Analysis of 24 Hours Internet Attacks

A Brief Overview of Malicious Traffic Targeting Featureless Servers on the Web

Tim Britton [1], Ian Liu-Johnston [1], Ian Cugnière [1], Swati Gupta [1], Danton Rodriguez [1], Julien Barbier [1], Sebastien Tricaud [2]

[1] Holberton School

[2] Honeynet Project

Abstract:

For the past decades, bots and botnets have been on the front page of newspapers and are one of the main topics of discussion in the news media. The range of the attacks and their targets have been increasing.[1] A recent example, the Mirai network - a botnet built through insecure Internet of Things (IoT) devices -, has been at the center of attention after it provoked an internet outage primarily on the East Coast.[2] A study also found that "80 percent of spam was sent by botnets by 2009".[3] Despite this, most of our everyday life relies heavily on the internet and is still vulnerable to malicious attacks. This paper aims to explore where such attacks originate and how the attacks occur. We set up and decided to observe what happens to an internet-facing server that should not encounter anything but local network activity. To investigate further, we set up honeypots on that server to see how the flow of traffic changed, and what bots and other clients would do. We wish to share our findings and thus humbly contribute to more awareness about the risks faced by anyone using the internet.

1. Introduction & Methodology

In order to observe malicious activity on the internet we set clear steps to decide which methodology to adopt based on the traffic we would receive.

We first set up a bare Amazon AWS instance whose data center resides in Ashburn, Virginia. To get a general overview of the traffic that would come to the server, we did not run any services that would be useful to anybody else, and did not connect the Internet Protocol (IP) address to any domain name. Very shortly after renting the server, we set up a packet capture for a 24-hour period with tshark/wireshark. We did this to identify the most promising traffic/protocols to observe and concentrate our efforts on. We then analyzed the packet capture file with tools such as tshark/wireshark, Computer Incident Response Center's (CIRCL) Border Gateway Protocol (BGP) ranking API, and p0f.

1.1 BGP Rankings

CIRCL's BGP Ranking provides data to calculate the security ranking of Internet Service Providers (ISPs).[4] We compared the list of IP addresses that accessed our server during the initial 24-hour period to calculate the risk assessment for these IP addresses.[5] The closer the BGP ranking is to zero, the more malicious the IP address is. At the time, the most malicious IP in our logs scored 0 out of 13,043. This IP address accessed the TCP port 3380, which is used by the SNS Channels. Other notable ports and protocols that were accessed by high risk IP addresses were various ephemeral ports such as the Intel

Remote Desktop Management Interface (IRDMI) protocol, the port 8089, and the port 81 (TorPark Onion routing). See Table 1.0 for the list of the most malicious IP addresses we captured (as of February 13rd, 2017):

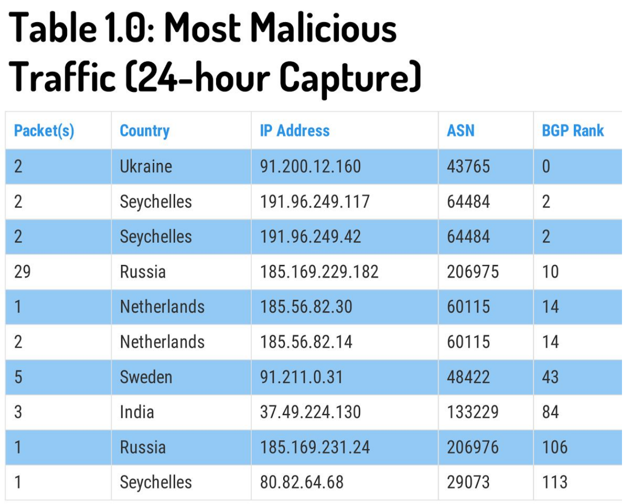


Table 1.0: Most Malicious Traffic (24-hour Capture)

Packet(s)	Country	IP Address	ASN	BGP Rank
2	Ukraine	91.200.12.160	43765	0
2	Seychelles	191.96.249.117	64484	2
2	Seychelles	191.96.249.42	64484	2
29	Russia	185.169.229.182	206975	10
1	Netherlands	185.56.82.30	60115	14
2	Netherlands	185.56.82.14	60115	14
5	Sweden	91.211.0.31	48422	43
3	India	37.49.224.130	133229	84
1	Russia	185.169.231.24	206976	106
1	Seychelles	80.82.64.68	29073	113

We realized that some protocols were attracting remarkably more traffic than others. For example, for application layer protocols, we recorded 255,796 connection attempts through Secure Shell (SSH), while we received only 1 connection to the Connection-less Lightweight Directory Access Protocol (CLDAP), cf. table 1.1 in Annex.

1.2 Protocol Selection

While the scope of this paper might seem enormous, if not quixotic, after parsing the data, we decided to focus our efforts on the following protocols: the Hypertext Transfer Protocol (HTTP), the Session Initiation Protocol (SIP), SSH, and the Telecommunications Network (Telnet) protocol. These protocols were selected for the following reasons:

- The HTTP protocol is the most widely used protocol on the internet, and thus provides the biggest resource for exploits;
- The SIP protocol is often used for enterprise-level telecommunication systems;
- The SSH protocol represents many bruteforce attempts and accounts for more than half of the total traffic from our initial packet capture;
- The Telnet protocol: The Mirai botnet and its successor Hajime exploit this protocol, as IoT devices - which often use Telnet - comprise the majority of targeted systems.

We thus used honeypots to attract the traffic that targets the protocols above and analyzed the data they produced.[6] Honeypots are pieces of software designed to reproduce the same functionality as vulnerable servers, and can act as decoys to attract intruders.

In this paper, we will first detail the traffic observed per protocol. Then, we will expand on the botnets' patterns that we were able to notice. It will allow us to better grasp how these botnets infect and interact with a targeted system.

2. Protocols of interest

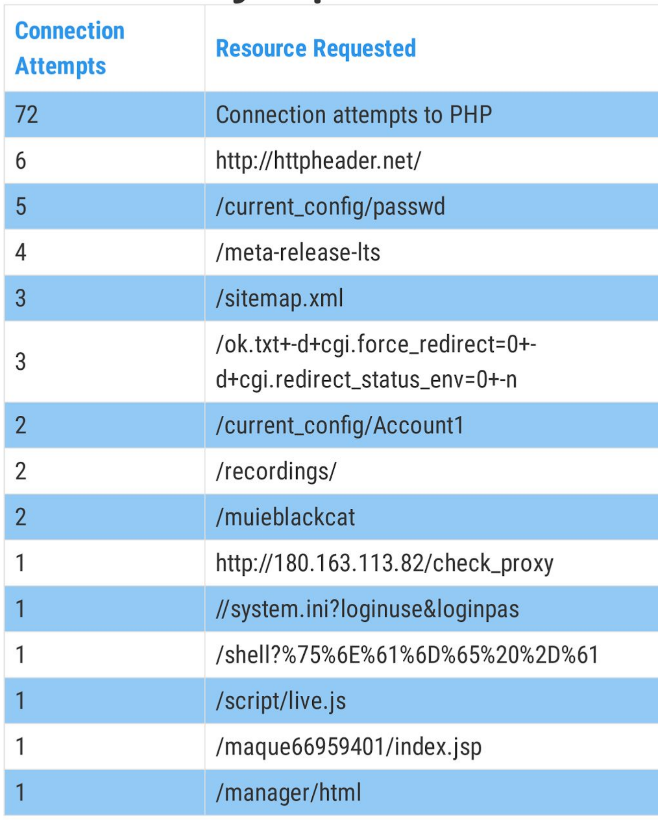
2.1 HTTP

"The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers." [7]

Because HTTP is one of the most widely used protocols on the internet, it is also one of the main vectors for exploitation. We recorded 245 HTTP packets during our initial 24-hour packet capture.

To investigate this protocol further, we setup a web-application honeypot, Glastopf, running in a Docker container. This honeypot provided basic functionality to record GET or PUT requests, IP addresses, Uniform Resource Identifiers (URIs), and timestamps. It was also designed to simulate SQL vulnerabilities and record injection attempts. We launched this honeypot alongside another packet capture for a second 24-hour period on March 7th, 2017.

Table 2.0: Overview of Glastopf URIs (12-day capture)



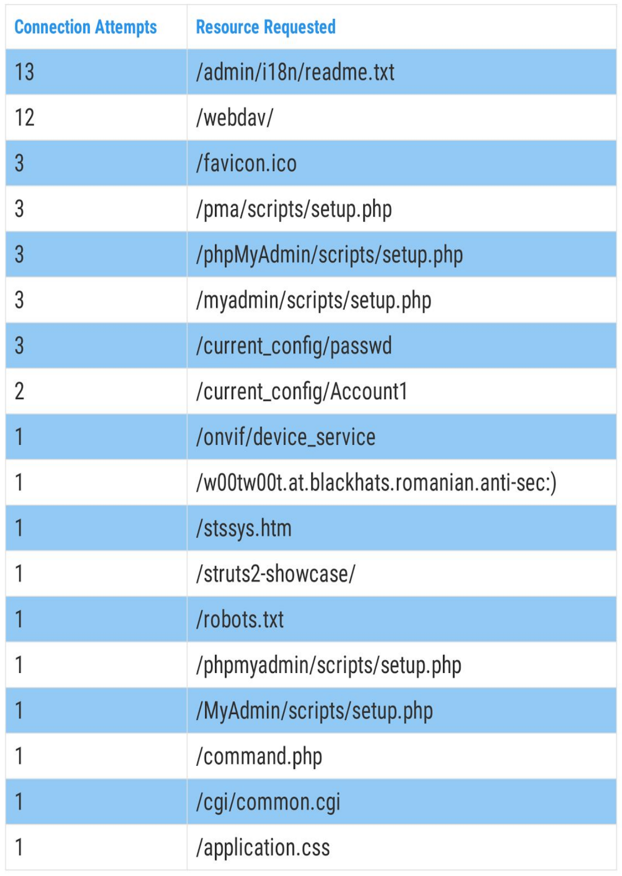
Connection Attempts	Resource Requested
72	Connection attempts to PHP
6	http://httpheader.net/
5	/current_config/passwd
4	/meta-release-lts
3	/sitemap.xml
3	/ok.txt+-d+cgi.force_redirect=0+-d+cgi.redirect_status_env=0+-n
2	/current_config/Account1
2	/recordings/
2	/muieblackcat
1	http://180.163.113.82/check_proxy
1	//system.ini?loginuse&loginpas
1	/shell?%75%6E%61%6D%65%20%2D%61
1	/script/live.js
1	/maque66959401/index.jsp
1	/manager/html

The resulting logs from Glastopf did not yield much data, possibly because of specialized functionality, and possibly because Glastopf is an older honeypot. In total only 14 connections were made, two of which were requests for `meta-release-lts`, and `/phpmyadmin/scripts/setup.php`, while the rest were for the website root.

To give us a broader picture of possible connections, we decided to run Glastopf for a much longer period of time. In addition, we wrote a basic honeypot in NodeJS to run simultaneously and compare the traffic from both servers.[8] The custom honeypot served a static authentication page and had logging capabilities. We ran both web servers on port 80 for a 12-day period.

By doing so, we were able to cross-reference the data obtained from those sources, which gave us a more refined understanding of the kind of attacks currently targeting the HTTP protocol (cf. Table 2.0 and Table 2.1).

Table 2.1: Overview of NodeJS URIs



Connection Attempts	Resource Requested
13	/admin/i18n/readme.txt
12	/webdav/
3	/favicon.ico
3	/pma/scripts/setup.php
3	/phpMyAdmin/scripts/setup.php
3	/myadmin/scripts/setup.php
3	/current_config/passwd
2	/current_config/Account1
1	/onvif/device_service
1	/w00tw00t.at.blackhats.romanian.anti-sec:)
1	/stssys.htm
1	/struts2-showcase/
1	/robots.txt
1	/phpmyadmin/scripts/setup.php
1	/MyAdmin/scripts/setup.php
1	/command.php
1	/cgi/common.cgi
1	/application.css

In our observations of the Glastopf logs, the majority of client requests were for PHPMyadmin and vulnerable Wordpress plugins. There were various other requests to interesting URIs such as `/system.ini`, `/struts2-showcase/`, and `/current-config/passwd`. Regardless of where the requests originated from, be it Bangalore, Austria or the United States, the attacks were fairly homogenous between our honeypots and relied on well-known attack vectors.

The most interesting log entries for the NodeJS honeypot provided links to trojans and other binaries. They were sent to our servers in various HTTP headers, with the ability to install and execute malware through known exploits.

One type of malicious request attempted to leverage an exploit known as "Shell Shock" or "Bashdoor",[9] which was made public in 2014. It exploited a vulnerability in web servers that uses Bash to process the User-Agent header for certain requests. The User-Agent string would be sent as a malformed function definition with bash commands immediately following the function definition. Such a string could adopt the following format:

```
( ) { :;}; /bin/bash -c 'echo vulnerable'
```

The Shellshock exploit in our logs attempted to download a shell script, a Perl script, and a tar archive renamed to have a .jpg extension. The tar archive compressed all other files. The shell script created a crontab that would re-download the trojan and verify that the service was running. All Bash scripts ensured persistence for the Perl script named "DDoS Perl IrcBot v1.0 / 2012 by DDoS Security Team". We also encountered a modified version of the Perlbot that also leveraged the Shellshock vulnerability in other logs.

Another series of notable exploit attempts leveraged a recent vulnerability in the Apache Struts web framework.[10]

This exploit was revealed March 7th, 2017. The initial log entries were dated shortly after the exploit was publicly released. There were multiple exploit attempts that used the same methodology as in the exploit-db python script.[11] The exploit works by leveraging a vulnerability in parsing the Content Type header field, which allows remote code execution.[12]

Subsequent attempts were enumeration techniques from Metasploit modules,[13] that followed this convention:

```
%{#context['com.opensymphony.xwork2.
dispatcher.HttpServletResponse'].add
Header('tkplpyg','tkplpyg')}.multipa
rt/form-data
```

The first few malicious clients downloaded a malicious script for Linux and a binary named "UnInstall.exe" for Windows. This binary was also found during the same time period in a pirated torrent of Skyrim.

The Linux script stopped all firewalls, set the DNS server to 8.8.8.8, removed any instances of Apache from /etc/init.d/ and instances of the malicious binary. Then it downloaded a series of dynamic libraries as well as the malicious binary masquerading as an Apache process.

The binary itself was a bitcoin miner based off of cpuminer2.3.3, that mined into the Stratum mining pool.[14] It used the hash algorithm cryptonight, with the username "sqwukiomcage." This illustrates the recent trend of increasing attacks aimed at cryptocurrencies.

Next, it set a crontab to download and execute the script again, and checked through "known_hosts" in all ".ssh" directories, and in all ".bash_history" files to find any servers that were connected through ssh. Finally it downloaded the trojan onto those new servers.

2.2 Telnet - esp. IoT Honeypot

"The purpose of the TELNET Protocol is to provide a fairly general, bi-directional, eight-bit byte oriented communications facility. Its primary goal is to allow a standard method of interfacing terminal devices and terminal-oriented processes to each other. It is envisioned that the protocol may also be used for terminal-terminal communication ("linking") and process-process communication (distributed computation)." [15]

Telnet is an unsafe protocol as all information (including authentication) is exchanged over plain text. Nonetheless, it is still used for IP cameras, routers and other IoT devices to allow remote system administration, such as firmware upgrades. It is used instead of SSH on these devices because it is easy to implement and lightweight, making it acceptable for devices without a lot of resources.

Telnet was the second port to attract the most traffic, with 606 interactions detected during the 24-hour capture. We decided to study Telnet because of the aforementioned trend regarding botnets, such as Mirai, targeting IoT devices. Our intention was to study how those trends - noted by many experts - translate in near real-time observation. We thus decided to investigate further.

In order to do so, we installed H-M-S Telnet honeypot[16] in order to become the target of a more consequent flux of attacks/interactions. This honeypot has no specific content but does mimic a GoAhead wifi camera based on an exploit uncovered by Pierre Kim.[17]

Focusing on IoT devices paid off as the traffic captured demonstrates. On March 31st, we saw 1,075 connections, with a total of 306 unique IP addresses, connect to the Telnet honeypot server. 835 of those 1,075 connections resulted in the connecting party successfully logging in (providing any two lines of input for a username and password), with 267 unique IPs logging in. Of the 835 logins, 750 were generated by bots that successfully connected and created enough traffic to fit into an identifiable pattern. We observed patterns left by the following variants of Mirai: Mirai scanner, Mirai/Ecchi downloader, OBJPRN Mirai variant, Hajime downloader, and the Mirai 'xkajdnabw' variant. Around November 2016, the Mirai source code was released to the general public. Mirai's original source code continues to be modified and new variants spread and keep infecting vulnerable devices. The variants mentioned previously echo concerns about Mirai being an easily replicable and highly adaptable botnet.[18]

The countries of origin from our analysis directly correlated to some of those that were observed by experts in the field of cybersecurity: Taiwan, Russia, South Korea, Turkey, China, the United States, Brazil, and Iran (see Map 2.0).[19]

2.3 SIP

SIP is based on an HTTP-like request/response transaction model and is used to establish, modify, and terminate multimedia sessions such as audio or video calls, Internet telephony call.[20]

The SIP protocol caught our attention for several reasons. Not only did it account for a significant portion of the traffic we received, but most of its applications are in enterprise telecommunication systems.

However, SIP ended up being a disappointing lead as the only packets we observed were generated by SIPVicious.

As noted by Cisco:

"The tool could also be used to scan the IP or VoIP telephony network. Due to a flaw in the processing of SIP messages by the telephony device firmware, an attacker could use any number or any SIP address in the INVITE message to scan random networks to determine availability of live hosts. The attacker could initiate an INVITE session and determine a successful detection by receiving a phone ring as a response. This detection could allow the attacker to conduct further attacks such as host spoofing to make phone calls using the detected IP phone identity."[21]

It can also be used in a non-malicious way to audit a network.[22]

All the packets were transferred over UDP, using the second version of the SIP protocol. The headers of the SIP packets show connection attempts. The To and From Header fields indicate that after the initial INVITE request, attackers tried random strings in the Contact header field to directly connect with a user at another end.

Let us note that the results of this traffic may be attributed to a recent release of SIPVicious. The latest release of SIP vicious occurred on February 4th. We ran our initial packet capture concurrently on February 4th - 5th.

2.4 SSH

"The Secure Shell (SSH) is a protocol for secure remote login and other secure network services over an insecure network." [23]

Although statistically we got an enormous volume of SSH connection attempts, we decided not to investigate the protocol as thoroughly as HTTP or Telnet because the traffic is mostly comprised of brute force attempts.

While the number of connection attempts was significant, the vast majority of the attempts came from three IP addresses whose geolocation data corresponded to the Guangdong and Jiangsu areas in China. To be precise, 130,598 of the 140,606 SSH connection attempts we analyzed, came from just three IP addresses. In fact, over 99.8% or 140,417 of the SSH connection attempts appeared to have originated from China. The highest number of connection attempts from an IP address whose geolocation data was outside of China was 49.

A general trend we noticed by analyzing the auth logs from the 24-hour capture on the AWS server was IP addresses trying to connect via SSH as root on all ports from 1-65535, which may explain the high volume of traffic from individual IP addresses. When we analyzed the auth logs from the DigitalOcean server, we noticed that there were a number of bruteforce attempts against a range of user names, but not against a range of ports, contrary to the AWS instance.

3. Botnets detected

After having briefly mentioned the Botnets encountered, we will expand on that subject as it is one of the most notable and worrying trends in today's security landscape.

3.1 Hajime

According to Rapidity Network, Hajime was first spotted around October 5th, 2016.[24] The traffic generated is similar to Mirai's; after sequentially using a list of randomly generated passwords, Hajime attempts to open a new shell, and uses the `/bin/busybox` trick seen with Mirai (with ECCHI normally) to verify whether it is inside an actual shell. Then, it checks `/proc/mounts`, and attempts to go to a seemingly random directory and to run a set of commands to ensure that it can actually write to the directories listed. Hajime also checks to see if it has access to netcat or wget, and finally, runs the Data Duplicator (DD) command against /bin/echo to grab the first 52 bytes and analyze the Executable and Linkable Format (ELF) header, to get architecture information for the current machine.

Finally, Hajime will download a script for the appropriate architecture with `wget` (if accessible and if the control server has that resource).

```
2017-03-31 05:22:16,698 - RECEIVED
INPUT 202.174.185.162 : ['rm .s;
wget http://73.9.22.205:49511/.i;
chmod +x .i; ./.i; exit']
2017-03-31 05:22:17,498 - RECEIVED
INPUT 202.174.185.162 : ['q']
2017-03-31 05:22:17,504 - Lost
connection to 202.174.185.162:39189
```

Other researchers have noted that it will attempt to echo-assemble a binary if wget and nc are unavailable. In this case, the downloaded malware has an MD5 sum of `91a02956678c4ff6aa9075cfe99db24d`.

The purpose of Hajime is unknown and open to speculation. The Hajime binary actually attempts to block the ports that Mirai uses to communicate with its command and control (C&C) servers, displaying the following message on infected systems:

```
Just a white hat, securing some
systems.
Important messages will be signed
like this!
Hajime Author.
Contact CLOSED
Stay sharp!
```

Like the other botnets found, Hajime's effects are only stored in memory, and resetting the device will clear it back to the same insecure settings Hajime found the device with.

3.2 Mirai

Mirai is likely the most famous botnet, found to be responsible for a record-breaking 620 Gbps DDoS attack.[25] On October 21st, 2016, Dyn's DNS infrastructure was hit with an attack of a similar Gbps output, drawing Mirai into the public spotlight.[26] On Twitter, Octave Klaba, the founder and owner of OVH, reported attacks thought to stem from Mirai, reaching over 1Tbps.[27] Many researchers have done their own study of Mirai whose source code was released around October 1st, 2016, on Hackforums, by a user called Anna-Senpa.[38] This latter source includes everything from the scanner, the loader, even an API where customers can be allowed access to the C&C server to direct attacks.

As a result of the source being leaked, countless modified strains have been found in the wild and continue to be found.

Mirai first scans the environment it has penetrated. It logs on by randomly selecting username/logins from a predetermined list. Once it is in the system, it attempts to determine whether it is in a shell or not using shell and, checking access to the `/bin/busybox MIRAI` path to ensure it is receiving the proper response (`MIRAI: applet not found`).

If it receives the expected response, the Mirai downloader, ECCHI, shows up next. It starts out with similar commands to the scanner, running shell and sh, and then `/bin/busybox ECCHI`. Next, it checks the output of ps so as to discover other running processes. The Mirai source carries a list of processes to destroy, namely other botnets. If any are found, it will run `kill -9` against them. This is a territorial act, as it wants to ensure it is the only botnet in control of the device.

Like Hajime, it checks `/proc/mounts` for mounted filesystems and cycles through them to determine access.

To prep for the impending download, Mirai copies `/bin/echo`, empties it, and then `chmod`'s it to ensure it can execute the file. The last step before it can download is to cat `/bin/echo`. The bot ignores everything but the ELF header, and simply checks it for the architecture information.

Then, it checks for wget and tftp, and uses whichever is available. Mirai downloads the architecture specific file. Mirai supports x86, MIPS, MPSL, ARM, ARM5, ARM7, PPC, SPC, M68K, and SH4 architectures, and the source actually includes a cross compiler for this reason. Then it downloads it to the prepared dvrHelper file and runs it. That done, it runs the script and exits out, its mission accomplished.

The malicious binary causes a connection between the device and the Mirai C&C server. The device will begin the same scanning routine seen previously, alerting the C&C server of insecure bots so it can be hit by a loader and added to the botnet. The connection can also be used to broadcast a multitude of DDoS attacks to the clients: UDP flood, Valve query flooding, 'DNS water torture', Synchronization Acknowledgment (SYN/ACK) floods, Generic Routing Encapsulation (GRE) IP, Ethernet flooding, and HTTP flooding.

3.3 Bashlite

The Bashlite malware, like Qbot and its derivatives, has been around since at least early 2015 and the source is easy to find.[29] HackForums and LeakedFiles list several, and even GitHub has several repos with the source. The HackForums user Anna-Senpai even references it in the original forum post where the Mirai source code was posted: "However, I know every skid and their mama, it's their wet dream to have something besides Qbot." and then references the speed of Mirai versus Qbot: "Bots brute Telnet using an advanced SYN scanner that is around 80x faster than the one in Qbot, and uses almost 20x less resources."

A quick Google search results in numerous YouTube videos and forum posts with tutorials on how to set up Qbot. Most versions of Qbot lack any kind of honeypot detection, and simply log in and try to dump their malware. Each client malware is cross-compiled for different architectures, as we can see at one of the GitHub links.[30] NTPD is MIPS, bash is x86, tftp is ARMv6, etc. Basically, it attempts to download every possible file and run every possible one, hoping one will be the right architecture to infect the client.

Qbot/bashlite generally contains capabilities for HTTP, UDP and TCP flooding attacks, but with so many sources available the attack vectors can differ from bot to bot.

3.4 Bricker Bot

Bricker Bot made the headlines May 4th, 2017, after a RadWare article revealed details about the bot.[31] It returned to the spotlight on April 21st, 2017, with a BleepingComputer article that claimed to have found the author of Bricker Bot.[32]

Bricker Bot is a grey-hat bot. It attempts to log in to insecure Telnet devices and disable them, either temporarily or permanently. Bricker Bot uses Tor exit nodes to conceal the actual IP of the attacking device.

Its self-proclaimed creator, under the nickname janit0r, claims to have bricked around 2 million devices. According to Radware Bricker Bot would have permanently disabled some IoT devices, but its actual impact remains unknown.

On April 2nd, 2017, the honeypot server saw several different attack patterns matching either Bricker Bot or that of a similar botnet. Variations of the Bricker Bot attack pattern were seen 21 times, originating from 6 different IPs. The bot collected the OS information from `/proc/version` and `uname -a`, checks shadow/passwd, and checked the contents of `/etc/`. Then, after running su root to ensure that the user was root, it attempted to run a forkbomb by creating a function that repeatedly forked itself:

```
d(){ d|d & };d 2>/dev/null
```

Traffic received in the same timeframe and for several days caused the honeypot to hang on the passwd command, which the bot seemed to follow minutes later by trying to kill every running process and reboot.

Another type of traffic encountered, more malicious, redirected /dev/urandom to random device descriptors, attempting to clear the IP route and setting max threads to one.

After fixing the bug that caused the honeypot to hang on the 'passwd' command, we could see that the bot was attempting to change the password to a randomly generated string, which was different each time it connected.

The honeypot was modified for the passwd command to 'work'; running passwd would lock the honeypot out to anyone but those using the provided password, and in 48 hours of uptime, we did not witness any login attempt using the generated password.

On April 4th, 2017, the honeypot went down for approximately 6 hours and thereafter stopped receiving traffic. It is worth noting that we had set up two other honeypots - one of them in the same DigitalOcean region - that never received any traffic from that bot. The only difference being that the honeypot attracting Bricker Bot traffic was spun up in early March, whereas the other two honeypots were not started until the end of March.

The last time the honeypot monitored any Bricker Bot-like activity was April 24th, 2017, with some slight differences compared to the previous patterns observed. This time, it left a message in the 'message of the day' (motd) file warning that the system had been hacked, and then immediately returned to forkbomb it. The traffic resulted in 180 logins in one day, from two IPs.

Conclusion

More than twenty years ago, John Perry Barlow in his *Declaration of the Independence of Cyberspace* called for a civilization "more humane and fair" than that of governments. However, and as mentioned above, the number of malware attacks keeps increasing at an incredible pace. When we started our research for this paper, Mirai was the most malicious botnet in terms of scope. As of this writing WannaCry/WannaCrypt seems poised to take its place in the headlines.

The source of attempted exploits observed spans across the whole globe. Featureless servers encounter a consequent amount of traffic, but that volume is even greater when the server has a purpose, such as an IP camera, or a Wordpress website with a domain name.

Interestingly, we discovered that most of the attacks rely on old malware, which tends to indicate that those attacks are still successful and thus that steps to prevent them are largely ignored. Indeed, these exploits mostly rely on improperly configured or outdated software, and generic username/password combinations.

The attack patterns we recorded for HTTP, SSH, and SIP relied on generic exploit attempts that seemed to scan a range of IP addresses for well-known vulnerabilities. Telnet, on the other hand, relied on even simpler intrusion methods, by bruteforcing with default username and password combinations. Sometimes, these spray-and-pray attacks immediately attempted to download antiquated scripts, or more contemporary trojans, but none of the recorded attempts were covert enough to evade detection or overcome simple protective measures.

On the other hand, we came across more recent attacks among which several variants of the Mirai botnet. Albeit more recent, those botnets also rely on weak or nonexistent security measures on the part of their targets.

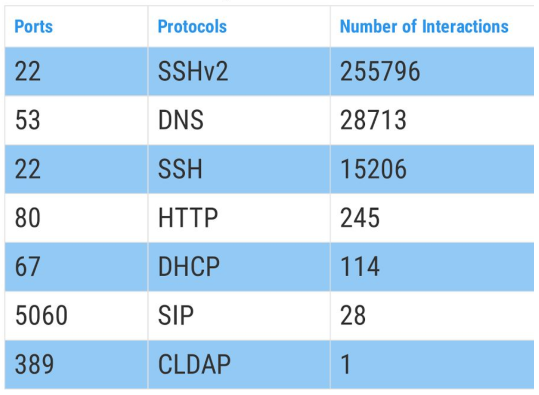
While this paper reaches its conclusion, our work is not done and we will keep developing our honeypots so as to respond more dynamically to malicious connection attempts.

Acknowledgments

This work would not have been possible without Sebastien Tricaud, the founder of the French Honeynet chapter, to whom we are especially indebted. He provided us guidance and leadership every step of the way, allowing us to succeed as a team.
A co-founder of Holberton School, Julien Barbier's help was invaluable. Not only did he introduce us to Sebastien, but he also gave us the opportunity, as students, to discover more about the field of information security.
We are grateful to Richard Sim and Philip Yoo, both students at Holberton School, for proofreading this paper and contributing to making it better.

Annex

Table 1.1: Overview of the Layer 7 Traffic Captured (24-hour capture)



Ports	Protocols	Number of Interactions
22	SSHv2	255796
53	DNS	28713
22	SSH	15206
80	HTTP	245
67	DHCP	114
5060	SIP	28
389	CLDAP	1

References

1. Lillian Ablon, Martin C. Libicki, Andrea A. Golay. 2017. *Golay Markets for Cybercrime Tools and Stolen Data*. RAND. pp. 21-23. [ONLINE] Available at: https://www.rand.org/content/dam/rand/pubs/research_reports/RR600/RR610/RAND_RR610.pdf. [Accessed 2 May 2017].
2. Lily Hay Newman. 2017. *What We Know About Friday's Massive East Coast Internet Outage*. Wired. [ONLINE] Available at: https://www.wired.com/2016/10/internet-outage-ddos-dns-dyn/. [Accessed 2 May 2017].
3. Lillian Ablon, Martin C. Libicki, Andrea A. Golay. 2017. *Golay Markets for Cybercrime Tools and Stolen Data*. RAND. pp. 21-23. [ONLINE] Available at: https://www.rand.org/content/dam/rand/pubs/research_reports/RR600/RR610/RAND_RR610.pdf. [Accessed 2 May 2017].
4. CIRCL. 2017. *BGP Ranking*. [ONLINE] Available at: https://www.circl.lu/projects/bgpranking. [Accessed 13 May 2017].
5. CIRCL. 2016. *Bgpranking-redis-api*. [ONLINE] Available at: https://github.com/CIRCL/bgpranking-redis-api. [Accessed 13 May 2017].
6. William W. Martin. 2001. *Honey Pots and Honey Nets - Security through Deception*. SANS Institute. [ONLINE] Available at: https://www.sans.org/reading-room/whitepapers/attacking/honey-pots-honey-nets-security-deception-41. [Accessed 2 May 2017].
7. IETF. 1999. *RFC 2616 - Hypertext Transfer Protocol*. [ONLINE] Available at: https://tools.ietf.org/html/rfc2616. [Accessed 13 May 2017].
8. Ian Liu-Johnston. 2017. *nodeJS_honeypot*. [ONLINE] Available at: https://github.com/ianliu-johnston/nodeJS_honeypot. [Accessed 13 May 2017].
9. CVE. 2014. *CVE-2014-6271*. [ONLINE] Available at: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271. [Accessed 2 May 2017].
10. CVE. 2017. *CVE-2014-6271*. [ONLINE] Available at: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5638. [Accessed 2 May 2017].
11. Exploit Database. 2017. *Apache Struts 2.3.5 < 2.3.31 / 2.5 < 2.5.10 - Remote Code Execution*. [ONLINE] Available at: https://www.exploit-db.com/exploits/41570/. [Accessed 13 May 2017].
12. Terrence DeJesus. 2017. *Apache Struts 2 Exploit Analysis.* NTT Security. [ONLINE] Available at: https://www.solutionary.com/resource-center/blog/2017/03/apache-struts-2-exploit-analysis/. [Accessed 18 May 2017].
13. CXSecurity.com. 2017. *Apache Struts Jakarta Multipart Parser OGNL Injection - CXSecurity.com*. [ONLINE] Available at: https://cxsecurity.com/issue/WLB-2017030143. [Accessed 18 May 2017].
14. Pooler. 2017. *cpuminer: CPU miner for Litecoin and Bitcoin*. [ONLINE]

Available at: https://github.com/pooler/cpuminer. [Accessed 18 May 2017].

15. IETF. 1983. *RFC 854 - Telnet Protocol Specification*. [ONLINE] Available at: https://tools.ietf.org/html/rfc854. [Accessed 13 May 2017].
16. Tim Britton and Holden Grissett. 2017. *telnet-honeypot*. [ONLINE] Available at: https://github.com/h-m-s/telnet-honeypot. [Accessed 13 May 2017].
17. Pierre Kim. 2017. *Multiple vulnerabilities found in Wireless IP Camera (P2P) WIFICAM cameras and vulnerabilities in custom http server*. [ONLINE] Available at: https://pierrekim.github.io/blog/2017-03-08-camera-goahead-0day.html. [Accessed 2 May 2017].
18. John Costello , Allison Nixon , Brian Hein , Ronnie Tokazowski , Zach Wikholm . 2016. *New Mirai Variant Leaves 5 Million Devices Worldwide Vulnerable — High Concentration in Germany, UK and Brazil*. Flashpoint. [ONLINE] Available at: https://www.flashpoint-intel.com/blog/cybercrime/new-mirai-variant-involved-latest-deutsche-telekom-outage/. [Accessed 2 May 2017].
19. Roland Dobbins. 2016. *Mirai IoT Botnet Description and DDoS Attack Mitigation. ARBOR Networks*. [ONLINE] Available at: https://www.arbornetworks.com/blog/asert/mirai-iot-botnet-description-ddos-attack-mitigation/. [Accessed 2 May 2017].
20. IETF. 2002. *RFC 3261 - Session Initiation Protocol*. [ONLINE] Available at: https://tools.ietf.org/html/rfc3261. [Accessed 13 May 2017].
21. Cisco. 2014. *SIPVicious SIP Auditing Tool Activity*. [ONLINE] Available at: https://tools.cisco.com/security/center/viewAlert.x?alertId=33141. [Accessed 2 May 2017].
22. Kali Linux. 2014. *SIPVicious Package Description*. [ONLINE] Available at: http://tools.kali.org/sniffingspoofing/sipvicious. [Accessed 2 May 2017].
23. IETF. 2006. *RFC 4253 - The Secure Shell (SSH) Transport Layer Protocol*. [ONLINE] Available at: https://tools.ietf.org/html/rfc4253. [Accessed 13 May 2017].
24. Sam Edwards, Ioannis Profetis. 2016. *Hajime: Analysis of a decentralized internet worm for IoT devices*. [ONLINE] Available at: https://security.rapiditynetworks.com/publications/2016-10-16/hajime.pdf. [Accessed 13 May 2017].
25. KrebsOnSecurity. 2016. *KrebsOnSecurity Hit With Record DDoS*. [ONLINE] Available at: https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-witth-record-ddos/. [Accessed 13 May 2017].
26. Dyn. 2016. *Dyn Statement on 10/21/2016 DDoS Attack*. [ONLINE] Available at: https://dyn.com/blog/dyn-statement-on-10212016-ddos-attack/. [Accessed 13 May 2017].
27. Octave Klaba. 2016. *we got 2 huge multi DDoS: 1156Gbps then 901Gbps*. [ONLINE] Available at: https://twitter.com/olesovhcom/status/778019962036314112. [Accessed 13 May 2017].
28. KrebsOnSecurity. 2017. *Who is Anna-Senpai, the Mirai Worm Author?*. [ONLINE] Available at:

https://krebsonsecurity.com/2017/01/who-is-anna-senpai-the-mirai-worm-author/. [Accessed 13 May 2017].

29. India Ashok. 2016. *One million IoT devices infected by Bashlite malware-driven DDoS botnet*. International Business Times. [ONLINE] Available at: https://www.ibtimes.co.uk/one-million-iot-devices-infected-by-bashlite-malware-driven-ddos-botnet-1578870. [Accessed 13 May 2017].
30. geniosa. 2016. *qbot*. [ONLINE] Available at: https://github.com/geniosa/qbot/blob/master/cc7.py.txt. [Accessed 13 May 2017].
31. Radware. 2017. *"BrickerBot" Results In PDoS Attack*. [ONLINE] Available at: https://github.com/geniosa/qbot/blob/master/cc7.py.txt. [Accessed 13 May 2017].
32. Catalin Cimpanu. 2017. *BrickerBot Author Claims He Bricked Two Million Devices. BleepingComputer*. [ONLINE] Available at: https://www.bleepingcomputer.com/news/security/brickerbot-author-claims-he-bricked-two-million-devices/. [Accessed 13 May 2017].