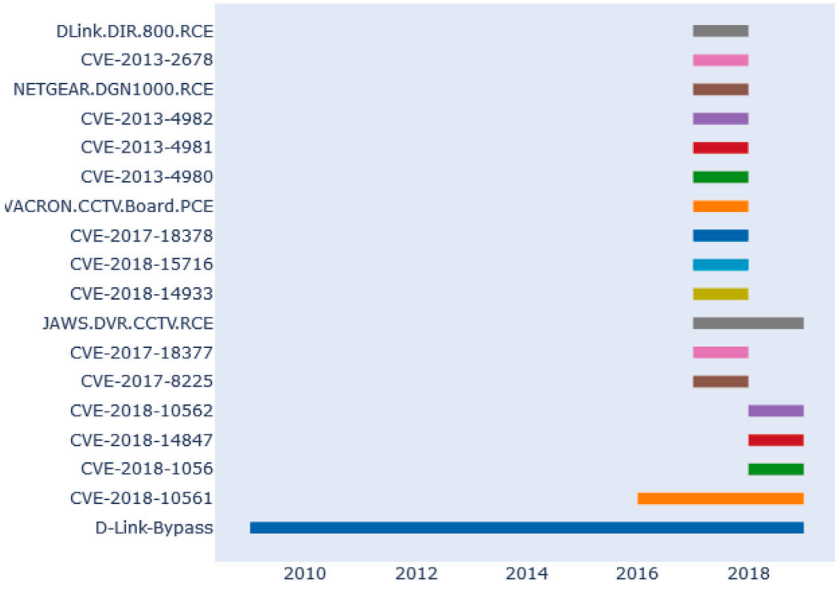


Fig. 11. Use of vulnerabilities over time.

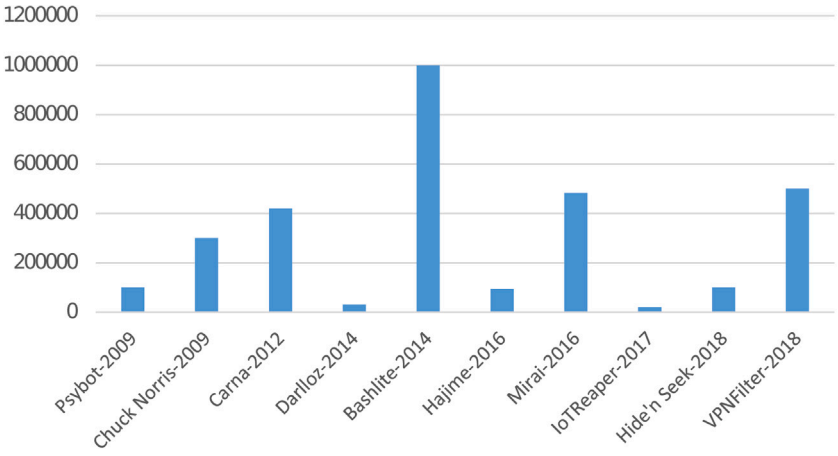


Fig. 12. Botnet size, in number of victims.

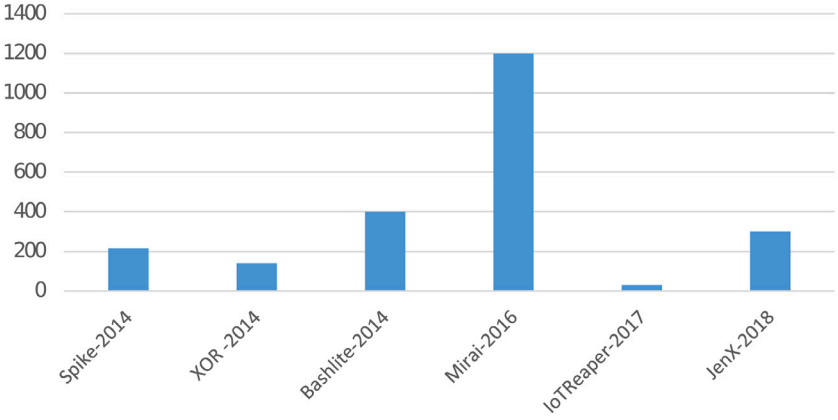


Fig. 13. DDoS Peak of botnets (in Gbps).

his goals, leaving only a small number of bots to continuously scan for new victims or re-infect lost bots.

Another way to estimate the botnet size is by counting the number of differences sources participating in a DDoS attacks. A limitation of this technique is that botmasters often rent out the DDoS power of their botnet, creating several different "small" DDoS attacks. This phenomenon makes it difficult to determine the real size of a botnet and the time it requires to infect its victims. Moreover for most of the botnets, the timeframe given in our sources for a number of victims is quite long, between one and two years. However, for some botnets, the

Table 7
Botnet DDoS attacks.

Botnet	Size (# victims)	Time frame	Damage (GBps)
Psybot (2)	100 000	2009	Unknown
Chuck Norris (3)	300 000	2010–2012	Unknown
Carna (6)	420 000	One night	None
Bashilte (7)	1 000 000	2014–2016	400
Darlloz (8)	31 000	2014	Unknown
Spike (9)	Unknown	Unknown	215
Wifatch (11)	60 000	2015	None
Hajime (15)	95 000	2016–2018	Unknown
Mirai (16)	483 000	One month	1200
Reaper (21)	20 000	Unknown	30
VPNFilter (26)	500 000	6 months	Unknown
Hide'n Seek (27)	100 000	One month	Unknown

timeframe is quite low, for example, the Mirai botnet was able to recruit around 100,000 devices in a single day. This variability indicates that the duration of the recruitment phase of a botnet can vary widely, making this metric only an approximate measure of the actual size of a botnet.

Due to lack of data, we were not able to fully answer the RQs3-5, and leave them open for future research. We are hopeful that future research by industry will be able to answer these questions in a few years as more data is gathered (see Table 7).

Our results allow us to provide an initial answer to our Main Research Question (MRQ): *How are IoT botnets evolving?*. We see that IoT botnets tend to become more complex and more efficient over the years since it is easy to add new features. Existing features can be easily reused, especially in the case of open source botnets. We also saw that over time, botnets moved away from dictionary attacks and tended to exploit vulnerabilities instead. This phenomenon will likely become more prevalent in coming years as vendors move away from devices with weak default credentials. The methods used to infect new victims are evolving faster than the kinds and goals of the attacks (DDoS, spying etc.).

Another interesting phenomenon is the use of exploits that target Android, Linux and Windows servers. This leads us to believe that the next few years may see the rise of "mega-botnets", able to simultaneously exploit IoT, personal computer and professionals servers. Moreover, until 2018 most of the infected IoT were routers and connected cameras, but since 2018 other types of objects have become the target of exploitation, including as industrial SCADA controller and even yacht control system (by Echobot). This development will surely spark the interest of malicious state actors, since it makes botnets an even more effective cyberweapon. More generally, we can summarize the evolution of IoT botnets as: more complex, more dangerous and more versatile.

8. Predicting the future evolution of IoT bots

In the previous sections we provided answers to the RQs raised in Section 3 and argued that one possible explanation as to why some features are widely propagated while other are not is the efficiency gains brought on by the feature. If a new feature appears that serves the same purpose as another existing feature, but accomplishes the task with more efficiency, then the new feature will gradually replace the old one. The gradual replacement of sequential scans with random scans is an illustrative example of this phenomenon.

In this section, we propose a model to study the spread of IoT bots. We posit that this model will allow researchers to better predict which features will be more widely adopted by the community of malware designers, by making it possible to simulate the spread of malware under different conditions. In particular, this model allows us to simulate the competition between several botnets, which, as documented in the previous sections, is an important reality of IoT bots.

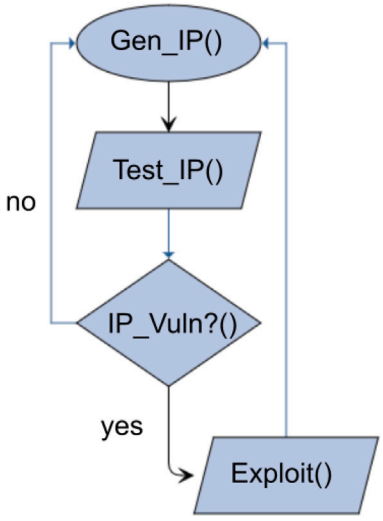


Fig. 14. General Bot state machine.

This tool can aid researchers in predicting the efficiency of specific features, or combinations of features, in order to identify the most efficient bots.

Because the spread of a botnet is analogous to the spread of a disease, we first turn our attention to epidemiological models.

8.1. Epidemiological modeling tools

As stated in Section 2, one of the main metrics used to measure the power of a botnet is its size. Since the number of potential victims is bounded, a given botnet will surpass another one if it can infect its victims in a shorter time span. In this context, epidemiological models help us glean insights about the spread of malware over time. We study four different epidemiological models of malware, each of which is based upon medical models and was used to model the propagation of older botnets such as Code Red or Morris [118]. In order to compare these models with our proposed model, we define 3 criteria. First is the simplicity of the model. We consider that a model is ***simple (1)*** if it combines multiples factors in a single value in order to simplify computation. The second criterion is the ***extensibility (2)*** of the models. A model is extensible if it is easy to add and describe a new functionality without the need to noticeably change the model. Conversely, a non-extensible model requires the introduction of complexity in order to describe a new behavior of a botnet. The last criteria is the ability of a model to describe ***competition (3)*** between botnets. It is a peculiar reality of IoT botnets that they compete for the same limited pool of potential targets, often using the same attack vectors (common dictionary, common exploits, multiples version of the same botnet etc.). Since other classes of malware do not usually compete for targets in this manner, this aspect of botnets' spread is not always included in epidemiological models.

S.I and S.I.S model

We first consider the Susceptible Infected (S.I) model introduced by Kermack et al. in 1927 [119]. In this model, each member of a population is assigned state, either Susceptible or Infected by the disease. The S.I.S. model [120] incorporates the possibility of reinfection by returning an individual who has previously recovered from the disease to the Susceptible state, where they could potentially be infected again. While originally proposed to study the spread of biological diseases, both of these models have been adapted to study the propagation of worms that employ a random scan strategy [118,121,122]. Both models assume that a defined proportion of the population is vulnerable

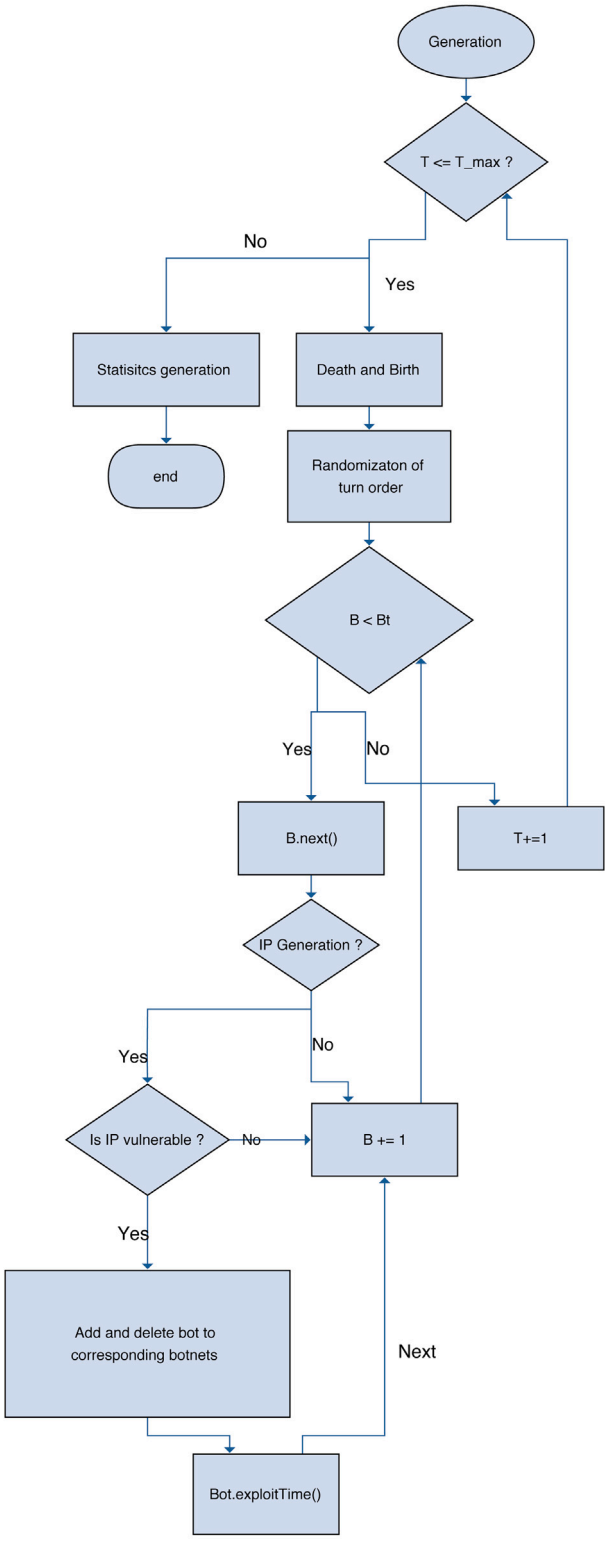


Fig. 15. Botnet infection process simulator.

to infection. Each newly infected victim will attempt to randomly infect other computers. The two models use differentials equations to abstract and describe behaviors of botnets. Several behaviors, such as the infection rate, are abstracted to a single value.

Markovian model

This model was created by Chen et al. [123] to model the topological propagation of botnets, in which bots attempt to infect computers on their own network. The authors show that the S.I and S.I.S models are often inaccurate when modeling this type of behavior. They claim that a proper modeling of topological propagation require us to specify a different infection rate for each node, something which is difficult to do when using S.I models. As an alternative, they propose a model based on Markovian chains, where each node has its own chains, with a distinct probability to infect other victims and a distinct death rate. The models is extensible but, as is the case for the S.I. model, it simplifies multiple behaviors to a single constant.

AAWAP model

The Analytical Active Worm Propagation (AAWAP) was created by Chen et al. [124] and is based on the propagation of the worms Morris, Code Red and Nimba. Unlike the previous models, the AAWAP model uses discrete time, instead of continuous time. The state of the population at time t is used to compute the state of the population at a time $t+1$. Moreover, this model takes into account the patching process, by introducing a patching rate. This model can also accommodate several types of scans. However, authors combine multiple scanning features in a single *scan and infection rate*.

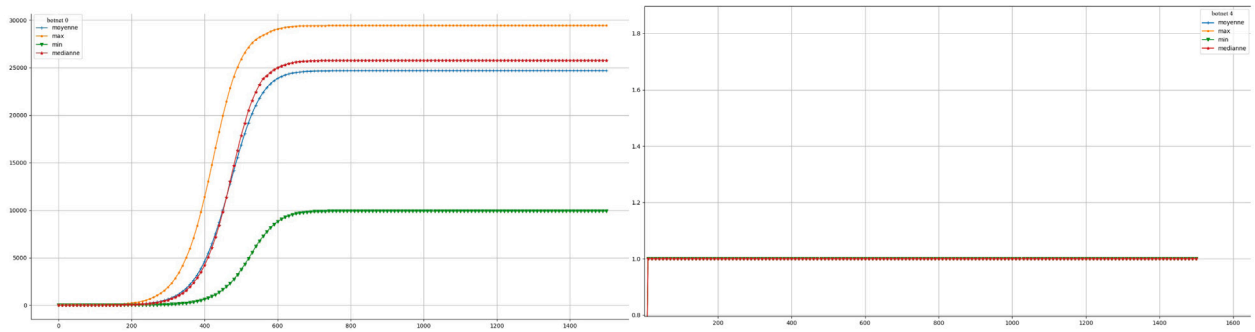
8.2. Game theory tools

The game theory model makes it possible to define the behavior of both botmasters and defenders in different situations [125]. They generally reach a Nash equilibrium between two factions. For example, Soper and Musacchio [126] model the probability of a defender detecting the infection on his computer, depending on the behavior of the botmaster. Bensoussan et al. [127] model the behavior between an attacker and a defender. The attacker wishes to infect a maximum number of computers while the defender seeks the opposite outcome. The infection model used is the S.I.S and the authors compute the Nash equilibrium of this game using the differential equations of the S.I.S model. These models do not provide more information on the infections process than the previously studied models.

8.3. Simulation tools

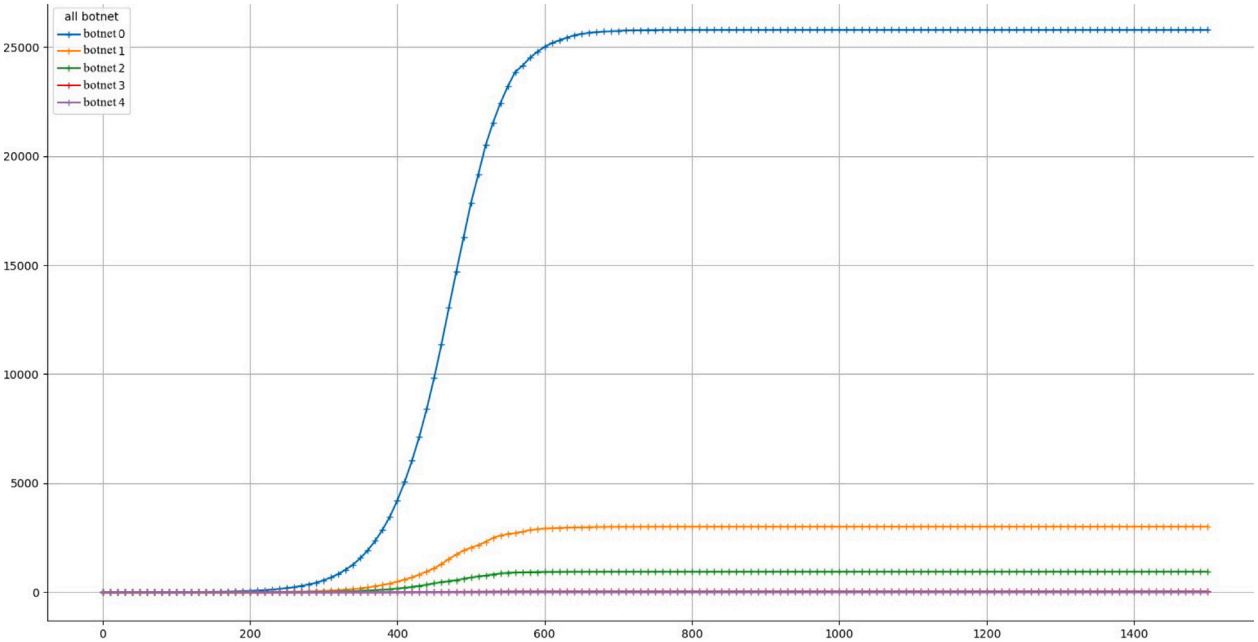
A few botnet attack simulation tools have been developed in the last few years [128–130]. These tools are agent-based simulation tools and often model low-level network behavior of the bots. Each agent can communicate with the environment and with other agent. The research teams implemented every network protocol used by agents and botnets to communicate. They generally use two teams to simulate attacks and defense, where the attacking team seeks to DDoS the defensing team. The goal of such simulation tools is to predict the expected size of DDoS attacks. Other tools such as the Leaf cybersecurity training platform [131] also exist but are beyond the scope of this paper.

Such tools are focused on simulating attacks and responses from the two teams after the creation of a botnet, while our tool is more focused on the infection process. Moreover, these tools are also restricted on DDoS attacks, to the detriment of other goal and do not allow for a study of different infection strategies. Furthermore, if a new communication protocol is created, it will need to be implemented in the tool before it can be studied. Our tool, on the other hand proposes a simpler simulation process by omitting implementation details such as the full network stack. This allow us easily simulate the behavior of a botnet population with 10 000 nodes, while the tools mentioned above are restricted to much smaller populations of about 300 nodes.



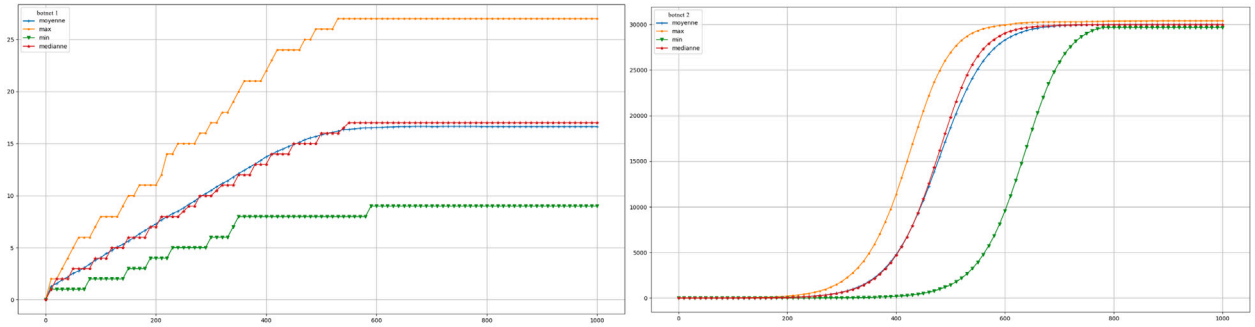
(a) Botnet #1 (b) Botnet #2

Fig. 16. Evolution of the size of botnets over 1500 turns (Experiment 1). The blue line correspond to the mean value of the botnet population over all the simulation, the orange one to the maximum, the green one to the minimum and the red one to the median size of the population.



(a) All

Fig. 17. Comparison of the evolution of the size of botnets over 1500 turns (Experiment 1). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

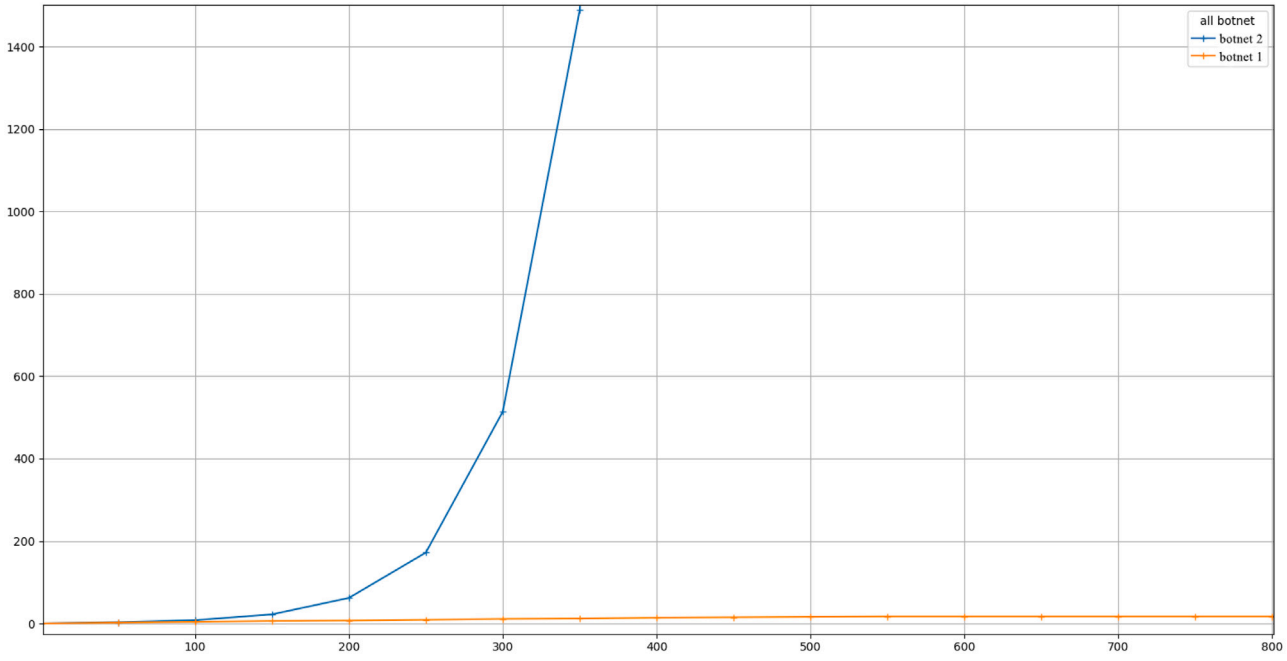


(a) Botnet #1 (b) Botnet #2

Fig. 18. Evolution of the size of botnets over 1000 turns (Experiment 2). The blue line correspond to the mean value of the botnet population over all the simulation, the orange one to the maximum, the green one to the minimum and the red one to the median size of the population.

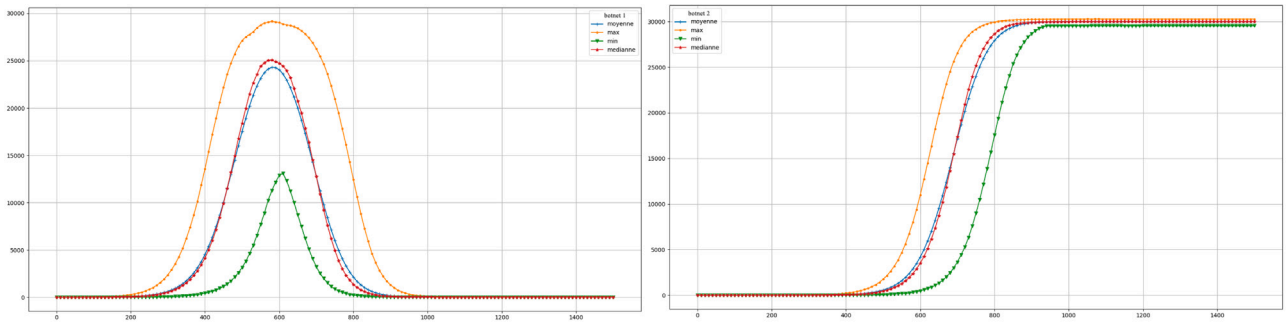
8.4. Our model: Botnet infection process simulator

All the models we studied abstract away the different scanning features, reducing them to constants such as *scan rate*. Moreover they are not suitable to model the competition of multiples botnets that try to infect a shared set of the potential victims. Thus we develop a new game-like model in which at each turn, each bot will perform a



(a) All

Fig. 19. Comparison of the evolution of the size of botnets over 1000 turns (Experiment 2).



(a) Botnet #1 (b) Botnet #2

Fig. 20. Close-up of the evolution of the size of all botnet over 1500 turns (Experiment 3). The blue line correspond to the mean value of the botnet population over all the simulation, the orange one to the maximum, the green one to the minimum and the red one to the median size of the population.

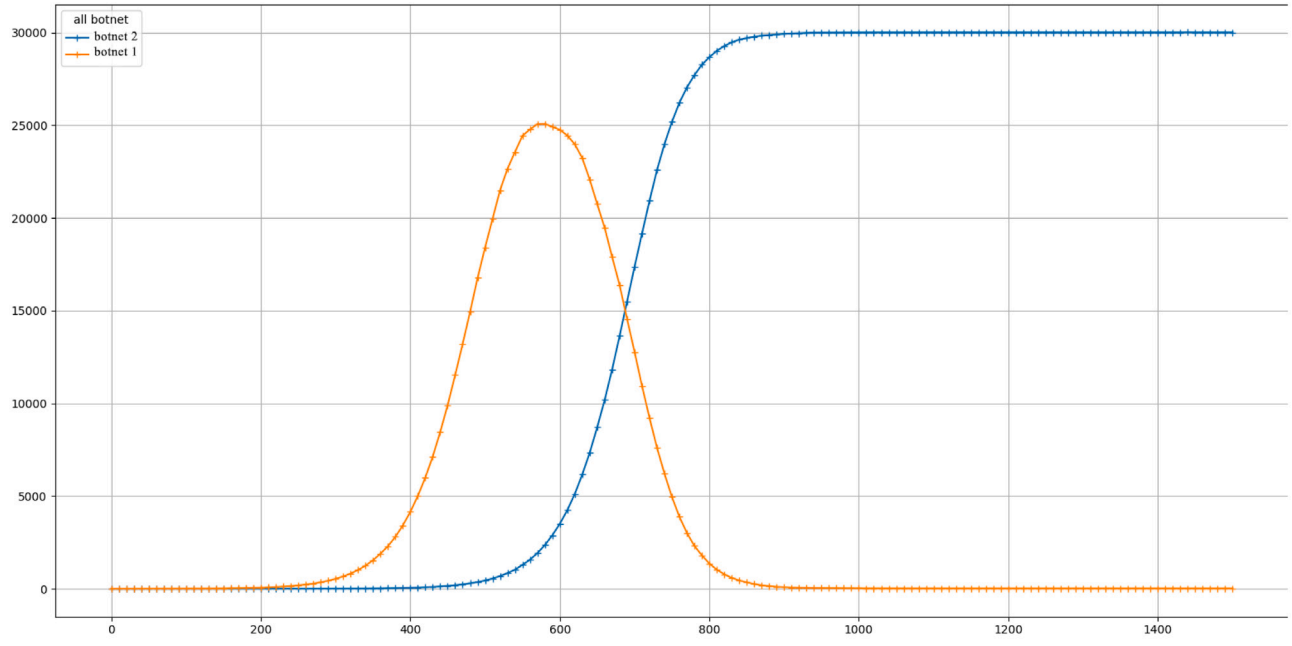
task, following its own state machine. We run multiple simulations and observe global tendencies and borderline cases.

We begin by defining the life cycle of each botnet using a state machine. Each task corresponds to a single state. To abstract time in a flexible manner, we define the number of turns taken up by each state of the finite state machine. This abstraction can then be translated into real time units by assigning a conversion rate (e.g. 1 turn = 0.5 ms). The number of turns for a state can be constant or defined by a preset function. Using a function allows us to incorporate into the model important parameters such as the latency, or congestion of the network, etc. Some states are mandatory, such as scan and exploitation. The scan state must generate an IP address, but the algorithm used to do so is not specified, and can be any known method (random, sequential, distributed sequential etc.). The exploitation state models the time taken by a bot to infect a victim (upload the payload etc.). There is no limit to the number of state of a bot, and each state can be composed of multiples functions, algorithm etc. The general state machine of a bot is given in Fig. 14.

Before starting a simulation, we first generate a set of victims. This is done using a configuration file that contains (1) the total size of the population of targetable devices, (2) the proportion of this population that is vulnerable to infection by each bot present in the simulation (e.g. 30%) and (3) the proportion of potential victims that are common between multiples botnet.

At the onset of the simulation, each botnet disposes of a single infected bot (the first one created by the botmaster, which is not part of the total population). Upon infection, each bot will initialize and run its own finite state machine and transmit to the gamemaster either an IP address or a message indicating that the bot is processing (this could happen, for instance, if the bot requires multiple turns to generate an IP address). If a bot provides an IP address, the gamemaster will verify if the corresponding device is tagged as a potential victim of the botnet. If so, the exploitation phase will begin at the next turn. A bot may also patch victims, protecting them against exploitation from other botnet, or remove other botnets from already infected devices (as Wifatch or Hajime do).

At the end of each turn, the order in which the bots will play the subsequent is randomized. The game also sets a birthrate and a death rate, that capture the frequency at which new devices enter the population. The death rate and birthrate are defined by a frequency (one occurrence every X turns) that follows a normal law with μ, X and σ defined in the configuration file. Each new IoT added to the pool will



(a) All

Fig. 21. Comparison of the evolution of the size of all botnet over 1500 turns (Experiment 3). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 8
Comparison of different malware epidemiological models. Si (1) = Simplicity; Ex (2) = Extensibility; Co (3) = Competition.

Model	Si (1)	Ex (2)	Co (3)	Type
S.I	✓	✗	✗	Deterministic
S.I.S	✓	✗	✗	Deterministic
Markovian	✓	✓	✗	Stochastic
AAWAP	✓	✓	✗	Stochastic
Our model	✗	✓	✓	Stochastic

randomly be assigned as vulnerable or not, following the probabilities set in the configuration file. The state machine of our model is given in Fig. 15.

The model we propose is highly extensible, since the number of states in the FSMs that captures the malware's behavior is not limited, nor is the ability to use any kind of function or algorithm in each phase. Moreover, we did not incorporate multiples behaviors in a single parameter, thus avoiding thorny simplifications. Finally, by design, our model is made to simulate the competition between botnets. Our model is the only one able to do this. Table 8 presents a summary of the characteristics of the models we studied.

9. Simulations

Using our model, we simulated several scenarios in order to observe the impacts of different features on the size of the resulting botnet. In this section, we present 3 experiments that illustrate the competition between similar botnets, the competition between botnets that utilize different scanning strategies, and the competition between botnets using the same strategy, but where one can suppress the other. The source code, alongside with the raw data for these experiments, as well as other experiments that we elided out of space considerations, other are available on the author's repository.[10]

[10] https://github.com/bvignau/The_Botnet_Game.

9.1. Experiment parameters

Our model allows us to ascribe different behaviors for each bot, as well as global parameter such as birthrate and death rate that describe the ecosystem of IoT devices. In this paper, because our focus is on the features of the botnets, we used constant values for these parameters, namely $\mu = 5$ and $\sigma = 2$, and $X = 150$ turns for the birthrate and $X = 200$ turns for the death rate. These values capture the intuition that the number of IoT connected devices is slowly growing. We further assume an initial population of 100 000, of which 30% are vulnerable. In each simulation, we run the model for as many turns as is needed to reach an equilibrium point. Each experiment consist of 100 simulations. We show minimum, maximum, mean and median of the size of the bot's population.

Experiment 1

In the first experiment, we seek to observe the competition between 5 replicas of the same botnet. This situation occurs when the source code of a botnet is released, and multiple replicas of this botnet are created within a short time span. Because they are replicas of the same botnet they are each able to immunize victims against each other. This process is quite frequent, since botnets generally implements a system to avoid two infections of the same object. We ran the simulation for 1500 turns, with a total population of 100 000 computers, 30% of which are vulnerable to every botnet. The simulation usually reaches equilibrium after about 800 turns. The simulation uses 5 replicas, each starting at a different turn. The main parameters are given in Table 9.

Experiment 2

This second experiment was designed to observe the impact of two different scanning strategy on the size of the resulting botnet. The first strategy we implemented is the sequential scan, as used by Hydra and Psybot. When using this scanning method, each bot will scan the entire IP space, starting at the beginning and scanning in increasing order. The second strategy is the random scan, used by most of the botnets in this study. The experiment ran for 1000 turns. Because creating a random number uses more computational resources and time than simply incrementing an integer, we decrease the time taken

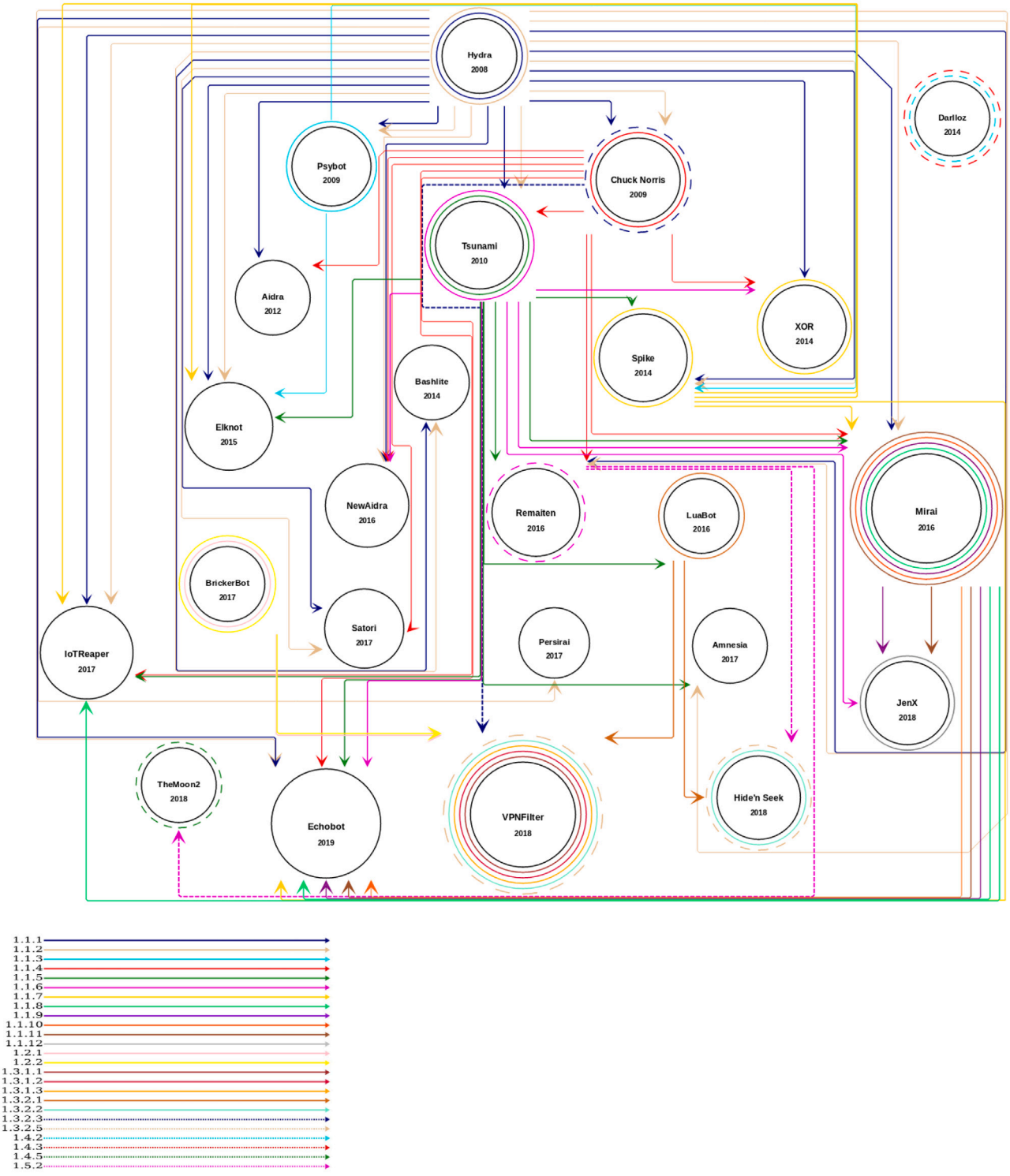


Fig. 22. Goal feature propagation multigraph.

to generate an IP address by the sequential scan. Since we want to observe only the impact of the scan strategy, each botnet immunizes its victims against every other bot, but none of the botnet are capable of removing other bots from already infected devices. The two botnet begin their operations at the same time. The main parameters are given in Table 10.

Experiment 3

The final experiment aims to observe the impact of the immunity and suppression of other botnet on the spread on malware. We used two botnets, with identical parameters, except that the second one has the ability to suppress the first one in an infected victim. Furthermore it immunizes the victim against the first botnet. Such a bot thus models a behavior similar to the one presented by Wifatch, Mirai or Hajime, that identify other older IoT botnets and delete them. In this experiment, the

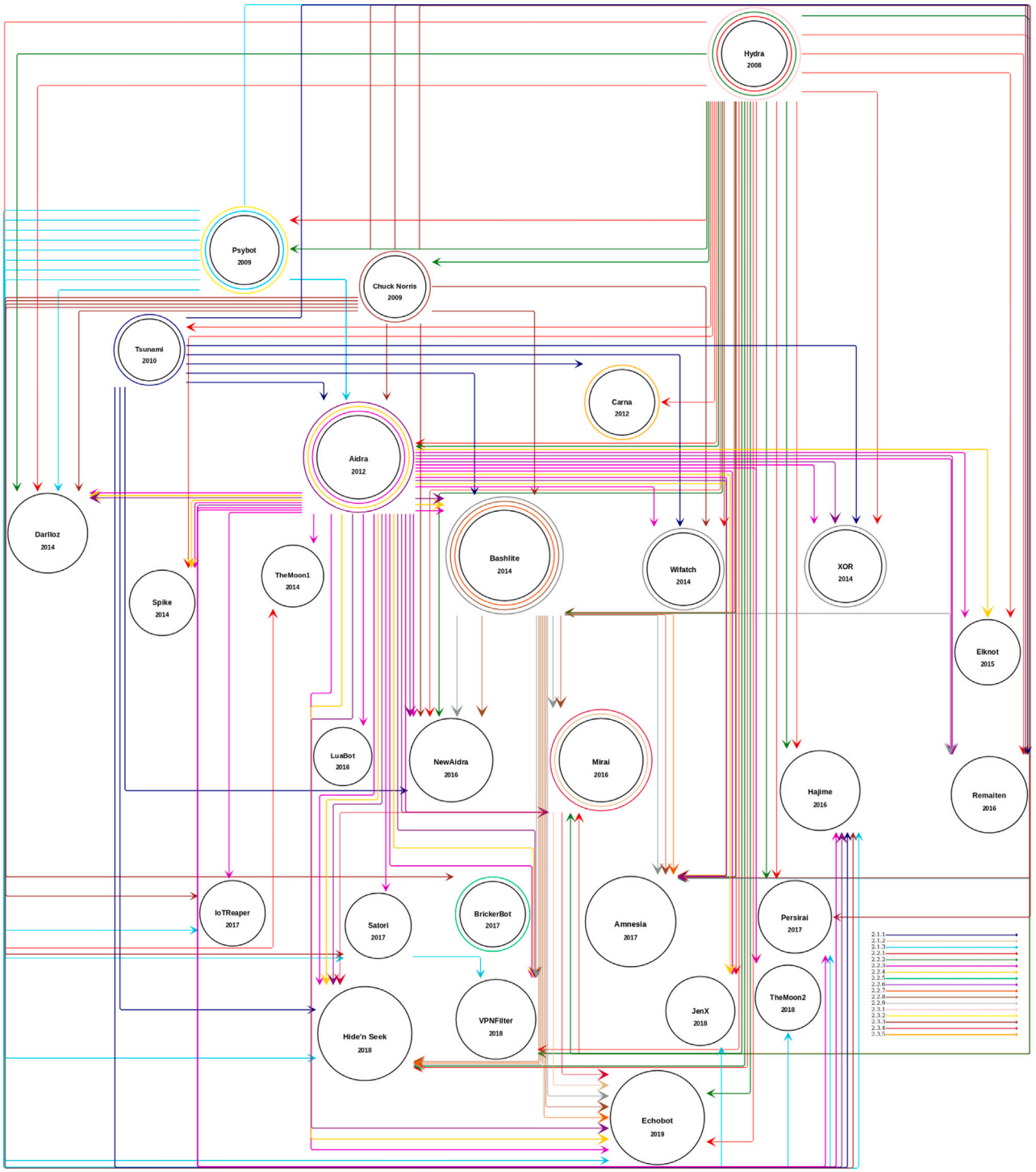


Fig. 23. Infection feature propagation multigraph.

Table 9
Experiment 1: competition between replicas of the same botnet.

Botnet	Scan method	IP gen. time	Test time	Exploit time	Remov.	Immunity	Start (t)
#1	Rand.	3	5	4	–	#1–#5	0
#2	Rand.	3	5	4	–	#1–#5	100
#3	Rand.	3	5	4	–	#1–#5	150
#4	Rand.	3	5	4	–	#1–#5	300
#5	Rand.	3	5	4	–	#1–#5	1000

Table 10
Experiment 2 : Competition between random and sequential scan.

Botnet	Scan method	IP gen. time	Test time	Exploit time	Remov.	Immunity	Start (t)
#1	Seq.	1	5	4	–	#1 #2	0
#2	Rand.	3	5	4	–	#1 #2	0

second botnet is started with a delay of 200 turns. The experiment ran for 1500 turns. The main parameters are given in Table 11.

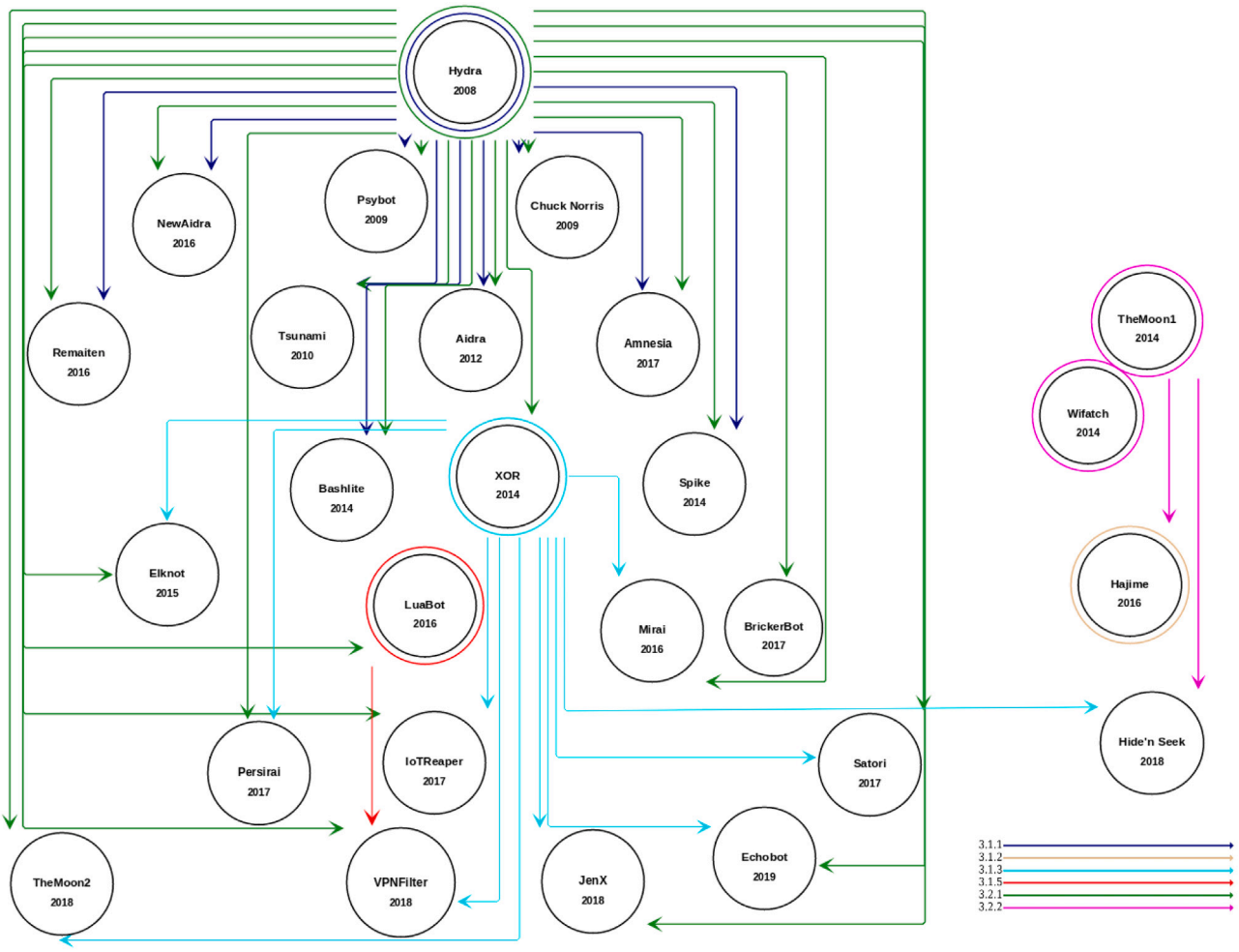


Fig. 24. Organization feature propagation multigraph.

Table 11
Experiment 3: Competition with removing.

Botnet	Scan method	IP gen. time	Test time	Exploit time	Remov. time	Immunity	Start (t)
#1	Rand.	3	5	4	–	–	0
#2	Rand.	3	5	4	#1	#1 #2	200

9.2. Results

Experiment 1

Fig. 16a shows the size of the first botnet over the time of our experiment. This botnet starts its scan at time t=0. We can observe a normal behavior, in which the botnet is able to infect most of the vulnerable population (around 30 000 bots) in the best case. The variation between min and max is very sizable, but in general the botnet is able to infect a large proportion of the potential victims. Fig. 16b shows the size of the last botnet over time. This botnet start its scan at t=1000. We can observe that it was too late, and the botnet was not able to recruit any bots. At this point, the first botnet has already reached its equilibrium point, with a maximum number of infections. Figures that capture the behavior of the other botnets of the experiment are available on our GitHub repository.[11] Fig. 17a presents the mean population of the five botnets over time. We can observe that the delay has a large influence on the final size of the botnets. The difference between botnet 0 (in blue) and botnet 1 (in orange) is only 100 turns, and yet botnet 0 infects about 20,000 more units than botnet 1. This results from the exponential ability of botnets to recruit new victims using a random scan.

Experiment 2

Fig. 18a shows the size of the first botnet over time. This botnet uses a sequential scan. We can see a poor efficiency, with a maximum size of 25 bots and a minimum around 10 bots. The equilibrium state is reached after about 600 turns. Fig. 18b shows the size of the second botnet over the time of our experiment. This botnet uses a random scan, far more efficient than the sequential scan. In most cases, the botnet is able to infect around 30,000 victims, i.e. the totality of vulnerable devices. The main difference with the previous scenario is the time required to achieve this goal. The orange curve shows an equilibrium point after 600 turns and the green one, representing the worst case, show an equilibrium point after 800 turns. From these two figures, we can deduce that the second botnet has greater influence on the equilibrium point, since it is able to infect most devices faster than the first botnet. Finally, Fig. 19a shows a close up of the evolution of the size of the population of the two botnets, for the first 800 turns. We can observe the substantial difference of efficiency between the two strategies. While the botnet utilizing sequential scan grows at a rate proportional to the elapsed time, the one that uses random scan grows at an exponential rate. This phenomenon is easily explained: with a random scan, the mean size of the population follows a binomial cumulative distribution, since each infected bot can infect more targets.

[11] https://github.com/bvignau/The_Botnet_Game.

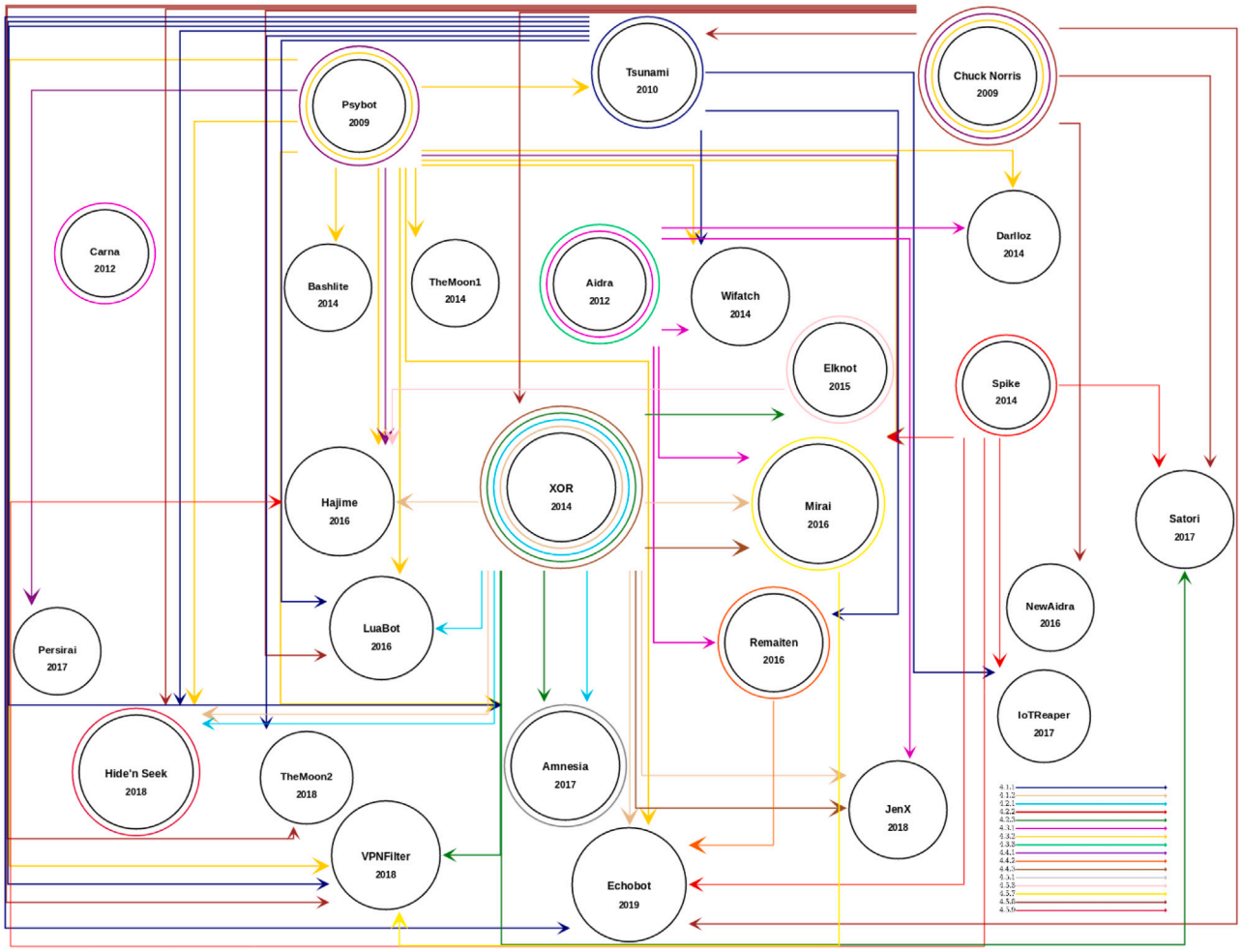


Fig. 25. Efficiency feature propagation multigraph.

Table 12
Botnet sources.

Botnet	Year	Academic	Web
Hydra (1)	2008	[10,17–20]	[21,22]
Psybot (2)	2009	[6,10,17–20,23,24]	[21]
Chuck Norris (3)	2009	[6,10,17–20]	[21]
Tsunami (4)	2010	[10,17,18,25]	[21]
Aidra (5)	2012	[10,17,18,26]	[27]
Carna (6)	2012	[17,18,29]	[28]
Bashlite (7)	2014	[10,17–19,26,30–35]	[36,37]
Darlloz (8)	2014	[17,18,26,31,32,38]	∅
Spike (9)	2014	[10,18,39]	[39]
TheMoon (10)	2014	[18]	[40]
Wifatch (11)	2014	[17–19,31,38,41]	[42]
XOR DDOS (12)	2014	[10]	[43–47]
Elknot (13)	2015	[10]	[48]
Remaiten (14)	2016	[10,17–19,31,38]	[49]
Hajime (15)	2016	[10,18,19,30,31,38,41,50–53]	[54]
Mirai (16)	2016	[7,10,17–19,25,30,31,33,38] [26,32,34,41,51,65–69]	[55–64]
New Aidra (17)	2016	[10,18]	[70]
LuaBot (18)	2016	[10,30,51,71]	[72]
Amnesia (19)	2017	[18,25,31,38]	[73,74]
BrickeBot (20)	2017	[18,30,31,38,41]	[75–78]
IoT Reaper (21)	2017	[38,41,79]	[80–84]
Persirai (22)	2017	[51]	[85,86]
Satori (23)	2017	[38,41]	[87–89]
JenX (24)	2018	∅	[90,91]
TheMoon 2 (25)	2018	∅	[92–94]
VPN Filter (26)	2018	[41,97]	[96,98–102]
Hide'n seek (27)	2018	∅	[103–108]
EchoBot (28)	2019	∅	[109–115]

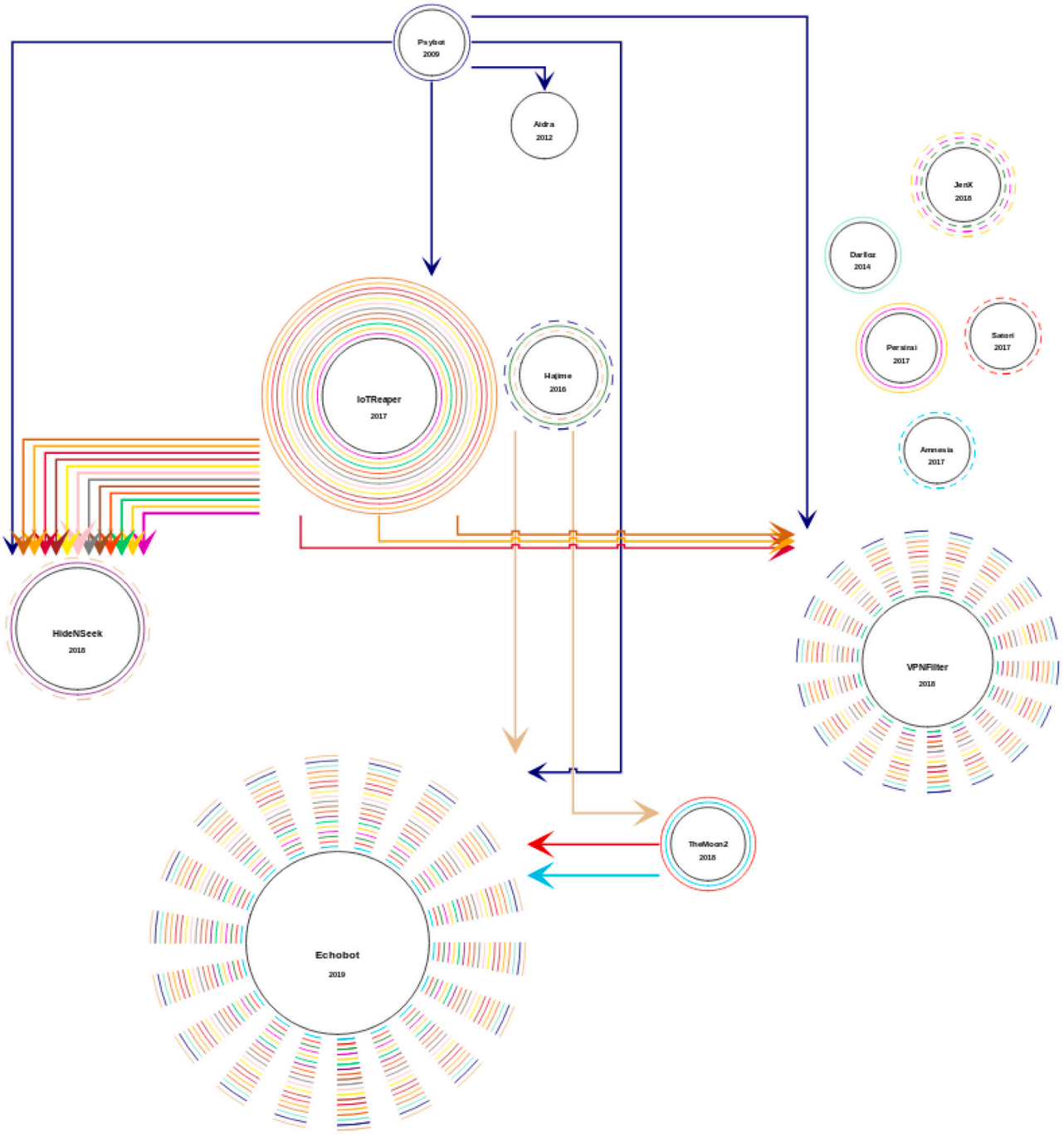


Fig. 26. Vulnerability feature propagation multigraph.

With a sequential scan, only the first bot is able to recruit new bot, and the growth of the botnet is limited by its ability to scan and infect targets quickly.

This first experiment illustrates vividly why the sequential scan was quickly replaced by a random scan in our sample set of botnets. These two strategies are easy to implement so the difficulty of implementation did not impact the adoption of the feature. Moreover, the scanning strategy is one of the core feature for a botnet, so it is normal to see an early evolution from a trivial but inefficient strategy, to a far more efficient feature.

Experiment 3

Fig. 20a shows the size of the population of the botnet #1 over the time. This botnet use a random scan and does not provide any immunity against the second botnet. We can observe a normal progression until turn 600. At this time the botnet begins to lose bots faster than new targets are recruited, until the botnet eventually disappeared.

Fig. 20b show the evolution of the second botnet, with the same scanning strategy, but capable of deleting the first botnet from infected devices. This botnet starts its scan at turn 200, giving his adversary a substantial head start. But unlike experiment 1, the second botnet is not hindered by this delay, and is eventually able to infect every vulnerable device.

Fig. 21a shows the evolution of the population of the two botnets. From this figure we can observe that the maximal size of botnet #1 is reached just before the second botnet acquires sufficient bots to enter in its exponential growth phase. Then we can also clearly see how the second botnet steals bots of the first botnet. This kind of behavior is

Table 13
Botnet features part 1.

Botnet	Features	Total
Hydra (1)	1.1.1; 1.1.2; 2.1.1; 2.1.3; 2.2.1; 2.2.2; 2.3.1; 3.1.1; 3.2.1;	8
Psybot (2)	1.1.1; 1.1.2; 1.1.3; 2.1.1; 2.1.3;2.2.1; 2.2.2; 2.3.2; 3.1.1; 3.2.1; 4.3.2; 4.4.1	12
Chuck Norris (3)	1.1.1; 1.1.2; 1.1.4; 1.3.2.3; 2.1.1; 1.5.2; 2.2.2; 2.3.3; 3.1.1; 3.2.1; 4.3.2; 4.4.1; 4.5.8;	11
Tsunami (4)	1.1.1; 1.1.2; 1.1.4, 1.1.5; 1.1.6; 2.1.1; 3.1.1; 3.2.1; 4.1.1; 4.3.2; 4.5.8;	12
Aidra (5)	1.1.1; 1.1.4; 2.1.1; 2.1.3; 2.2.1; 2.2.2; 2.2.3; 2.2.4; 2.2.6; 2.3.3;3.1.1; 3.2.1; 4.3.1; 4.3.3;	14
Carna (6)	2.1.1; 2.2.1; 2.3.5; 4.3.1;	4
Bashlite (7)	1.1.1; 1.1.2; 2.1.1; 2.2.1; 2.2.2; 2.2.3; 2.2.4; 2.2.6; 2.2.7; 2.2.8; 2.2.9; 2.3.3; 3.1.1; 3.2.1; 4.3.2;	15
Darlloz (8)	1.4.2; 1.4.3; 2.1.1; 2.1.3; 2.3.3; 2.2.1; 2.2.2; 2.2.3; 2.2.4; 2.2.6; 4.3.1; 4.3.2;	12
Spike (9)	1.1.1; 1.1.2; 1.1.3; 1.1.5; 1.1.7; 2.2.1; 2.2.3; 2.2.4; 3.1.1; 3.2.1; 4.2.5;	11
TheMoon (10)	2.1.3; 2.2.1; 3.2.2; 4.3.2;	4
Wifatch (11)	2.1.1; 2.2.1; 2.2.3; 2.2.9; 2.3.3; 3.1.3; 3.2.2; 4.1.1; 4.3.1; 4.3.2;	10
XOR DDOS (12)	1.1.1; 1.1.4; 1.1.7; 1.1.8; 2.1.1; 2.2.3; 2.26; 2.2.9; 3.1.3; 3.2.1; 4.1.2; 4.2.1; 4.2.3; 4.4.3; 4.5.8;	16
Elknot (13)	1.1.1; 1.1.2; 1.1.3; 1.1.6; 1.1.7; 1.1.8; 2.2.1; 2.2.3; 2.2.4; 3.1.3; 3.2.1; 4.2.3; 4.5.3;	12
Remaiten (14)	1.1.1; 1.1.2; 1.1.4; 1.1.5; 1.5.2; 2.1.1; 2.2.1; 2.2.3; 2.2.6; 2.2.9; 2.3.3; 3.1.1; 3.2.1; 4.1.1; 4.3.1; 4.4.2;	16
Hajime (15)	2.1.1; 2.1.3; 2.2.1; 2.2.2; 2.2.3; 2.2.4; 2.3.3; 3.1.2; 3.2.2; 4.1.1; 4.1.2; 4.3.2; 4.4.1; 4.2.2; 4.5.3;	15
Mirai (16)	1.1.1; 1.1.2; 1.1.4; 1.1.5; 1.1.6; 1.1.7; 1.1.8; 1.1.9; 1.1.10; 1.1.11; 2.1.2; 2.2.1; 2.2.2; 2.2.3; 2.2.6; 2.2.8; 2.2.9; 2.3.4; 3.1.3; 3.2.1; 4.1.2; 4.2.2; 4.3.1; 4.3.2; 4.4.1; 4.4.3; 4.5.7;	27
New Aidra (17)	1.1.1; 1.1.2; 1.1.4; 1.1.6; 2.1.1; 2.2.1; 2.2.2; 2.2.3; 2.2.6; 2.2.8 2.2.9; 2.3.3; 3.1.1; 3.2.1; 4.5.8;	15

Table 14
Botnet features part 2.

Botnet	Features	Total
LuaBot (18)	1.1.5; 1.3.2.1; 2.2.3; 3.1.5; 3.2.1; 4.1.1; 4.2.1; 4.3.2; 4.5.8;	9
Amnesia (19)	1.1.2; 1.1.5; 2.1.3; 2.2.1; 2.2.2; 2.2.3; 2.2.4; 2.2.6; 2.2.7; 2.2.8; 2.2.9; 2.3.3; 3.1.1; 3.2.1; 4.1.1; 4.2.1; 4.2.3; 4.3.2; 4.5.1;	19
BrickeBot (20)	1.2.1; 1.2.2; 2.1.1; 2.2.5; 2.3.3; 3.2.1;	6
IoT Reaper (21)	1.1.1; 1.1.2; 1.1.4; 1.1.5; 1.1.7; 1.1.8; 2.1.3; 2.2.3; 2.3.3; 3.2.1; 3.1.3; 4.1.1; 4.2.2;	13
Persirai (22)	1.1.2; 2.1.3; 2.2.1; 2.2.2; 2.2.3; 2.3.3; 3.1.3; 3.2.1; 4.4.1;	9
Satori (23)	1.1.1; 1.1.2; 1.1.4; 2.1.2; 2.1.3; 2.2.3; 2.3.3; 3.1.3; 3.2.1; 4.2.2; 4.2.3; 4.5.8;	11
JenX (24)	1.1.6; 1.1.9; 1.1.11; 1.1.12; 2.1.3; 2.2.1; 2.2.3; 2.2.4; 2.3.2; 3.2.1; 4.1.2; 4.3.1; 4.4.3; 3.1.3;	12
TheMoon 2 (25)	1.4.5; 1.4.6; 1.5.2; 2.1.3; 2.2.1; 2.2.3; 3.1.3; 3.2.1; 4.1.1; 4.5.8;	10
VPN Filter (26)	1.2.1; 1.2.2; 1.3.1.1; 1.3.1.2; 1.3.1.3; 1.3.2.1; 1.3.2.2; 1.3.2.3; 1.3.2.5; 2.1.3; 2.2.1; 2.2.2; 2.2.3; 2.2.4; 2.2.6; 2.2.9 3.1.3; 3.1.5; 3.2.1; 4.1.1; 4.2.3; 4.2.6; 4.3.2; 4.5.7; 4.5.8;	25
Hide'n Seek (27)	1.3.2.1; 1.3.2.2; 1.3.2.5; 1.4.6; 1.5.2; 2.1.1; 2.1.3; 2.2.1; 2.2.2; 2.2.3; 2.2.4; 2.2.6; 2.2.7; 2.2.8; 2.2.9; 2.3.4; 3.1.3; 3.2.2; 4.1.1; 4.1.2; 4.2.1; 4.3.2; 4.5.8; 4.5.9;	24
EchoBot (28)	1.1.1; 1.1.2; 1.1.4; 1.1.5; 1.1.6; 1.1.7; 1.1.8; 1.1.9; 1.1.10; 2.1.2; 2.1.3; 2.2.1; 2.2.2; 2.2.3; 2.2.4; 2.2.6; 2.2.7; 2.2.8; 2.2.9; 2.3.4; 3.1.3; 3.2.1; 4.1.1; 4.1.2; 4.2.2; 4.3.2; 4.4.2; 4.5.8; 1.1.11;	29

analogous to a pattern that has been observed during pandemics, in which the growth of the disease slows as patients are cured and become immune. This pattern is visible in Fig. 21a, with the yellow curve being analogous to infected individuals and the blue one representing cured individuals.

These results show that our tool can be usefully to explain why some features are preserved and spread across multiple botnets while others die out. Moreover, using this tool, we can begin to reason about new or hypothetical features, or novel combination of features and predict their efficiency. For example, we could experiment with a malware that uses the Z-map algorithm [132] to scan the network for potential targets. Finally, our third experiment shows the possibility of creating an effective patching system based on botnets' behaviors. We could create a distributed patching system where each patched devices also searches for other vulnerable devices and patches them. Solutions based on patching botnet are little studied but some initial research indicates it is highly promising [26,42,133–135].

10. Limitations and future work

In this section we expose the limitations and threats to the validity of our study and discuss potential avenues for future research.

10.1. Limitations and threats to validity

The first problem we encountered was the difficulty of finding information about certain malware and their relative botnets. While some botnets, such as Mirai have been heavily studied by the academics community, others notably TheMoon or Spike are the topic of few studies. Moreover botnets can have multiples names, which complicates the analysis process. A threat to validity arises if a malware family was not included because none of the sources we found contained information about them.

A limitation to the effective use of Feature propagation graphs as aids to understand the spread of malware arises from their complexity, both with respect to the number of features and the number of bots they contain. The addition of other statistics and Gantt diagram aids in this respect. Our FPM provides a general view and contains all relevant data

about malware evolution, but other data is needed to highlight certain specific phenomena.

10.2. Future work

Since new botnets appear every year, we will update our taxonomy with any novel features introduced by new botnets. Moreover, it would be interesting to extend our taxonomy to all kinds of botnet. Already several features are common between IoT botnet and "traditional botnets" (that target computers and servers). Features such as virtualization detection and evasion have been used by IoT botnets since 2017, while these features have been common in traditional botnets for several years. Finally, new botnets such as Echobot can exploit IoT as well as Linux and Windows servers.

Our modeling tool offers an effective way to describe and compare the impact of different behavior on the size of the botnet and the infection speed. However, it could be interesting to study the time needed to perform each action in a real botnet (such as Mirai) and create a simulation with empirically-based values. This would allow us to better model the real-life behavior of bots in the wild. We believe that our tool also can be useful to researchers seeking to create a distributed patching system, using botnets' most effective features. This system could help slow the spread of future malware.

11. Conclusion

This survey describes the evolution of IoT botnets. To this end, we defined a Main Research Question and divided it into smaller Research Questions to which we could provide definite answers. To answer these questions, we conducted a systematic literature review and we gathered as much data as possible concerning IoT botnets. We identified 28 IoT botnets families that appeared between 2008 and 2019. Using this methodology, we were able to construct a relevant taxonomy based on the feature of each botnets. We defined 77 taxa and used them to create Feature Propagation Multi-graphs that show the evolution of IoT botnets. Using these graphs, we were able to answer of our research questions.

We show that IoT malware tend to become more complex and more efficient over the years. To explain why some features are reused by other botnet and other are not, we propose three main hypotheses: the consistency of the feature with the goal of the botnet, the efficiency of the feature compared to other ones used to achieve the same goal, and finally the difficulty of implementing the feature. We also highlighted the evolution of the size, speed of infection and damages caused by IoT botnets.

Furthermore, in order to verify some of our hypotheses on the evolution of IoT botnets, we developed a tool that simulates the infection of such malware. Unlike previous models, our tool is effective to describe the competition between botnets and we can easily add new behaviors and study them. We illustrate its use by showing the different outcomes of using a sequential and a random scan which help to understand why the sequential scan was so quickly replaced by a random scan.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] S. Ropert, Internet of Things : A fast-growing market with 42 billion connected objects in 2015 and the promise of +14% annual growth up to 2025, https://en.idate.org/internet-of-things-2/.

[2] B. Farahani, F. Firouzi, V. Chang, M. Badaroglu, N. Constant, K. Mankodiya, Towards fog-driven IoT ehealth: Promises and challenges of IoT in medicine and healthcare, Future Gener. Comput. Syst. 78 (2018) 659–676, http://dx.doi.org/10.1016/j.future.2017.04.036.

[3] S. Hu, H. Wang, C. She, J. Wang, Agont: Ontology for agriculture internet of things, in: D. Li, Y. Liu, Y. Chen (Eds.), Computer and Computing Technologies in Agriculture IV, Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN: 978-3-642-18333-1, 2011, pp. 131–137.

[4] M. Wazid, A.K. Das, R. Hussain, G. Succi, J.J. Rodrigues, Authentication in cloud-driven IoT-based big data environment: Survey and outlook, J. Syst. Archit. 97 (2019) 185–196.

[5] J. Valente, M.A. Wynn, A.A. Cardenas, Stealing, spying, and abusing: Consequences of attacks on internet of things devices, IEEE Secur. Privacy 17 (5) (2019) 10–21.

[6] P. Čeleda, R. Krejčí, J. Vykopal, M. Drašar, Embedded malware - an analysis of the chuck norris botnet, in: 2010 European Conference on Computer Network Defense, 2010, pp. 3–10, http://dx.doi.org/10.1109/EC2ND.2010.15.

[7] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J.A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, Y. Zhou, Understanding the mirai botnet, in: 26th USENIX Security Symposium (USENIX Security 17), USENIX Association, Vancouver, BC, ISBN: 978-1-931971-40-9, 2017, pp. 1093–1110.

[8] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, in: EASE '14, ACM, New York, NY, USA, ISBN: 978-1-4503-2476-2, 2014, pp. 38:1–38:10, http://dx.doi.org/10.1145/2601248.2601268, URL http://doi.acm.org/10.1145/2601248.2601268.

[9] B. Vignau, R. Khoury, S. Hallé, 10 years of IoT malware: A feature-based taxonomy, in: 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), 2019, pp. 458–465, http://dx.doi.org/10.1109/QRS-C.2019.00088.

[10] M. De Donno, N. Dragoni, A. Giaretta, A. Spognardi, Ddos-capable IoT malwares: Comparative analysis and mirai investigation, Secur. Commun. Netw. 2018 (2018) 1–30, http://dx.doi.org/10.1155/2018/7178164.

[11] N. Hachem, Y. Ben Mustapha, G.G. Granadillo, H. Debar, Botnets: Lifecycle and taxonomy, in: 2011 Conference on Network and Information Systems Security, 2011, pp. 1–8, http://dx.doi.org/10.1109/SAR-SSI.2011.5931395.

[12] D. Dagon, G. Gu, C.P. Lee, W. Lee, A taxonomy of botnet structures, in: Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007), 2007, pp. 325–339, http://dx.doi.org/10.1109/ACSAC.2007.44.

[13] R.S. Rawat, E.S. Pilli, R.C. Joshi, Survey of peer-to-peer botnets and detection frameworks., IJ Netw. Secur. 20 (3) (2018) 547–557.

[14] N. Hoque, D.K. Bhattacharyya, J.K. Kalita, Botnet in ddos attacks: Trends and challenges, IEEE Commun. Surv. Tutor. 17 (4) (2015) 2242–2270.

[15] J. Sirmer, A. Streda, Botception: hire a botnet to spread one's own botnet, 2018, https://www.virusbulletin.com/conference/vb2018/abstracts/botception-hire-botnet-spread-ones-own-botnet.

[16] C. Heffner, D-link routers - authentication bypass (1), 2010, https://www.exploit-db.com/exploits/15666.

[17] K. Angrishi, Turning internet of things(IoT) into internet of vulnerabilities (iov) : Iot botnets, 2017.

[18] B. Botticelli, IoT honeypots: State of the art, 2017, https://fr.slideshare.net/BiagioBotticelli/state-of-the-art-iot-honeypots.

[19] A.C. Zaddach, Jonas, IoT Malware Comprehensive Survey, Analysis Framework and Case Studies, in: Black Hat 2018.

[20] P. Čeleda, R. Krejčí, V. Krmíček, Revealing Botnets Using Network Traffic Statistics.

[21] M. Janus, Heads of the hydra. Malware for network devices, 2011, https://securelist.com/heads-of-the-hydra-malware-for-network-devices/36396/.

[22] Iadmin, Hydra IRC bot, the 25 minute overview of the kit., 2018, http://insecurety.net/author/iadmin/.

[23] DroneBL, Drobebl, 2009, https://dronebl.org/blog/8.

[24] L. Ďurfina, J. Křoustek, P. Zemek, Psybot malware: A step-by-step decompilation case study, in: 2013 20th Working Conference on Reverse Engineering (WCRE), 2013, pp. 449–456, http://dx.doi.org/10.1109/WCRE.2013.6671321.

[25] C. Frank, C. Nance, S. Jarocki, W.E. Pauli, S. Madison, Protecting IoT from Mirai botnets; IoT device hardening, in: Proceedings of the Conference on Information Systems Applied Research, Austin, TX, USA, 2017, pp. 1508.

[26] J. Scott Sr, W. Summit, Rise of the machines: The dyn attack was just a practice run december 2016, 2016.

[27] F. Ding, IoT malware, 2017, https://github.com/ifding/iot-malware.

[28] C. Botnet, Internet census 2012, port scanning /0 using insecure embedded devices, 2012, http://census2012.sourceforge.net/paper.html.

[29] L. Markowsky, G. Markowsky, Scanning for vulnerable devices in the internet of things, in: 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 1, IEEE, 2015, pp. 463–467.

[30] G. Kambourakis, C. Kolias, A. Stavrou, The mirai botnet and the IoT zombie armies, in: MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM), 2017, pp. 267–272, http://dx.doi.org/10.1109/MILCOM.2017.8170867.

[31] S. Manoharan, IOT malware:An analysis of device hijacking, 2018.

[32] A. Wang, R. Liang, X. Liu, Y. Zhang, K. Chen, J. Li, An inside look at iot malware, in: International Conference on Industrial IoT Technologies and Applications, Springer, 2017, pp. 176–186.

[33] J.M. Ceron, K. Steding-Jessen, C. Hoepers, L.Z. Granville, C.B. Margi, Improving IoT botnet investigation using an adaptive network layer, Sensors 19 (3) (2019) 727.

[34] A. Marzano, D. Alexander, O. Fonseca, E. Fazzion, C. Hoepers, K. Steding-Jessen, M. Chaves, I. Cunha, D. Guedes, W. Meira Jr, The evolution of Bashlite and Mirai IoT botnets, 2018, http://dx.doi.org/10.1109/ISCC.2018.8538636.

[35] Y.M.P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, C. Rossow, IoTpot: A novel honeypot for revealing current IoT threats, J. Inform. Process. 24 (3) (2016) 522–533.

[36] N. Security, Agile QBot variant adds nbotloader netgear bug in its new update, 2017, https://blog.newskysecurity.com/agile-122bf2f4e2f3.

[37] T. Spring, BASHLITE family of malware infects 1 million IoT devices, 2016, https://threatpost.com/bashlite-family-of-malware-infects-1-million-iot-devices/120230/.

[38] A. Kumar, T.J. Lim, EDIMA: early detection of IoT malware network activity using machine learning techniques, 2019, CoRR, abs/1906.09715, arXiv:1906.09715.

[39] Akamai, Spike ddos toolkit, 2014.

[40] J.B. Ullrich, Linksys worm "themoon" summary: What we know so far, 2014, https://isc.sans.edu/forums/diary/Linksys+Worm+TheMoon+Summary+What+we+know+so+far/17633.

[41] A.O. Prokofiev, Y.S. Smirnova, Counteraction against internet of things botnets in private networks, in: 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2019, pp. 301–305, http://dx.doi.org/10.1109/EIConRus.2019.8657100.

[42] Unknown, 2016, https://gitlab.com/rav7teif/linux.wifatch/.

[43] MalwareMustDie, MMD-0028-2014 - linux/xor.ddos : Fuzzy reversing a new China ELF, 2014, https://blog.malwaremustdie.org/2014/09/mmd-0028-2014-fuzzy-reversing-new-china.html.

[44] S.J. Vaughan-Nichols, Incompetence, not linux, is behind the XOR ddos botnet, 2015, https://www.zdnet.com/article/incompetence-not-linux-is-behind-the-xor-ddos-botnet/.

[45] K. Shortt, XOR DDos mitigation and analysis, 2015, https://isc.sans.edu/forums/diary/XOR+DDOS+Mitigation+and+Analysis/19827/.

[46] Akamai, XOR DDos botnet launching 20 attacks a day from compromised linux machines, says akamai, 2015, https://www.akamai.com/us/en/about/news/press/2015-press/xor-ddos-botnet-attacking-linux-machines.jsp.

[47] Akamai, 1 case study: Fastdns infrastructure battles xor botnet, 2015, https://www.akamai.com/de/de/multimedia/documents/state-of-the-internet/fast-dns-xor-botnet-case-study.pdf.

[48] MalwareMustDie, MMD-0026-2014 - linux/aes.ddos: Router malware warning | reversing an ARM arch ELF, 2014, http://blog2.malwaremustdie.org/2014/09/reversing-arm-architecture-elf-elknot.html.

[49] M. Malik, M.-E. M.Léveillé, Meet Remaiten – a Linux bot on steroids targeting routers and potentially other IoT devices, 2016, https://www.welivesecurity.com/2016/03/30/meet-remaiten-a-linux-bot-on-steroids-targeting-routers-and-potentially-other-iot-devices/.

[50] E. Sam, P. Ioannis, Hajime: Analysis of a decentralized internet worm for IoT devices, 2016, https://www.cs.umd.edu/class/fall2017/cmsc818O/papers/hajime-rapidity.pdf.

[51] C. Kolias, G. Kambourakis, A. Stavrou, J. Voas, DDoS in the IoT: Mirai and other botnets, Computer 50 (7) (2017) 80–84, http://dx.doi.org/10.1109/MC.2017.201.

[52] S. Herwig, K. Harvey, G. Hughey, R. Roberts, D. Levin, Measurement and analysis of hajime, a peer-to-peer IoT botnet., in: NDSS, 2019.

[53] R. Emergency Response Team, Hajime - friend or foe ? 2017, https://security.radware.com/ddos-threats-attacks/hajime-iot-botnet/.

[54] V.D.W. Jornt, D. Vicente, N. Yury, K. Zykov, Hajime, the mysterious evolving botnet, 2017, https://securelist.com/hajime-the-mysterious-evolving-botnet/78160/.

[55] BadCyber, New mirai attack vector – bot exploits a recently discovered router vulnerability, 2016, https://badcyber.com/new-mirai-attack-vector-bot-exploits-a-recently-discovered-router-vulnerability/.

[56] B. Krebs, New mirai worm knocks 900k germans offline, 2016, https://krebsonsecurity.com/2016/11/new-mirai-worm-knocks-900k-germans-offline/.

[57] D. Ben, Breaking down mirai: An IoT ddos botnet analysis, 2016, https://www.imperva.com/blog/malware-analysis-mirai-ddos-botnet/.

[58] Cloudflare, Inside the infamous mirai IoT botnet: A retrospective analysis, 2017, https://blog.cloudflare.com/inside-mirai-the-infamous-iot-botnet-a-retrospective-analysis/.

[59] J. Leyden, Researchers expose mirai vuln that could be used to hack back against botnet, 2016, https://www.theregister.co.uk/2016/10/28/mirai_botnet_hack_back/.

[60] D. Pauli, Boffin's anti-worm bot could silence epic Mirai DDoS attack army, 2016, https://www.theregister.co.uk/2016/10/31/this_antiworm_patch_bot_could_silence_epic_mirai_ddos_attack_army/.

[61] I. Arghire, New mirai variants have built-in domain generation algorithm, 2016, https://www.securityweek.com/new-mirai-variants-have-built-domain-generation-algorithm.

[62] us-cert, Alert (TA16-288a) – heightened ddos threat posed by mirai and other botnets, 2016, https://www.us-cert.gov/ncas/alerts/TA16-288A.

[63] MalwareTech, Mapping mirai: A botnet case study, 2016, https://www.malwaretech.com/2016/10/mapping-mirai-a-botnet-case-study.html.

[64] Flashpoint, Flashpoint monitoring of mirai shows attempted DDoS of trump and clinton websites, 2016, https://www.flashpoint-intel.com/blog/trending/attempted-ddos-trump-and-clinton-websites/.

[65] R. Hallman, J. Bryan, G. Palavicini Jr., J. Divita, J. Romero-Mariona, Ioddos — The internet of distributed denial of service attacks: A case study of the mirai malware and IoT-based botnets, 2017, http://dx.doi.org/10.5220/0006246600470058.

[66] J.A. Jerkins, Motivating a market or regulatory solution to IoT insecurity with the mirai botnet code, in: 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), 2017, pp. 1–5.

[67] Y. Ji, L. Yao, S. Liu, H. Yao, Q. Ye, R. Wang, The study on the botnet and its prevention policies in the internet of things, in: 2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD)), 2018, pp. 837–842.

[68] J. Margolis, T.T. Oh, S. Jadhav, Y.H. Kim, J.N. Kim, An in-depth analysis of the mirai botnet, in: 2017 International Conference on Software Security and Assurance (ICSSA), 2017, pp. 6–12.

[69] H. Sinanović, S. Mrdovic, Analysis of mirai malicious software, in: 2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), 2017, pp. 1–5.

[70] MalwareMustDie, MMD-0059-2016 - Linux/IRCTelnet (new Aidra) - A DDoS botnet aims IoT w/ IPv6 ready, 2016, https://blog.malwaremustdie.org/2016/10/mmd-0059-2016-linuxirctelnet-new-ddos.html.

[71] A. Costin, Lua code: Security overview and practical approaches to static analysis, in: 2017 IEEE Security and Privacy Workshops (SPW), 2017, pp. 132–142, http://dx.doi.org/10.1109/SPW.2017.38.

[72] MalwareMustDie, MMD-0057-2016 - linux/luabot - IoT botnet as service, 2017, https://blog.malwaremustdie.org/2016/09/mmd-0057-2016-new-elf-botnet-linuxluabot.html.

[73] C. Xiao, U.. Cong Zheng, New IoT/linux malware targets DVRs, forms botnet, 2017, https://unit42.paloaltonetworks.com/unit42-new-iotlinux-malware-targets-dvrs-forms-botnet/.

[74] J. Leyden, 'amnesia' IoT botnet feasts on year-old unpatched vulnerability, 2017, https://www.theregister.co.uk/2017/04/07/amnesia_iot_botnet/.

[75] Radaware, "BrickerBot" results in PDoS attack, 2017, https://security.radware.com/ddos-threats-attacks/brickerbot-pdos-permanent-denial-of-service/.

[76] C. Cimpanu, BrickerBot dev claims cyber-attack that affected over 60,000 Indian modems, 2017, https://www.bleepingcomputer.com/news/security/brickerbot-dev-claims-cyber-attack-that-affected-over-60-000-indian-modems/.

[77] Radaware, "brickerBot" pdos back with vengeance, 2017, https://security.radware.com/ddos-threats-attacks/brickerbot-pdos-back-with-vengeance/.

[78] Radaware, "brickerBot" the dark knight of IoT, 2017, https://blog.radware.com/security/2017/04/brickerbot-dark-knight-iot/.

[79] P. Moriuchi, S. Chohan, Mirai-variant IoT botnet used to target financial sector in january 2018, 2018, https://go.recordedfuture.com/hubfs/reports/cta-2018-0405.pdf.

[80] Trend Mirco System, New rapidly-growing IoT botnet - REAPER, 2018, https://success.trendmicro.com/solution/1118928-new-rapidly-growing-iot-botnet-reaper#collapseTwo.

[81] S.E. FortiGuard, Reaper: The Next Evolution of IoT Botnets, https://www.fortinet.com/blog/threat-research/reaper-the-next-evolution-of-iot-botnets.html.

[82] yegenshen, IoT_reaper: A rappid spreading new IoT botnet, 2017.

[83] K. Brian, Fear the reaper, or reaper madness ? 2017, https://krebsonsecurity.com/2017/10/fear-the-reaper-or-reaper-madness/#more-41321.

[84] C. Point, IoTRoop botnet: The full investigation, 2017, https://research.checkpoint.com/iotroop-botnet-full-investigation/.

[85] New Jersey Cybersecurity and Communications Integration Cell, Persirai, 2017, https://www.cyber.nj.gov/threat-profiles/botnet-variants/persirai.

[86] TrendMicro, Persirai: New internet of things (IoT) botnet targets IP cameras, 2017, https://blog.trendmicro.com/trendlabs-security-intelligence/persirai-new-internet-things-iot-botnet-targets-ip-cameras/.

[87] NewSkySecurity, Masuta : Satori creators' second botnet weaponizes a new router exploit., 2018, https://blog.newskysecurity.com/masuta-satori-creators-second-botnet-weaponizes-a-new-router-exploit-2ddc51cc52a7.

[88] Netlab360, Warning: Satori, a mirai branch is spreading in worm style on port 37215 and 52869, 2018, https://blog.netlab.360.com/warning-satori-a-new-mirai-variant-is-spreading-in-worm-style-on-port-37215-and-52869-en/.

[89] Netlab360, Botnets never die, satori REFUSES to fade away, 2018, https://blog.netlab.360.com/botnets-never-die-satori-refuses-to-fade-away-en/.

[90] Radware, Jenx botnet: A new IoT botnet threatening all, 2018, https://security.radware.com/ddos-threats-attacks/threat-advisories-attack-reports/jenx/.

[91] A. Davila, Jenx botnet: A new IoT botnet threatening all, 2019, https://unit42.paloaltonetworks.com/home-small-office-wireless-routers-exploited-to-attack-gaming-servers/.

[92] B. Lab, A new phase of themoon, 2019, https://blog.centurylink.com/a-new-phase-of-themoon/.

[93] S. Tara, Themoon rises again, with a botnet-as-a-service threat, 2019, https://threatpost.com/themoon-botnet-as-a-service/141393/.

[94] Netlab 360, GPON exploit in the wild (IV) - themoon botnet join in with a 0day(?), 2018, https://blog.netlab.360.com/gpon-exploit-in-the-wild-iv-themoon-botnet-join-in-with-a-0day/.

[95] Kaspersky, Blackenergy APT attacks in Ukraine, 2020, https://www.kaspersky.com/resource-center/threats/blackenergy.

[96] T.U. William Largent, New VPNFilter malware targets at least 500K networking devices worldwide, 2018, https://blog.talosintelligence.com/2018/05/VPNFilter.html.

[97] S. Sicato, J. Costa, P.K. Sharma, V. Loia, J.H. Park, VPNFilter malware analysis on cyber threat in smart home Network, 9, (13) 2019, p. 2763,

[98] T.U. William Largent, VPNFilter update - VPNFilter exploits endpoints, targets new devices, 2018, https://blog.talosintelligence.com/2018/06/vpnfilter-update.html.

[99] T.U. Edmund Brumaghin, VPNFilter III: More tools for the Swiss army knife of malware, 2018, https://blog.talosintelligence.com/2018/09/vpnfilter-part-3.html.

[100] C. Cimpanu, FBI Takes Control of APT28's VPNFilter Botnet, https://www.bleepingcomputer.com/news/security/fbi-takes-control-of-apt28s-vpnfilter-botnet/.

[101] C. Cimpanu, Ukraine says it stopped a vpnfilter attack on a chlorine distillation station, 2018, https://www.bleepingcomputer.com/news/security/ukraine-says-it-stopped-a-vpnfilter-attack-on-a-chlorine-distillation-station/.

[102] VPNFilter-affected Devices Still Riddled with 19 Vulnerabilities.

[103] B. Botezatu, New hide 'n seek IoT botnet using custom-built peer-to-peer communication spotted in the wild, 2018, https://labs.bitdefender.com/2018/01/new-hide-n-seek-iot-botnet-using-custom-built-peer-to-peer-communication-spotted-in-the-wild/.

[104] TrendMicro, "Hide 'n seek" botnet uses peer-to-peer infrastructure to compromise IoT devices, 2018, https://www.trendmicro.com/vinfo/es/security/news/internet-of-things/-hide-n-seek-botnet-uses-peer-to-peer-infrastructure-to-compromise-iot-devices.

[105] L. Arsene, Hide and seek IoT botnet learns new tricks: Uses ADB over internet to exploit thousands of android devices, 2018, https://labs.bitdefender.com/2018/09/hide-and-seek-iot-botnet-learns-new-tricks-uses-adb-over-internet-to-exploit-thousands-of-android-devices/.

[106] A. Şendroiu, V. Diaconescu, Hide'n'seek: an adaptive peer-to-peer IoT botnet, 2018.

[107] Avast, Let's play hide 'n seek with a botnet, 2018, https://blog.avast.com/hide-n-seek-botnet-continues.

[108] MalwareTech, Tracking the hide and seek botnet, 2019, https://www.malwaretech.com/2019/01/tracking-the-hide-and-seek-botnet.html.

[109] I. Ilascu, New echobot botnet variant uses over 50 exploits to propagate, 2019, https://www.bleepingcomputer.com/news/security/new-echobot-botnet-variant-uses-over-50-exploits-to-propagate/.

[110] R. Nigam, New mirai variant adds 8 new exploits, targets additional IoT devices, 2019, https://unit42.paloaltonetworks.com/new-mirai-variant-adds-8-new-exploits-targets-additional-iot-devices/.

[111] L. Cashdollar, Latest ECHOBOT: 26 infection vectors, 2019, https://blogs.akamai.com/sitr/2019/06/latest-echobot-26-infection-vectors.html.

[112] I. Ilascu, Echobot botnet spreads via 26 exploits, targets oracle, vmware apps, 2019, https://www.bleepingcomputer.com/news/security/echobot-botnet-spreads-via-26-exploits-targets-oracle-vmware-apps/.

[113] R. Nigam, Mirai variant ECHOBOT resurfaces with 13 previously unexploited vulnerabilities, 2019, https://unit42.paloaltonetworks.com/mirai-variant-echobot-resurfaces-with-13-previously-unexploited-vulnerabilities/.

[114] K. Eli, Z. Maxim, P. Raymond, Echobot malware now up to 71 exploits, targeting SCADA, 2019, https://www.f5.com/labs/articles/threat-intelligence/echobot-malware-now-up-to-71-exploits--targeting-scada.

[115] C. Cimpanu, Le nouveau malware d'Echobot est un concentré de vulnérabilités, 2019, https://www.zdnet.fr/actualites/le-nouveau-malware-d-echobot-est-un-concentre-de-vulnerabilites-39886143.htm.

[116] D. Moore, C. Shannon, G. Voelker, S. Savage, et al., Network telescopes: Technical report, Tech. rep., Cooperative Association for Internet Data Analysis (CAIDA), 2004.

[117] M. Bailey, E. Cooke, F. Jahanian, J. Nazario, D. Watson, et al., The internet motion sensor-a distributed blackhole monitoring system., in: NDSS, 2005.

[118] C.C. Zou, W. Gong, D. Towsley, Code red worm propagation modeling and analysis, in: Proceedings of the 9th ACM Conference on Computer and Communications Security, ACM, 2002, pp. 138–147.

[119] W.O. Kermack, A.G. McKendrick, A contribution to the mathematical theory of epidemics, Proc. R. Soc. Lond. Ser. A 115 (772) (1927) 700–721.

[120] L.J. Allen, Some discrete-time SI, SIR, and SIS epidemic models, Math. Biosci. 124 (1) (1994) 83–105.

[121] D. Moore, C. Shannon, G.M. Voelker, S. Savage, Internet quarantine: Requirements for containing self-propagating code, in: IEEE INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428), 3, IEEE, 2003, pp. 1901–1910.

[122] S. Staniford, V. Paxson, N. Weaver, et al., How to own the internet in your spare time., in: USENIX Security Symposium, Vol. 2, 2002, pp. 14–15.

[123] Z. Chen, C. Ji, Spatial-temporal modeling of malware propagation in networks, IEEE Trans. Neural Netw. 16 (5) (2005) 1291–1303.

[124] Z. Chen, L. Gao, K. Kwiat, Modeling the spread of active worms, in: IEEE INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428), Vol. 3, IEEE, 2003, pp. 1890–1900.

[125] V.N. Kolokoltsov, A. Bensoussan, Mean-field-game model for botnet defense in cyber-security, Appl. Math. Optim. 74 (3) (2016) 669–692.

[126] B. Soper, J. Musacchio, A botnet detection game, in: 2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton), IEEE, 2014, pp. 294–303.

[127] A. Bensoussan, M. Kantarcioglu, S.C. Hoe, A game-theoretical approach for finding optimal strategies in a botnet defense model, in: International Conference on Decision and Game Theory for Security, Springer, 2010, pp. 135–148.

[128] I. Kotenko, A. Konovalov, A. Shorov, Agent-based modeling and simulation of botnets and botnet defense, in: Conference on Cyber Conflict. CCD COE Publications. Tallinn, Estonia, Citeseer, 2010, pp. 21–44.

[129] Q. Wu, S. Shiva, S. Roy, C. Ellis, V. Datla, On modeling and simulation of game theory-based defense mechanisms against DoS and DDoS attacks, in: Proceedings of the 2010 Spring Simulation Multiconference, 2010, pp. 1–8.

[130] Y. Wang, J. Ma, L. Zhang, W. Ji, D. Lu, X. Hei, Dynamic game model of botnet ddos attack and defense, Secur. Commun. Netw. 9 (16) (2016) 3127–3140.

[131] M. Ficco, F. Palmieri, Leaf: An open-source cybersecurity training platform for realistic edge-IoT scenarios, J. Syst. Archit. 97 (2019) 107–129.

[132] D. Adrian, Z. Durumeric, G. Singh, J.A. Halderman, Zippier zmap: Internet-wide scanning at 10 gbps, in: 8th USENIX Workshop on Offensive Technologies (WOOT 14), USENIX Association, San Diego, CA, 2014.

[133] J.A. Jerkins, Motivating a market or regulatory solution to IoT insecurity with the mirai botnet code, in: 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), 2017, pp. 1–5, http://dx.doi.org/10.1109/CCWC.2017.7868464.

[134] R. Anderson, Why information security is hard - an economic perspective, in: Seventeenth Annual Computer Security Applications Conference, 2001, pp. 358–365, http://dx.doi.org/10.1109/ACSAC.2001.991552.

[135] M. De Donno, N. Dragoni, A. Giaretta, M. Mazzara, AntibIoTic: Protecting IoT devices against ddos attacks, in: P. Ciancarini, S. Litvinov, A. Messina, A. Sillitti, G. Succi (Eds.), Proceedings of 5th International Conference in Software Engineering for Defence Applications, Springer International Publishing, Cham, 2018, pp. 59–72.

Benjamin Vignau, Ms. Eng, Ms. Sc obtained his first Master of Engineering, specialized in computer security, at the National Institute of Applied Sciences of Bourges (France) in 2019. He obtained a Master of Science at the Université du Québec à Chicoutimi. His researches focused on the Internet of Things security and especially on IoT Botnets.

Raphaël Khoury, Ph.D., P. Eng obtained his Ph.D. from Université Laval and was a post-doctoral fellow at the Defense Research and Development Canada,(DRDC-RDDC) in Valcartier, Canada. He is currently a professor at the Université du Québec à Chicoutimi. His research interest include all aspect of software security. He is the recipient of FQRNT and NSERC research grants. His research has also been funded by industry.

Sylvain Hallé, Ph.D. is the Canada Research Chair on Software Specification, Testing and Verification at Université du Québec à Chicoutimi, Canada, and current head of the Formal Computer Science Laboratory (LIF). He received the BS degree in mathematics from Université Laval in 2002 and the M.Sc. in mathematics and Ph.D. in computer science from Université du Québec à Montréal in 2004 and 2008, respectively. Before entering UQAC in 2010, he has been a postdoctoral research fellow at University of California, Santa Barbara. His major research interests include formal methods, runtime verification and event stream processing. He is a senior member of the ACM and the IEEE. He is the head of two major research projects at LIF, the BeepBeep event stream processing engine and the Cornipickle web layout testing tool. BeepBeep has been the subject of a textbook published in 2018, while Cornipickle has earned the Best Tool Paper Award from ICST in 2015.

Abdelwahab (Wahab) Hamou-Lhadj is a professor in the Department of Electrical and Computer Engineering at Concordia University. His research interests include software development and operations, software modeling, software tracing and logging, anomaly detection, and the application of machine learning to these areas. He has many years of experience leading research and development projects in various companies including Ericsson. Ubisoft, CAE, and Defence R&D Canada. Dr. Hamou-Lhadj received his Ph.D. from the University of Ottawa in 2005. He is a long-lasting member of IEEE and ACM, a licensed engineer with the province of Quebec, a major contributor to OMG certification programs, and a governance board member of ITACS BTM Forum.