Modern Botnets

KIREPRO1PE

Research Project

Petrut Andrei Jianu - peji@itu.dk

December 2020

Master of Science, Computer Science
ITU Copenhagen

Contents

Abstract

Botnets are a costly problem and difficult to manage and investigate. This investigation difficulty is partly owed to the fact that the literature is mostly made of short, focused articles which makes the information gathering process rather slow. This paper aims to facilitate further investigative efforts. Furthermore, for prevention and detection of botnet attacks, one should be able to understand what they are and how they work, which is why this paper is designed to be understood by someone new in the field, but also useful to the experienced researcher. In my analysis I have taken into account published papers and technical white papers or write-ups from security researchers and organizations and, where such information was available, included details about the tools and methods used in investigation. This literature review resulted in an overview table and showed that the most common vulnerabilities exploited by botnets are human errors. The general lack of security measures and knowledge can be diminished through education.

1 Introduction

Botnets represent a collection (a network) of Internet-connected devices, such as computers, smartphones or, as it will be shown in section 3, Internet of Things devices. These machines are infected with malicious software such that a bad-intended actor can remotely control them and perform actions without the knowledge or consent of their legitimate users, thus the name of bots (or zombies).

Botnets have had different architectures and scopes over time, but their main purpose was always connected to financial gains for attackers through illegitimate manners - more details about this can be found in section 1.2.

1.1 Motivation

The damage done by botnets costs the world hundreds of billions of dollars yearly in the form of advertisement fraud (est. $42 billion in 2019 [1]), power consumption and denial of service related costs. However, botnets could potentially do much more damage than just financial losses - [2] details the disruption of the Trickbot botnet by Microsoft due to concerns for the safety of key infrastructure during the presidential elections in the USA. Through Distributed Denial of Service (DDoS) attacks or through distributed phishing campaigns, malicious actors can influence elections by denying access to targeted parts of the population., creating distrust, anger and confusion among peers. The biggest concern here was not that the malware could actually alter the vote counts, but that it would create panic, anger and confusion among peers if their voting station were to be hit by ransomware and their votes couldn't be counted.

Botnets are more than DDoS attacks and in order to be able to detect them and take them down we must first understand how they work and what are their purposes, which is why in this paper I present an overview over the most notable ones from the recent years.

1.2 History

In 2001, EarthLink - an Internet service provider from USA - won a 25$ million lawsuit against Khan C. Smith who

was found guilty of spam that caused millions of dollars worth of damages. ”EarthLink estimates Smith’s scheme accounted for as much as 12 percent of the company’s total e-mail traffic between September 2000 and August 2001 and cost the company an estimated $4.1 million in bandwidth and delivery expenses, lost profits, administrative and attorney fees and other anti-spam measures” [3]. This is considered to be the first official detection of a botnet.
As mentioned in section 1, botnets have evolved over the years not only in regards to architecture and technology, but also in their purposes. Initially they were designed for phishing emails as it can be seen at the beginning of this section, but, as time went by, hackers realised they can steal credentials from computers without prompting the users to type them and, furthermore, they could use the computing power of infected devices for their own advantage. This was the case of the Gameover_ZeuS botnet, which was built on top of the ZeuS trojan [4], [5] and used the Cutweil botnet to spread [6], [7].
Modern botnets are mostly used for, but not limited at, DDoS attacks (further detailed in section 3) and ransomware propagation (see Trickbot, section 5). Since most botnets have a loader module that can install additional malware, cryptocurrency mining or credential-stealing modules usually come along.

1.3 Goals

Future investigations into zombie networks should be easier than they are now, and I intend to enable information gathering by collecting key pieces of knowledge, artifacts and analysis methods into this paper. literature review of the last years to gain a better, more thorough understanding of powerful recent botnets. The goal is to note down key differences and similarities and extract other key information into one place for further research, as well as to provide insights into the research methods that have been used so far.

1.4 Related work

Over the years, security researchers have shown a growing interest regarding botnets as they have been more and more disruptive. [8] provides a complete analysis over the Mirai botnet and [9] presents an overview over the IoT threat landscape, while [10] pictures the evolution of botnets through history. To my knowledge, there are no other papers that analyze the existing literature on botnets.

2 Analysis Metrics and Terminology

Sections 3, 4 and 5 contain detailed analysis of Mirai, FritzFrog and, respectively, the Trickbot botnets. In order to paint a clearer picture, the key metrics for analysis (see subsection 2.1) and important terms (see subsection 2.2) are defined in this section.

2.1 Key metrics

- **Scope**
  What is the main purpose of this botnet? How does the typical target look like? What is the interest of the attackers?
- **Control Mechanism** Botnets’ structure can be either centralized (see figure 1) or decentralized (also known as peer to peer or, simply, P2P - see figure 2) - both cases will be analyzed in this paper. While hybrid architectures do exist, no such botnet is reviewed in this paper. In the case of a centralized botnet, the party controlling the infected devices, often called the *botmaster* uses a server to send and receive information from them. In order to mitigate the drawback of having a single point of failure, botmasters usually set up multiple mirror servers and change their IP oftenly to increase the difficulty of any potential investigation and decrease the posibility of getting caught or disrupted. The term will be further noted as C2 or C2 server.

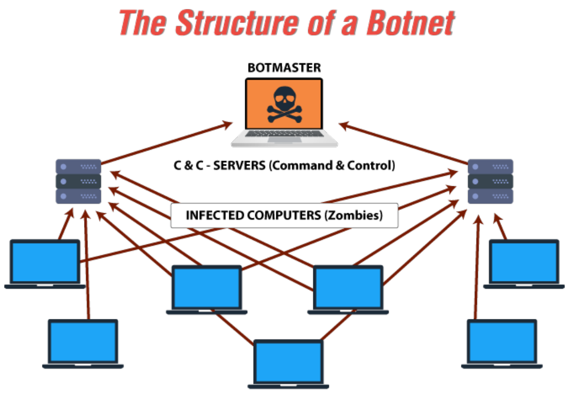

The Structure of a Botnet
BOTMASTER
C & C - SERVERS (Command & Control)
INFECTED COMPUTERS (Zombies)

Fig. 1. Centralized Botnet Structure - Source: EC-Council

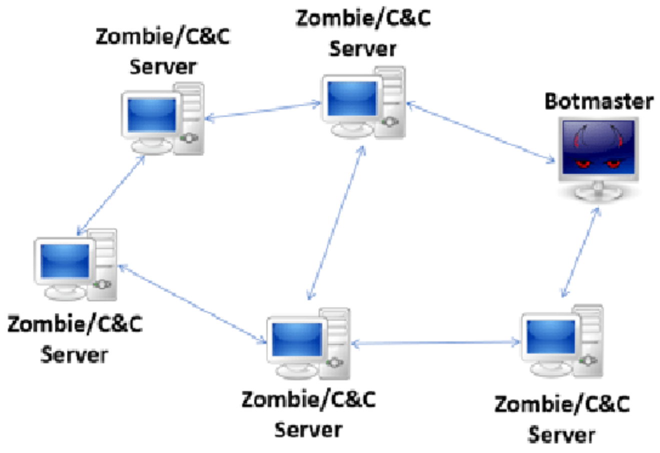


Fig. 2. Decentralized Botnet Structure - Source:[11]

In the case of decentralized botnets, each bot acts as a C2 server, propagating commands and information through the network. In this distributed model, bots receive a list of *neighbouring* devices and forwards commands from the botmaster to them while also sending back received information. This architecture is much more difficult to implement in practice compared to centralized botnets, but it is also harder to take down or disrupt.

- **Growth Mechanism**
  How is a machine infected? How does infection spread throughout the same network - if so? What are the attack vectors and the exploited vulnerabilities? How do nodes authenticate themselves to the rest of the network?
- **Update Mechanism**
  Are infected machines receiving update? If so, how does that happen?

2.2 Terminology

- **Active Scanning**
  The term of active scanning represents the act of sending test traffic across the network and querying endpoints to collect basic information such as device name, IP address as well as information about installed firmware, software and operating system, including their versions.[12]
- **Network Telescope**
  A network telescope is a system that monitors traffic on a given address space to observe large-scale events, patterns and suspicious activities such as worm scanning, DDoS activity or even an innocent configuration mistake. While some traffic may be legitimate, most of the addresses observed are supposed to be unused and therefore the assumption that traffic directed to them comes from a malicious party is made. [13]
- **Honeypots**
  Honeypots mimic the behavior of a real device or system in order to make malicious actors believe they are attacking a legitimate target. This helps defensive security efforts by allowing experts to track and analyze malware behaviour in order to protect actual assets. To make them more appealing, honeypots are usually designed with vulnerabilities that can be exploited more or less easily, but they still need to enforce a level of security measure to avoid unwanted collateral damage such as lateral intrusion into the legitimate network or illegitimate usage of the resources allocated to the honeypot. Since they are cut off from the *real* network, it is assumed that, with a low false-positive rate, all traffic flowing through the honeypot is coming from cybercriminals. Therefore, analysis on said traffic can help detect patterns, countries of origin and to assess how the implemented security measures behave in the face of a real threat on a dummy target.
- **Bulletproof Hosting**
  Everything on the Internet is *hosted* (stored) somewhere and threats are not an exception of this. Since you need to be able to constantly, uninterruptedly handle a high amount of traffic, you would usually make use of the hundreds of web-hosting services that are available throughout the world, but this is not the case for cybercriminals because providing hosting for malware or malware-associated services is illegal in most countries. Hence, to fill the gap in supply, some web hosting providers are much more indulgent in regards to the contents of their servers and operate under a *"No questions asked"* policy and they are typically located in countries with more *relaxed* legislation on this matter. The amount of content stored through these services is very high, ranging from legitimate content to botnet command and control servers, black-market websites, exploit kits and so on, which makes it even more difficult to link certain traffic or content to a specific customer.
- **Phishing**
  Phishing is the attempt to gain access to sensitive information such as login credentials or credit card data by making fraudulent claims such as trying to appear as a trustworthy entity. There are different types of phishing, but the ones that will be talked about in this paper are bulk phishing (spam emails advertising free products or messages such as "Enter your credit card details to receive $1000") and spearphishing - phishing targeted at a specific company or individual, such as carefully tailored resumes (that contain malware) sent out to a human resources employee.
- **Brute Force Attack**
  A brute force attack is an attempt to guess the correct login credentials by using a predefined dictionary of common words and passwords and submitting them. There are some tools that attempt to prevent this, such as Fail2Ban [14], but there are also workarounds known by malicious actors.
- **Packing**
  Packing executables with tools such as the Ultimate Packer for Executables is a common technique not only for malware, but also for legitimate software. A packer compresses and compartmentalizes an executable, obfuscating the binary and its purpose in the process. This part is particularly useful for malicious software authors to avoid detection by unsuspecting users, but a respectable security solution should detect and unpack it automatically.

3 The Mirai Botnet

In the fall of 2016, the world witnessed one of the largest DDoS attacks up to that point with traffic amounts of over 600 Gbps targeted at the Krebs on Security blog [15], OVH [16] and the DNS provider Dyn [17] - causing outages in multiple world regions (figure 3). These massive attacks came from hundreds of thousands of Internet of Things devices comprising the Mirai botnet. Mirai was not the first IoT botnet, but it was the most *successful* one. Although Mirai focused on DDoS attacks, the IoT landscape is still vulnerable four years later and applications are numerous, from advertisement fraud to cryptocurrency mining and more. While there has been some evolution in regards to the security of IoT devices, there is still a lot of room for improvement in this matter as the starting moments were comparable to the 2000s desktops.

At its peak, Mirai infected approximately six hundred thousand devices. Most of this devices were located in Central and South America. The Mirai botnet started spreading from a single IP address belonging to a bulletproof host.

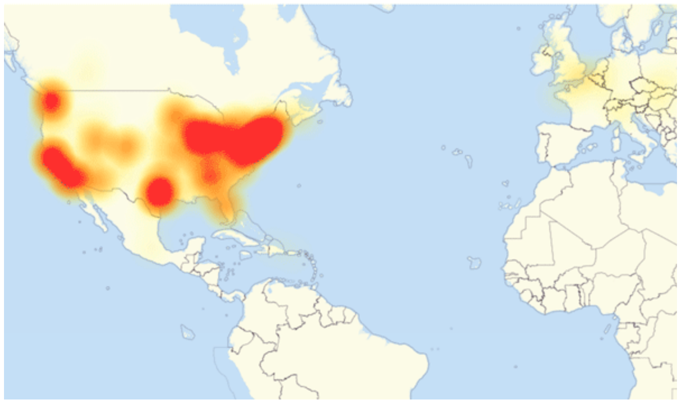


Fig. 3. Internet Outages Caused by Dyn DDoS Attack - Source :Down-Detector

3.1 Technical Analysis

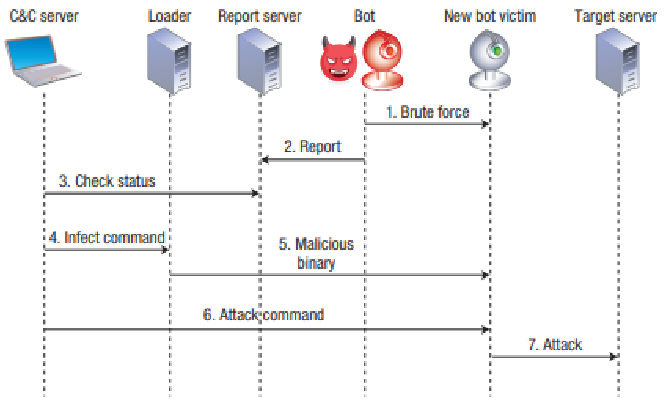
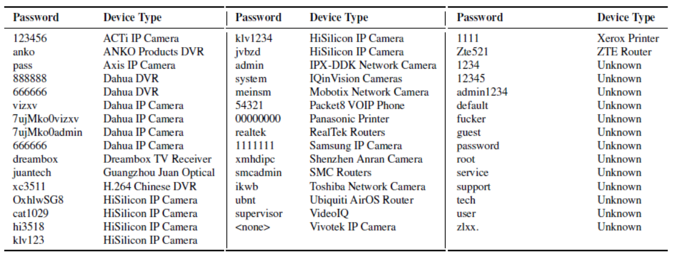


Fig. 4. Mirai Protocol Overview - Source:[9]

Figure 4 represents the infection protocol followed by Mirai bots. An intuitive explanation of this is that infected devices scan pseudorandom IPv4 addresses by sending TCP probes and when they devices that run Telnet or SSH they attempt to login by brute force with a dictionary of hardcoded credentials. The original set of passwords can be found in figure 5, on which the authors of [8] ran an investigative effort to trace them back to the vendors of the vulnerable devices. After a successful login, the responsible



Password	Device Type	Password	Device Type	Password	Device Type
123456	ACTi IP Camera	klv1234	HiSilicon IP Camera	1111	Xerox Printer
anko	ANKO Products DVR	jvbzd	HiSilicon IP Camera	Zte521	ZTE Router
pass	Axis IP Camera	admin	IPX-DDK Network Camera	1234	Unknown
888888	Dahua DVR	system	IQinVision Cameras	12345	Unknown
666666	Dahua DVR	meinsm	Mobotix Network Camera	admin1234	Unknown
vizxv	Dahua IP Camera	54321	Packet8 VOIP Phone	default	Unknown
7ujMko0vizxv	Dahua IP Camera	00000000	Panasonic Printer	fucker	Unknown
7ujMko0admin	Dahua IP Camera	realtek	RealTek Routers	guest	Unknown
666666	Dahua IP Camera	1111111	Samsung IP Camera	password	Unknown
dreambox	Dreambox TV Receiver	xmhdipc	Shenzhen Anran Camera	root	Unknown
juantech	Guangzhou Juan Optical	smcadmin	SMC Routers	service	Unknown
xc3511	H.264 Chinese DVR	ikwb	Toshiba Network Camera	support	Unknown
OxhlwSG8	HiSilicon IP Camera	ubnt	Ubiquiti AirOS Router	tech	Unknown
cat1029	HiSilicon IP Camera	supervisor	VideoIQ	user	Unknown
hi3518	HiSilicon IP Camera	<none>	Vivotek IP Camera	zlxx.	Unknown
klv123	HiSilicon IP Camera				

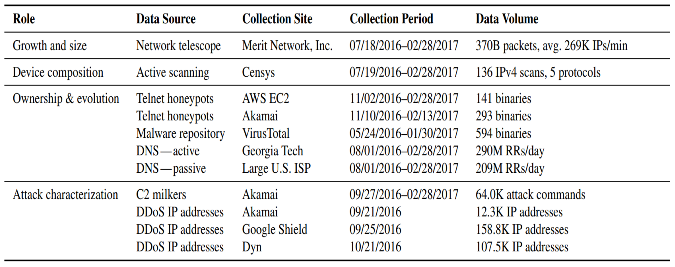
Fig. 5. Mirai Default Passwords - Source:[8]

bot sends the new victim's IP address to the C2 server along with the associated credentials and system information. In an asynchronous response, the server will trigger a loader to infect the device with system specific malware.

Furthermore, upon infection Mirai obfuscates its presence by deleting the downloaded binary and renaming its process into a pseudorandom string. As a consequence, the infection does not persist after a system reboot. Additionally, the malware killed other processes bound to the TCP ports 22 and 23 and processes known to be associated with competing malware (including other variants of Mirai) [8]. The new bot will now follow the same protocol - scan for vulnerable hosts and await (DDoS) commands from the server.

3.2 Investigation

The Mirai investigation comprised a massive forensic effort that included a network telescope, analyzing binaries from telnet honeypots, active scanning and C2 milking as well as analysis of logs from DDOS targets. A detailed overview of the data sources can be seen in figure 6.



Role	Data Source	Collection Site	Collection Period	Data Volume
Growth and size	Network telescope	Merit Network, Inc.	07/18/2016–02/28/2017	370B packets, avg. 269K IPs/min
Device composition	Active scanning	Censys	07/19/2016–02/28/2017	136 IPv4 scans, 5 protocols
Ownership & evolution	Telnet honeypots	AWS EC2	11/02/2016–02/28/2017	141 binaries
	Telnet honeypots	Akamai	11/10/2016–02/13/2017	293 binaries
	Malware repository	VirusTotal	05/24/2016–01/30/2017	594 binaries
	DNS—active	Georgia Tech	08/01/2016–02/28/2017	290M RRs/day
	DNS—passive	Large U.S. ISP	08/01/2016–02/28/2017	209M RRs/day
Attack characterization	C2 milkers	Akamai	09/27/2016–02/28/2017	64.0K attack commands
	DDoS IP addresses	Akamai	09/21/2016	12.3K IP addresses
	DDoS IP addresses	Google Shield	09/25/2016	158.8K IP addresses
	DDoS IP addresses	Dyn	10/21/2016	107.5K IP addresses

Fig. 6. Data Sources - Source: [8]

3.2.1 *Network Telescope*

Using a network telescope operated by Merit Network, Antonakakis et. al analyzed the large footprint left by Mirai's mass-scanning strategy over a seven month period (July 2016 to February 2017). They uniquely identified Mirai by the artifact left during the stateless scanning where every probe contained a TCP sequence number representing the destination IP address. To account for DHCP churn [18], the team grouped packets from a single IP address in a temporal window into logical scans [8] and located the IPs using Maxmind [19].

3.2.2 *Active Scanning*

Using Censys [20], the team actively scanned the IPv4 space and aggregated application layer data about Internet hosts, focusing on scans of HTTPS, FTP, SSH, Telnet, and CWMP between July 2016 and February 2017. To comply with the constraints imposed by the service, limitations and the fact that Mirai disabled outward-facing services such as HTTP, analysis was restricted to banners that were collected within 20 minutes of scanning activity. Using regular expressions provided by the NMap service on top of their own to process the banners of infected devices, they were able to identify 31.5% of the devices' type, model and/or manufacturer. Unfortunately, lack of data made it impossible to get a positive identification for most of the FTP, Telnet, CWMP and SSH banners. More specific information about the identification results can be found in figure 7.

Protocol	Banners	Devices Identified	
HTTPS	342,015	271,471	(79.4%)
FTP	318,688	144,322	(45.1%)
Telnet	472,725	103,924	(22.0%)
CWMP	505,977	35,163	(7.0%)
SSH	148,640	8,107	(5.5%)
Total	1,788,045	587,743	(31.5%)

Fig. 7. Devices Identified - Source: [8]

3.2.3 Telnet Honeypot

In order to monitor the evolution of the malware, the authors of [8] set up Telnet honeypots taking the form of a BusyBox shell [21] to log all incoming Telnet traffic and download & contain all binaries that attackers tried to install on the host via *wget* or *tftp* - infection methods found in Mirai's source code. Furthermore, to avoid actually aiding the botnet, outgoing requests such as scanning and denial of service traffic were blocked. From their own honeypots along with ones that served a similar purpose, but were hosted on a different public cloud provider and operated by Akamai, the team was able to gather 141 unique Mirai binaries, respectively 293 from the Akamai honeypots. Together with other 594 from VirusTotal, a total of 1,028 samples were collected for analysis. 74% of these samples were comprised of MIPS 32-bit, ARM 32-bit and x86 32-bit architectures. By extracting the login dictionaries, IP blacklists and C2 domains from these binaries, a total of 67 C2 domains were identified and 48 distinct dictionaries (containing 371 unique passwords) were found.

3.2.4 C2 Monitoring

To monitor the DDoS attacks activity in real time, Akamai operated a *C2 Milker* for five months that connected to the C2 domains found in subsection 3.2.3. In other words, Akamai simulated a bot in the network and communicated with the C2 server using the Mirai protocol. Attacks were grouped by shared C2 infrastructure and temporal similarity which resulted coverage of 15,194 attacks from 146 unique IP clusters [8], including the Dyn [17] and the Liberia attacks [22].

3.2.5 DDoS Traces Analysis

For a better understanding of the botnet operation, the team behind [8] also investigated network traces and aggregate statistics from Akamai, Google Shield, and Dyn. The purpose of this was to associate the IPs detected in the aforementioned scans with the ones involved in the attacks and to get an idea of the traffic volume generated by Mirai.

4 The FritzFrog Botnet

Since the beginning of 2020, this decentralized botnet has been infecting SSH servers all over the world, mainly focusing on enterprise, government and academic targets. An image of the detected breaches as of August, 2020 can be seen in figure 8. The principal modus operandi of this botnet is similar to Mirai in the sense that it tries to brute force credentials via SSH. Because of its peer-to-peer architecture, the botnet operates on a *no single point of failure* principle and all the nodes communicate constantly to keep the network ”alive, resilient and up to date”. [23] Perhaps the most interesting point about FritzFrog is that it uses a unique, not seen before peer-to-peer protocol. This will be further elaborated in the technical analysis (subsection 4.1). Over 20 variants have been identified in the wild so far.

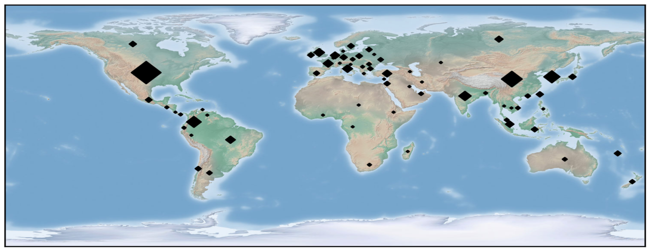


Fig. 8. FritzFrog infection map - Source: [23]

4.1 Technical Analysis

FritzFrog is written in Golang and it works in a highly volatile manner, operating only from the memory under process name such as *ifconfig* and *nginx*, leaving no trace on the disk. That is, with the exception of an SSH key added as a backdoor to ensure that the authors still have access should the credentials change. So far, this key appears to be unique and hardcoded - see appendix C. For investigation purposes, the Guardicore team wrote a program in Golang, which they call *frogger*. The frogger performs the key exchange protocol to infiltrate the network with their own nodes, send commands and receive their outputs. This allowed the researchers to monitor the P2P traffic.

The communication protocol used by FritzFrog is completely proprietary, it is not based on any existing peer to peer protocols. It runs over a symmetrically encrypted channel using AES and bases its key exchange infrastructure on the Diffie-Hellman protocol.

The first step after a successful login is to run the UPX-packed malware under inconspicuous process names and immediately remove it from disk. At this point, port 1234 is used for listening for commands and, usually, the first ones direct the new victim to synchronize with the rest of the network. Because traffic on the 1234 port is unusual it can easily be flagged, so in order to avoid detection and to maneuver through firewall systems, the authors designed the malware to connect the malicious party to the new victim over SSH and run a netcat client which will, in turn, connect to the malware server. Going forward, all commands will be used as netcat input.[23] Figure 9 presents a visual representation of these steps.

The full list of over 30 botnet commands implemented by the authors can be seen in appendix A. Before being sent, commands, along with their parameters and responses are molded into designated data structures and serialized in the JSON format. Then, this data is encrypted using AES. Furthermore, Guardicore Labs researchers observed that targets are evenly distributed through the network through

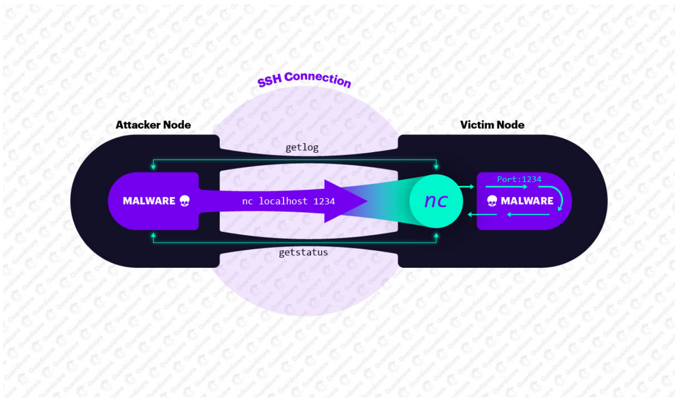


Fig. 9. FritzFrog Protocol - Source: [23]

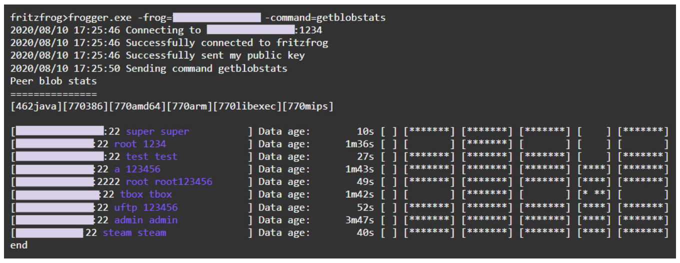
a vote-casting system such that no two nodes attempt to brute force the same target.

The malware is written in a highly advanced manner, running on multiple threads in the memory. Also in the memory, the malware stores the entire database of nodes and attack targets.

Targets are stored in a queue-like data structure and processed through the *Cracker* module, which will scan and attempt to brute force them. The *DeployMgmt* module is, suggestively, managing the malware deployment on breached machines. Once a successful credential is found, the target is queued for infection. The *Owned* module adds a successfully infected machine to the botnet. The full module list can be found in appendix B. Furthermore, a worker thread is always online on infected machines which receives commands, parses them and redirects to the appropriate function in the code.

FritzFrog also runs several, periodic commands to monitor the system state and feed it to the network and determine whether to run the cryptominer or not. If the answer is yes, the malware will start an additional process under the name *libexec* to use the XMRig-based miner on the public mining pool *web.xmrpool.eu:5555*.

Furthermore, as mentioned before, FritzFrog runs filelessly - so how does it share code? In order to achieve this level of stealth, the malware has its files split into small chunks of binary data - known as *blobs* - stored in the memory and divided among nodes. To monitor the available blobs, they are stored in a map along with the corresponding hash value (figure 10).



```
fritzfrog>frogger.exe -frog=[redacted] -command=getblobstats
2020/08/10 17:25:46 Connecting to [redacted]:1234
2020/08/10 17:25:46 Successfully connected to fritzfrog
2020/08/10 17:25:46 Successfully sent my public key
2020/08/10 17:25:50 Sending command getblobstats
Peer blob stats
===============
[462java][770386][770amd64][770arm][770libexec][770mips]

[[redacted]:22 super super           ] Data age:      10s [ ] [*******] [*******] [*******] [    ] [*******]
[[redacted]:22 root 1234             ] Data age:    1m36s [ ] [       ] [*******] [       ] [    ] [       ]
[[redacted]:22 test test             ] Data age:      27s [ ] [*******] [*******] [*******] [    ] [*******]
[[redacted]:22 a 123456              ] Data age:    1m43s [ ] [*******] [*******] [*******] [****] [*******]
[[redacted]:2222 root root123456     ] Data age:      49s [ ] [*******] [*******] [*******] [****] [*******]
[[redacted]:22 tbox tbox             ] Data age:    1m42s [ ] [       ] [*******] [       ] [* **] [       ]
[[redacted]:22 uftp 123456           ] Data age:      52s [ ] [*******] [*******] [*******] [****] [*******]
[[redacted]:22 admin admin           ] Data age:    3m47s [ ] [*******] [*******] [*******] [****] [*******]
[[redacted]22 steam steam            ] Data age:      40s [ ] [*******] [*******] [*******] [****] [*******]
end
```

Fig. 10. getblobstats command output - Source:[23]

In order for a node A to receive a file from node B it sends the *getblobstats* command to receive information about node B's available blobs and then asks for the missing blob(s) by its hash value. This can be done either through the P2P command *getbin* or over HTTP. Node A may have to query multiple nodes to get all the needed blobs, at which point the *Assemble* module assembles them and runs the resulted ”binary”.

When compared to previous botnets, the only similarity - which is, at best, circumstantial - is that function naming and version numbers resemble the ones of Rakos [24], a P2P botnet written in Golang which surfaced around 2016 and also targeted SSH servers.

Known indicators of compromise are fileless processes under the names *ngingx, ifconfig and libexec* and traffic on ports 1234 or 5555 (usually indicating traffic to the mining pool).

5 The Trickbot Botnet

Trickbot was first reported in 2016 and since then it produced tens of millions of dollars worth of damage. Its main purpose, but not limited to, is stealing banking credentials and possibly cryptocurrency wallets information as well. The author(s) put great effort into the design and evolution of this malware piece, as over the years it has evolved to attack PayPal, customer relationship management software, and in 2017 it was upgraded with a self-spreading worm component. Furthermore, as of 2019 it started stealing cookies and targeting security software such as Windows Defender to prevent detection & disinfection and expanded to stealing PIN codes for online platforms such as T-Mobile and T-Mobile Sprint. As if that weren't already bad, in late 2019 the botnet became able to steal OpenSSH and OpenVPN keys, spread itself through the network, bypass Windows User Access Control and steal active directory credentials. To profit by the panic around Covid-19, the phishing campaigns were spreading fake news about the virus and infecting users through it and was considered the most successful one to do so. Also in 2020, the malware was discovered to be able to bypass two-factor authentication of Android systems and, according to Maciej Kotowicz at MalwareLab, it checked whether it was running in a virtual machine by checking the screen resolution. In July 2020, the first Linux infections were reported.[25]–[27]

Trickbot became so notorious that even Forbes published an article about it in [28], stating that the company DeepInstinct discovered a database that had over 250 million email addresses, including US government workers' from agencies such as Homeland Security and NASA, as well as accounts from Canadian and British entities. In figure 11, you can see the global distribution of Trickbot infections.

5.1 Take down attempt

Due to concerns about the safety of the elections in US, as well as all the chaos it has been spreading, in the fall of 2020 Microsoft along with other several private companies and state actors targeted Trickbot in a structured offensive response in an attempt to say that enough is enough. Microsoft won in court the right to do so, setting a legal precedent against malware (however it was not the first attack the company orchestrated against botnets).

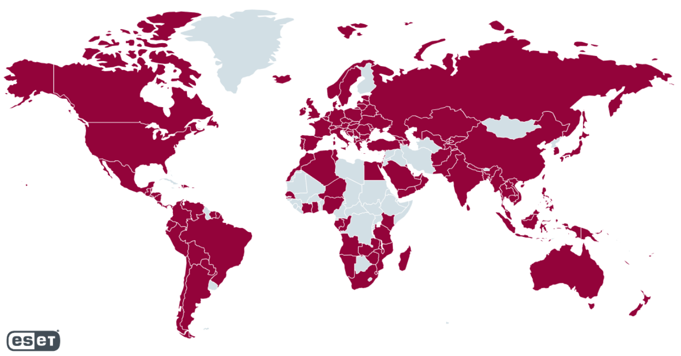


Fig. 11. Trickbot Worldwide Distribution Overview - Source: ESET

To disrupt the botnet, the ”blue” team sent a configuration file that had the command and control address set to localhost (127.0.0.1) to infected computers, blocking them from receiving instructions. A number of command and control domains have also been taken down (through the court order) and there have been other previous actions against Trickbot.

However, this great effort did not cause much damage in the long term as the botnet infrastructure remained largely intact and thanks to the fallback mechanisms that it has implemented it can recover some of the disinfected computers. What was harmed mostly was the reputation of the botnet, as cybercriminals pay a great amount on having access to reliable infrastructure at specific times and this attack proved that Trickbot is not as invulnerable as it was thought. [29], [30]

5.2 Technical Analysis

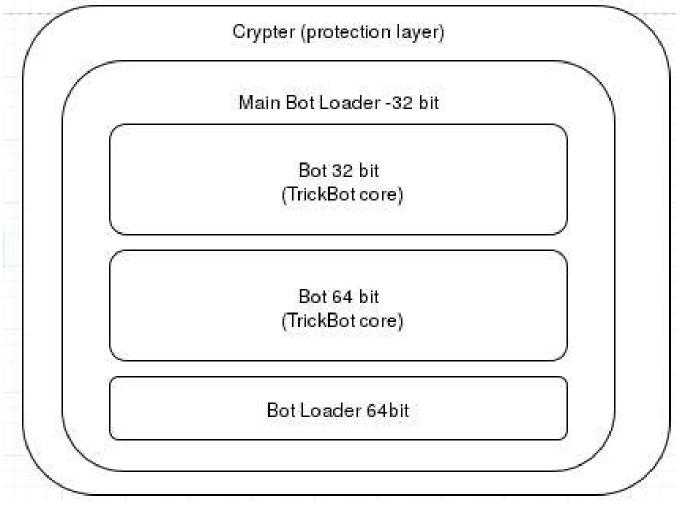

Crypter (protection layer)
Main Bot Loader -32 bit
Bot 32 bit (TrickBot core)
Bot 64 bit (TrickBot core)
Bot Loader 64bit

Fig. 12. Trickbot Architecture Overview - Source: Malware Bytes

Figure 12 provides an overview of the Trickbot architecture. Trickbot uses HTTP requests to download modules and update itself, send (stolen) information to the controlling actor and receive commands (in other words, to communicate with the C2 server).

Figure 13 portraits the typical attack chain implemented by TrickBot and further below I provide more details about

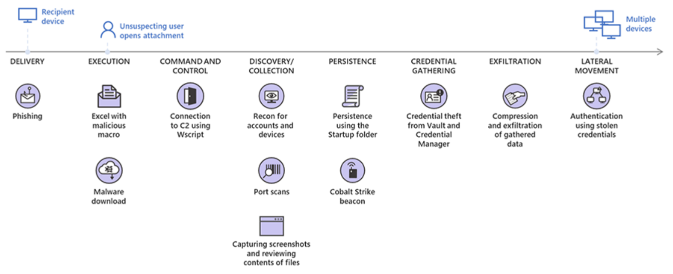

Recipient device
Unsuspecting user opens attachment
Multiple devices
DELIVERY
EXECUTION
COMMAND AND CONTROL
DISCOVERY/ COLLECTION
PERSISTENCE
CREDENTIAL GATHERING
EXFILTRATION
LATERAL MOVEMENT
Phishing
Excel with malicious macro
Connection to C2 using Wscript
Recon for accounts and devices
Persistence using the Startup folder
Credential theft from Vault and Credential Manager
Compression and exfiltration of gathered data
Authentication using stolen credentials
Malware download
Port scans
Cobalt Strike beacon
Capturing screenshots and reviewing contents of files

Fig. 13. Trickbot Attack Chain Overview - Source: Microsoft Security Blog

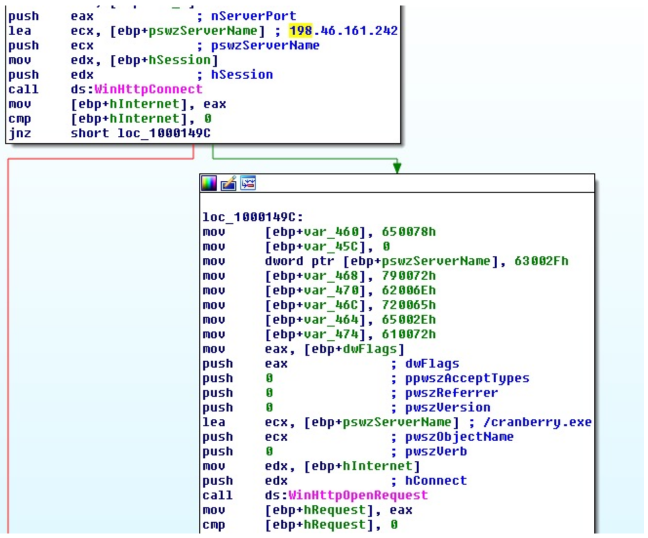
these steps and the modules used. Due to insufficient available information, not all the modules are covered in detail, but a full list of (known) modules can be found in appendix D.

To spread itself, the botnet initially infects a patient zero through spearphishing techniques such as sending fake resumes to human resources departments or fake invoices to accounting that, when opened, execute a script (if and only if the unsuspecting victim enables macros) that runs a Windows PowerShell command. This, in turn, will download the TrickBot binary and execute it. At this point, the malware makes sure that it is run automatically by implementing a task in Windows Task Scheduler. Then, it sends a GET request to the C2 server to inform it about the new victim and its system information, then proceeds with another command to download reconnaissance modules (and others), starting with *pwgrab32*, *systeminfo32*, *injectdll32* and decrypts them.

pwgrab32 is responsible for gathering credentials, autofill data and other information from browsers. In order to do this, the main malware module creates a suspended *svchost.exe* process and injects code from the main module's memory through the *WriteProcessMemory* API and modifies the Original Entry Point to execute the above mentioned code. The credential stealing module is then copied, along with other information such as the C2 server IP list from the main module, and executed [31] to gather credentials from Internet Explorer, Chrome, Firefox and Edge. All the collected information is sent in plain text over HTTP to the C2 server after the process is finished. Furthermore, this module is also able to gather credentials from Outlook, FileZilla and WinSCP. A comprehensive analysis of this module can be found at [31].

Further propagation is done through the *tabDll* module, which exploits the EternalRomance or EternalBlue vulnerabilities in Windows SMB, and/or the *shareDll* module through Windows Network Shares. Besides that, the malware can send phishing emails using the already infested victim's credentials, increasing the chance that unsuspecting coworkers and friends will open the malicious attachments. In 2020, the worm module was changed from *Mworm* to *Nworm*, bringing along a major change in Trickbot's HTTP traffic. From this point in time, infected computers run the malware from memory to ensure that no artefacts are left behind and the binary is encrypted during transfer over the Internet, strengthening its detection avoidance mechanism. [32]

The *mexec* module is responsible for downloading and executing binaries in the memory. It is a new module that appeared in the spring of 2020 and it contains obfuscating elements such as dynamically rebuilding strings during execution to prevent static analysis (see figure 14). The most notable aspect of this module is that it adds another lane for loading downloading and loading other malware on top of the already-present modules of Trickbot. The reverse-engineering analysis of this module can be found at [33].

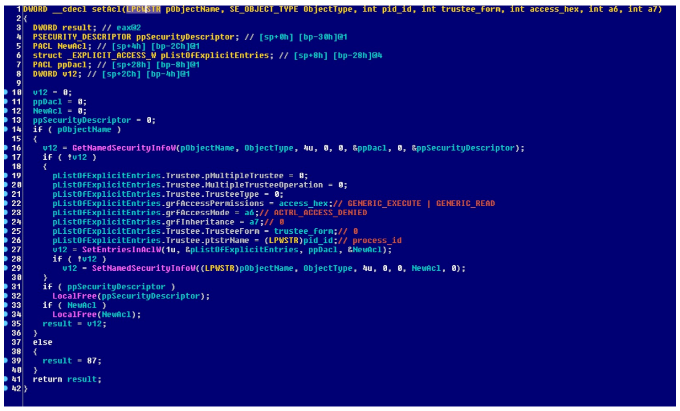


```
push    eax             ; nServerPort
lea     ecx, [ebp+pswzServerName] ; 198.46.161.242
push    ecx             ; pswzServerName
mov     edx, [ebp+hSession]
push    edx             ; hSession
call    ds:WinHttpConnect
mov     [ebp+hInternet], eax
cmp     [ebp+hInternet], 0
jnz     short loc_1000149C
```

```
loc_1000149C:
mov     [ebp+var_460], 650078h
mov     [ebp+var_45C], 0
mov     dword ptr [ebp+pswzServerName], 63002Fh
mov     [ebp+var_468], 790072h
mov     [ebp+var_470], 62006Eh
mov     [ebp+var_46C], 720065h
mov     [ebp+var_464], 65002Eh
mov     [ebp+var_474], 610072h
mov     eax, [ebp+dwFlags]
push    eax             ; dwFlags
push    0               ; ppwszAcceptTypes
push    0               ; pwszReferrer
push    0               ; pwszVersion
lea     ecx, [ebp+pswzServerName] ; /cranberry.exe
push    ecx             ; pwszObjectName
push    0               ; pwszVerb
mov     edx, [ebp+hInternet]
push    edx             ; hConnect
call    ds:WinHttpOpenRequest
mov     [ebp+hRequest], eax
cmp     [ebp+hRequest], 0
```

Fig. 14. Trickbot mexec dissassembly - Source:[33]

The downloaded file will then be written to disk in a destination that is based on the privileges of the running account: if possible, mexec will write to the Windows folder, otherwise it will try AppData and, lastly, the Temp folder. Furthermore, to make sure that it is executed (since it was downloaded from a remote location, previously unknown to the system) the module starts a sequence of Windows API methods to leverage privilege (figure 15).



```
DWORD __cdecl setAcl(LPCWSTR pObjectName, SE_OBJECT_TYPE ObjectType, int pid_id, int trustee_form, int access_hex, int a6, int a7)
{
  DWORD result; // eax@2
  PSECURITY_DESCRIPTOR ppSecurityDescriptor; // [sp+0h] [bp-30h]@1
  PACL NewAcl; // [sp+4h] [bp-2Ch]@1
  struct _EXPLICIT_ACCESS_W pListOfExplicitEntries; // [sp+8h] [bp-28h]@4
  PACL ppDacl; // [sp+28h] [bp-8h]@1
  DWORD v12; // [sp+2Ch] [bp-4h]@1

  v12 = 0;
  ppDacl = 0;
  NewAcl = 0;
  ppSecurityDescriptor = 0;
  if ( pObjectName )
  {
    v12 = GetNamedSecurityInfoW(pObjectName, ObjectType, 4u, 0, 0, &ppDacl, 0, &ppSecurityDescriptor);
    if ( !v12 )
    {
      pListOfExplicitEntries.Trustee.pMultipleTrustee = 0;
      pListOfExplicitEntries.Trustee.MultipleTrusteeOperation = 0;
      pListOfExplicitEntries.Trustee.TrusteeType = 0;
      pListOfExplicitEntries.grfAccessPermissions = access_hex;// GENERIC_EXECUTE | GENERIC_READ
      pListOfExplicitEntries.grfAccessMode = a6;// ACTRL_ACCESS_DENIED
      pListOfExplicitEntries.grfInheritance = a7;// 0
      pListOfExplicitEntries.Trustee.TrusteeForm = trustee_form;// 0
      pListOfExplicitEntries.Trustee.ptstrName = (LPWSTR)pid_id;// process_id
      v12 = SetEntriesInAclW(1u, &pListOfExplicitEntries, ppDacl, &NewAcl);
      if ( !v12 )
        v12 = SetNamedSecurityInfoW((LPWSTR)pObjectName, ObjectType, 4u, 0, 0, NewAcl, 0);
    }
    if ( ppSecurityDescriptor )
      LocalFree(ppSecurityDescriptor);
    if ( NewAcl )
      LocalFree(NewAcl);
    result = v12;
  }
  else
  {
    result = 87;
  }
  return result;
}
```

Fig. 15. Trickbot mexec process security - Source:[33]

Furthermore, the newest Trickbot version has been found capable of "reading, writing and deleting" firmware on infected computers, which would dramatically increase the difficulty of disinfection. On top of that, if companies infected with ransomware refuse to pay, the Trickbot authors could completely destroy their infrastructure. A comprehensive analysis of this module can be found in the joint report at [34].

6 Findings

Table 1 presents a summarized overview for the botnets' analysis by the key metrics defined in section 2.1, along with known indicators of compromise, disinfection suggestions and the status of the literature on that botnet. One of the most important take away from this analysis is that malware can infiltrate even in systems that employ powerful solution if the basic assumptions are not fulfilled. As you can see in the *Growth Mechanism* column in 1, all three analyzed botnets penetrate a network by abusing weak passwords or through phishing. It is my opinion that stronger education is required - both for the end-user, about the risks and best practices, and for the person responsible with designing the security measures in regards to their usability.

Furthermore, technical vulnerabilities have been patched almost four years ago - i.e. EternalBlue - but they are still being exploited in the wild because many devices have not received their updates. This problem exists both in home environments and enterprise.

Another interesting aspect is the way that criminal groups are behaving - they are dedicating focus and resources towards the activity that generates the highest revenue and as they evolve they also diversify their portfolios.

7 Conclusion

Botnets are a rising threat, one that is difficult to control and even harder to shut down. This paper has presented the landscape of this threat and collected key information that was made publicly available by security researchers over the world. Furthermore, insights into the investigation process and techniques are provided to facilitate further analyses on similar malware. The overview generated sets the layout for more complex future work which might involve a deeper analysis on one of these botnets or perhaps a completely different one - should it prove itself more impactful. Such investigation could draw inspiration from the techniques described in section 3 in the form of large-scale traffic monitoring and 4 by infiltrating the network with a mock node. Another possibility is to expand this project and collaborate with other researchers to create a thorough encyclopedia of more botnets and facilitate the education and investigation process even further.

TABLE 1
Botnets overview

Botnet Name	Target	Scope	Control Mechanism	Growth Mechanism	Update Mechanism	Attack Vectors	Disinfection	Symptoms of infection	Literature Status
Mirai	IoT Devices	DDOS	C2	Scanning and bruteforcing	Reinfection wget tftp	Weak passwords and general lack of security measures in IoT devices	Reboot	Unusual Traffic Random named processes	Complete
FritzFrog	SSH Servers Routers and IoT devices	Crypto Mining	P2P	Scanning and bruteforcing	P2P communication HTTP GET requests	Weak passwords and incomplete security measures	Remove SSH key Kill processes	Fileless processes Traffic on 1234, 5555	Relatively new botnet, literature is lacking
Trickbot	Windows Computers	Credential stealing Ransomware Bank Fraud Crypto Mining	C2	Spearphishing SMB Exploitation LDAP	HTTP GET requests	Human EternalBlue,EternalRomance LDAP UEFI/BIOS vulnerabilities	Different tools for home and business	Unusual Traffic Scheduled task Suspicious files	Advanced, but the botnet is constantly evolving

References

[1] *Advertising Fraud Losses to Reach $42 Billion in 2019, Driven by Evolving Tactics by Fraudsters*. Juniper Research. URL: https://www.juniperresearch.com/press/press-releases/advertising-fraud-losses-to-reach-42-bn-2019.

[2] Microsoft. *Trickbot Disrupted - Microsoft Security*. URL: https://www.microsoft.com/security/blog/2020/10/12/trickbot-disrupted/.

[3] Mary Jane Credeur. *EarthLink wins $25 million lawsuit against junk e-mailer's campaign*. URL: https://www.bizjournals.com/atlanta/stories/2002/07/22/story4.html.

[4] Brian Krebs. *'Operation Tovar' Targets 'Gameover' ZeuS Botnet, CryptoLocker Scourge*. URL: https://krebsonsecurity.com/2014/06/operation-tovar-targets-gameover-zeus-botnet-cryptolocker-scourge/.

[5] Kevin Stevens and Don Jackson. *ZeuS Banking Trojan Report*. Secure Works - Counter Threat Unit. URL: https://www.secureworks.com/research/zeus.

[6] *Backdoor:Win32/Pushdo.A*. Microsoft. URL: www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Backdoor%5C%3AWin32%5C%2FPushdo.A#tab=2.

[7] Brett Stone-Gross, Thorsten Holz, Gianluca Stringhini, et al. *The Underground Economy of Spam: A Botmaster's Perspective of Coordinating Large-Scale Spam Campaigns*. URL: https://web.archive.org/web/20140725024735/http://cs.ucsb.edu/~gianluca/papers/cutwail-leet11.pdf.

[8] Manos Antonakakis, Tim April, Michael Bailey, et al. *Understanding the Mirai Botnet*. URL: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis.

[9] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, et al. *DDoS in the IoT: Mirai and Other Botnets*.

[10] Stephan Cole. *An analysis of the evolution of botnets*. URL: https://lib.dr.iastate.edu/cgi/viewcontent.cgi?article=1368&context=creativecomponents.

[11] Rafał Karol Kasprzyk, Marcin Paź, and Zbigniew Tarapata. *Modeling and simulation of botnet based cyber-threats*. URL: https://www.researchgate.net/publication/320204415_Modeling_and_simulation_of_botnet_based_cyber-threats.

[12] Chris Sherry. *Advantages and Disadvantages of Active vs. Passive Scanning in IT and OT Environments*. URL: https://www.infosecurity-magazine.com/opinions/active-passive-scanning/.

[13] M. Bailey, E. Cooke, F. Jahanian, et al. *The Internet Motion Sensor - A Distributed Blackhole Monitoring System*.

[14] Free Software Foundation. *Fail2Ban*. URL: https://www.fail2ban.org/wiki/index.php/Main_Page.

[15] Brian Krebs. *Krebsonsecurity hit with record DDoS*. URL: https://krebsonsecurity.com/2016/09/krebsonsecurity-hitwith-%20record-ddos/.

[16] Octave Klaba. *Octave Klaba Twitter*. URL: https://twitter.com/olesovhcom/%20status/778830571677978624.

[17] S. Hilton. *Dyn analysis summary of Friday October 21 attack*. URL: http://hub.dyn.com/dyn-blog/dyn-analysis-summary-offriday-%20october-21-attack.

[18] K. Thomas, R. Amira, A. Ben-Yoash, et al. *The abuse sharing economy: Understanding the limits of threat exchanges*.

[19] Maxmind LLC. *Geoip2*. URL: https://www.maxmind.com/en/geoip2-city.

[20] Censys. *Censys*. URL: https://censys.io/.

[21] N. Wells. *Busybox: A swiss army knife for linux*.

[22] Brian Krebs. *Did the Mirai Botnet Really Take Liberia Offline?* URL: https://krebsonsecurity.com/2016/11/did-the-mirai-botnet-really-take-liberia-offline/.

[23] Ophir Harpaz. *FritzFrog: A New Generation of Peer-to-Peer Botnets*. URL: https://www.zdnet.com/article/new-fritzfrog-p2p-botnet-has-breached-at-least-500-enterprise-government-servers/.

[24] Peter Kálnai and Michal Malik. *New Linux/Rakos threat: devices and servers under SSH scan (again)*. URL: https://www.welivesecurity.com/2016/12/20/new-linuxrakos-threat-devices-servers-ssh-scan/.

[25] Sergiu Gatlan. *Microsoft: Trickbot in hundreds of unique COVID-19 lures per week*. URL: https://www.bleepingcomputer.com/news/security/microsoft-trickbot-in-hundreds-of-unique-covid-19-lures-per-week/.

[26] Lawrence Adams. *Trickbot malware now checks screen resolution to evade analysis*. URL: https://www.bleepingcomputer.com/news/security/trickbot-malware-now-checks-screen-resolution-to-evade-analysis/.

[27] Lawrence Adams. *Linux Warning: Trickbot is now infecting your systems*. URL: https://www.bleepingcomputer.com/news/security/linux-warning-trickbot-malware-is-now-infecting-your-systems/.

[28] Lee Mathews. *Stealthy TrickBot Malware Has Compromised 250 Million Email Accounts And Is Still Going Strong*. URL: https://www.forbes.com/sites/leemathews/2019/07/14/stealthy-trickbot-malware-has-compromised-250-million-email-accounts-and-is-still-going-strong/?sh=59d222654884.

[29] Jay Greene and Ellen Nakashima. *Microsoft seeks to disrupt Russian criminal botnet it fears could seek to sow confusion in the presidential election*. URL: https://www.washingtonpost.com/technology/2020/10/12/microsoft-trickbot-ransomware/.

[30] Catalin Cimpanu. *TrickBot botnet survives takedown attempt, but Microsoft sets new legal precedent*. URL: https://www.zdnet.com/article/trickbot-botnet-survives-takedown-attempt-but-microsoft-sets-new-legal-precedent/.

[31] Xiaopeng Zhang. *Deep Analysis of TrickBot New Module pwgrab*. URL: https://www.fortinet.com/blog/threat-research/deep-analysis-of-trickbot-new-module-pwgrab.

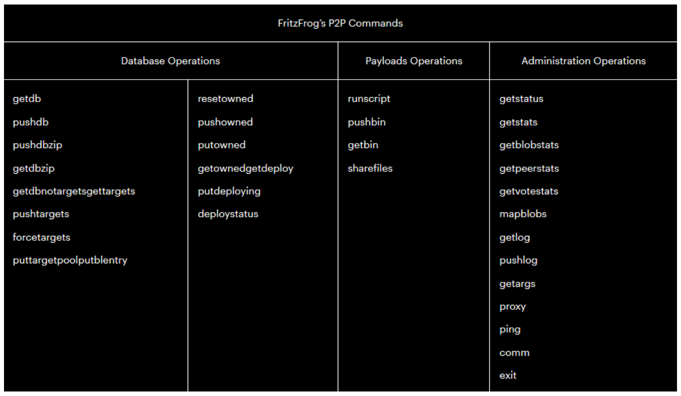
[32] Danny Palmer. *This new Trickbot malware update makes it even harder to detect*. URL: https://www.zdnet.com/article/this-new-trickbot-malware-update-makes-it-even-harder-to-detect/.

[33] Jason Reaves. *Deep Dive Into TrickBot Executor Module "mexec": Hidden "Anchor" Bot Nexus Operations*. URL: https://labs.sentinelone.com/deep-dive-into-trickbot-executor-module-mexec-hidden-anchor-bot-nexus-operations/.

[34] *TRICKBOT NOW OFFERS 'TRICKBOOT': PERSIST, BRICK, PROFIT*. AdvIntel and Eclypsium. URL: https://eclypsium.com/wp-content/uploads/2020/12/TrickBot-Now-Offers-TrickBoot-Persist-Brick-Profit.pdf.

[35] *TrickBot*. MS-ISAC. URL: https://www.cisecurity.org/wp-content/uploads/2019/03/MS-ISAC-Security-Primer-Trickbot.pdf.

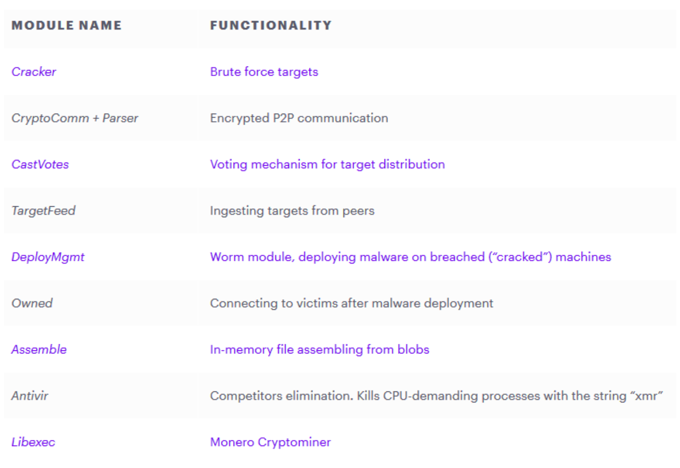
Appendix A
FritzFrog Commands



FritzFrog's P2P Commands			
Database Operations		Payloads Operations	Administration Operations
getdb	resetowned	runscript	getstatus
pushdb	pushowned	pushbin	getstats
pushdbzip	putowned	getbin	getblobstats
getdbzip	getownedgetdeploy	sharefiles	getpeerstats
getdbnotargetsgettargets	putdeploying		getvotestats
pushtargets	deploystatus		mapblobs
forcetargets			getlog
puttargetpoolputblentry			pushlog
			getargs
			proxy
			ping
			comm
			exit

Fig. 16. FritzFrog Command List - Source: [23]

Appendix B
FritzFrog Modules



MODULE NAME	FUNCTIONALITY
Cracker	Brute force targets
CryptoComm + Parser	Encrypted P2P communication
CastVotes	Voting mechanism for target distribution
TargetFeed	Ingesting targets from peers
DeployMgmt	Worm module, deploying malware on breached ("cracked") machines
Owned	Connecting to victims after malware deployment
Assemble	In-memory file assembling from blobs
Antivir	Competitors elimination. Kills CPU-demanding processes with the string "xmr"
Libexec	Monero Cryptominer

Fig. 17. FritzFrog Module List - Source: [23]

Appendix C
FritzFrog Backdoor Key

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQD
JYZIsncBTFc+iCRHXkeGfFA67j+kUVf7h/IL+sh
0RXJn7yDN0vEXz7ig73hC//2/71sND+x+Wu0zyt
QhZxrCPzimSyC8FJCRtcqDATSjvWsIoI4j/AJyK
k5k3fCzjPex3moc48TEYiSbAgXYVQ62uNhx7ylu
g50nTcUH1BNKDiknXjnZfueiqAO1vcgNLH4qfqI
j7WWXu8YgFJ9qwYmwbMm+S7jYYgCtD107bpSR7/
WoXSr1/SJLGX6Hg1sTet2USiNevGbfqNzciNxOp
08hHQIYp2W9sMuo02pXj9nEoiximR4gSKrNoVes
qNZMcVA0Kku01uOuOBAOReN7KJQBt
```

Appendix D
TrickBot Modules

Banking Information Stealers

- LoaderDll/InjectDll – Monitors for banking website activity and uses web injects (e.g. pop ups and extra fields) to steal financial information.
- Sinj – This file contains information on the online banks targeted by TrickBot and it uses redirection attacks (also known as web fake injections).
- Dinj – This file contains information on the online banks targeted by TrickBot and it uses server side web injections.
- Dpost – Includes an IP address and port for stolen banking information. If the user enters banking information for one of the listed banks, the information is sent to the dpost IP address. Most of the data exfiltrated by TrickBot is sent to the dpost IP address.

System/Network Reconnaissance

- mexecDll/aexecDll/onixDll - Acts as a downloader & loader for other modules
- Systeminfo – Harvests system information so that the attacker knows what is running on the affected system.
- permaDLL checks for administrator privileges on the system.
- Mailsearcher – Compares all files on the disk against a list of file extensions.
- NetworkDll – Collects more system information and maps out the network. Credential and User Information Harvesting
- ModuleDll/ImportDll – Harvests browser data (e.g. cookies and browser configurations).
- DomainDll – Uses LDAP to harvest credentials and configuration data from domain controller by accessing shared SYSVOL files.
- OutlookDll – Harvests saved Microsoft Outlook credentials by querying several registry keys.
- SqulDll – Force-enables WDigest authentication and utilizes Mimikatz to scrape credentials from LSASS.exe. The worming modules use these credentials to spread TrickBot laterally across networks.
- Pwgrab – Steals credentials, autofill data, history, and other information from browsers as well as several software applications.

Network Propagation

- WormDll and ShareDll – These are worming modules that abuse Server Message Block (SMB) and Lightweight Directory Access Protocol (LDAP) to move laterally across networks.
- TabDll – Uses the EternalRomance exploit (CVE-2017-0147) to spread via SMBv1.

Source: [35]