Analyzing Variation Among IoT Botnets Using Medium Interaction Honeypots

Bryson Lingenfelter
Dept. of Computer Science and Eng.
University of Nevada, Reno
Reno, USA
blingenfelter@nevada.unr.edu

Iman Vakilinia
School of Computing
University of North Florida
Jacksonville, USA
i.vakilinia@unf.edu

Shamik Sengupta
Dept. of Computer Science and Eng.
University of Nevada, Reno
Reno, USA
ssengupta@unr.edu

***Abstract*—Through analysis of sessions in which files were created and downloaded on three Cowrie SSH/Telnet honeypots, we find that IoT botnets are by far the most common source of malware on connected systems with weak credentials. We detail our honeypot configuration and describe a simple method for listing near-identical malicious login sessions using edit distance. A large number of IoT botnets attack our honeypots, but the malicious sessions which download botnet software to the honeypot are almost all nearly identical to one of two common attack patterns. It is apparent that the Mirai worm is still the dominant botnet software, but has been expanded and modified by other hackers. We also find that the same loader devices deploy several different botnet malware strains to the honeypot over the course of a 40 day period, suggesting multiple botnet deployments from the same source. We conclude that Mirai continues to be adapted but can be effectively tracked using medium interaction honeypots such as Cowrie.**

***Index Terms*—Botnet, Honeypot, Internet of Things, Mirai, Cowrie**

I. Introduction

Shortly after the Mirai malware made headlines in 2016 due to its usage in a 620 Gbps attack against *krebsonsecurity.com* [1], the botnet code was publicly released as open source software. The original version of Mirai targeted vulnerable Internet of Things (IoT) devices using a short wordlist based on default username and password combinations, amassing an army of hundreds of thousands of connected devices using just these credentials. The release of Mirai's source code has resulted in numerous clones and modifications by other malicious actors which have expanded upon the attacks present in the original version of Mirai. Despite receiving much attention after the 2016 attacks, IoT security is still a major issue and new botnets appear regularly. As a result, there is a need to keep track of developments in IoT botnets so current attacks can be appropriately dealt with.

Honeypots are a popular technique for capturing malicious activity. They provide a sandbox environment for malicious actors to attack, recording each action for later analysis. Honeypots have been used for a wide range of tasks, such as attack pattern comparison, root cause identification, risk assessment, attack frequency analysis, and attack origins analysis [2]. Medium interaction honeypots such as Cowrie [3] provide a fake shell and virtual file system to the attacker, such that the attacker can enter commands and see believable output without having access to a real system [4]. This is an ideal environment for tracking the state of IoT botnets, which are easy to capture samples from because they are automated and not able to perform honeypot detection as thoroughly as a real attacker. We collect data using the open source Cowrie honeypot, which is a continuation of the Kippo honeypot. Kippo/Cowrie honeypots have previously been used to cluster login sessions based on session time and attacker skill [5], and visualize attacker location and common commands [6].

In this paper we describe a honeypot configuration based on the Cowrie SSH/Telnet honeypot for capturing remote login sessions and downloaded files, expanding from our previous work analyzing password attempts [7]. For our analysis, we look at sessions which created and downloaded files. Login session analysis has previously been done using high interaction honeypots and classifying each command as part of a specific state [8]. We instead describe a simple method for clustering these sessions using edit distance to enumerate identical attack patterns. In light of these sessions almost universally being associated with the Mirai malware, we provide discussion of Mirai's functionality. Previous work provides complete description of the original malware's behavior, architecture, and attack methods [9], [10]; we instead focus on how much Mirai has been modified. There is some existing work in this area. Y. Liu and H. Wang [11] use branch name, configuration, IP addresses, attack methods, and credential dictionaries to distinguish between binary files from different Mirai variants. Other work describing Mirai has identified versions which target different ports and devices [10].

Using our clustering results, we note how many different Mirai variants attack our honeypots and how much they differ from one another and the publicly available Mirai source code in terms of commands entered, choice of filler words, and downloaded files. We aim to provide an idea of the amount of variation present in Mirai attacks both in terms of number of distinct variants and variation amount. We also provide analysis of variation in files from apparently identical malware

This research is supported by the National Science Foundation (NSF), USA, Award #1739032.

978-1-7281-3783-4/20/$31.00 ©2020 IEEE

loaders as well as identical IP addresses and find that many IP addresses serve as loader servers for multiple strains of Mirai-based botnets. Finally, we provide information on other attacks observed by the honeypot to give some perspective regarding the prevalence of Mirai.

II. Methodology

A. Honeypot Configuration

Malware data was collected from 3 honeypots running the Cowrie SSH/Telnet honeypot [3], the continuation of the popular Kippo honeypot. The Cowrie honeypot is a medium-interaction honeypot which provides a dummy shell implemented in python to the attacker with a virtual filesystem and prefab command outputs. Cowrie can be used to emulate an IoT system by setting the prefab outputs to be consistent with actual IoT devices, and is therefore a good environment for observing IoT botnets without the overhead of a high-interaction honeypot. Cowrie has also been modified multiple times during its development specifically in response to issues where sessions from the Mirai malware didn't result in a file download, making it particularly good for capturing samples of Mirai.

We made minimal alterations to Cowrie's default configuration as of February 2019. Cowrie's userdb was modified slightly to allow login for the three default cowrie users (root, tomcat, and oracle) given any password. Additionally, the honeypot was configured to accept both ssh and telnet connections on ports 22/2222 and 23/2223, respectively. This was done to allow as much traffic as possible. Each honeypot had an Apache webserver running on port 80 associated with a registered domain name and static web page. Finally, each honeypot had a Postfix mailserver running on port 25 with several valid email accounts set up. Each portion of the honeypot (login, web, and mail) ran on a separate virtual machine and received traffic through port forwarding. For the analysis in this paper, only the Cowrie component is necessary.

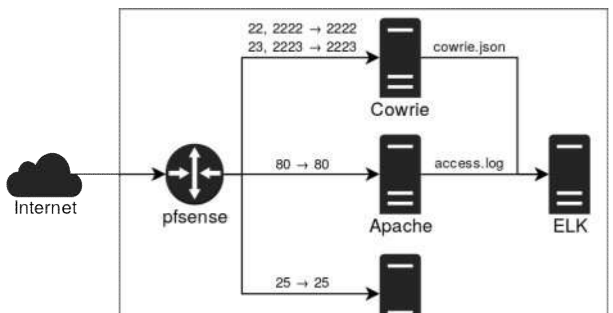

Fig. 1. Configuration of the honeypot used for data collection. Three honeypots were deployed in this manner.

To aggregate logs from the web and ssh servers, we used Filebeat to ship logs to a server running the ELK stack, composed of Elasticsearch, Logstash, and Kibana. Logstash is used to transform the JSON data generated by Cowrie and the Combined Log Format data generated by Apache into the format used by the Elasticsearch search and analytics engine. We make an ssh server available on the Postfix machine by forwarding port 22222 on the pfsense router to port 22 on the mail server. We also make the ELK machine accessible through ssh over port 22 from the internal network, allowing access to Elasticsearch and Kibana via an ssh tunnel through the mail server. This allows us to remotely access ELK without needing to expose the server publicly.

B. Dataset

Every event Cowrie adds to its log has a structure defined by the event type, such as `cowrie.login.success` and `cowrie.command.input`. Each event, regardless of type, has a session id field that can be used to identify events generated by the same session. To collect the data for this paper, we enumerated every unique session id associated with a `cowrie.session.file_download` event and used Elasticsearch to collect all other events for these ids. The file download event is used for both downloads from remote servers and files created during the login session, so we consider both for our analysis. The data was collected over a period of 40 days, from 4 February 2019 to 15 March 2019, and consists of 84,602 unique login sessions. These sessions account for 21.2% of the 398,233 session dataset; that is, roughly 20% of the time a connection was initiated with a honeypot it resulted in a successful login followed by a file creation or download. These login sessions resulted in a total of 312 unique files collected by the honeypot.

C. Attack Pattern Comparison

To identify identical attack patterns, each session was encoded as a list of the first argument for each command entered by the attacker. For example, if an attacker logged in, typed `cat /proc/mounts` followed by `echo test`, then ended the session, the encoded list would be `["cat", "echo"]`. A command containing `;` (which most shells treat as end of line), but not commands with `&&` or `||`, was split into multiple commands. If the first argument was a shell such as `/bin/busybox` or `/bin/bash` we instead use the second argument, which is the first argument to the shell being called. To identify identical attack patterns, Levenshtein distance between lists of arguments was used as a similarity metric. Levenshtein distance computes the number of edits (substitution, insertion, and deletion) required to match two sequences of non-equal length, providing a rough estimation of similarity between login sessions. Edit distance has previously been used for applications such as root cause analysis [12]. The metric was normalized by dividing by its upper bound, the largest number of arguments in either session. This metric does not consider state or different but functionally identical commands, but is sufficient for identifying connections from bots using the same code. Based on the results shown in Figure 2, we consider attack patterns identical if their normalized distance is less than 0.25

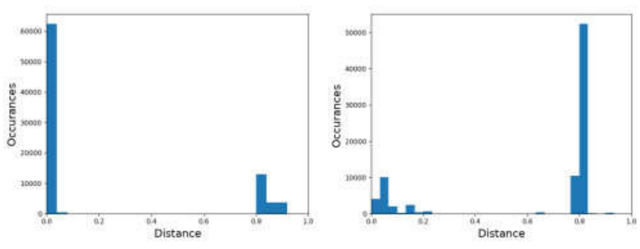

Fig. 2. Histogram of distance measures for the two most common attack patterns. For both patterns, there is a clear distinction between distances less than .25 and distances greater than .25.

III. Session Analysis

The two most common attack patterns seen on the honeypot are closely related to publicly available Mirai loader code. These two attack patterns account for 97.7% of the dataset, as well as 244 out of the 312 unique files. 236 of these files were identified as strains of the Mirai IoT botnet malware by at least one antivirus software used by the VirusTotal antivirus report aggregator [13]. Of the remaining eight, six were identified by at least one source as Gafgyt, another earlier IoT botnet malware with behavior very similar to Mirai [14] [15], and the other two were not flagged as malware by any antivirus at the time we submitted them for analysis. The two attacks are entirely accounted for by 125 unique IPs, of which 16 are common to both attacks.

To better understand the observed botnet loader sessions, we provide a brief description of Mirai's spreading functionality. The Mirai botnet grows through scanning of randomly generated IPs by infected devices. When an infected device detects another vulnerable device, it sends the credentials it used to access that vulnerable device to a report server. After a certain period of time, a loader device then connects to the compromised device using the reported credentials and downloads and executes the Mirai malware to add the device to the botnet. A visual depiction of this behavior is provided in Figure 3.

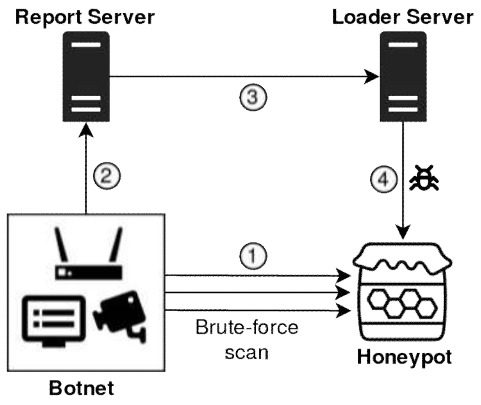

Fig. 3. Diagram showing how scans from a Mirai botnet result in attempted infection by a loader device.

The second most common attack pattern, which accounts for 23.4% of the observed sessions, appears to directly use the loader server code from the public release of the Mirai source code [16]. The loader attempts to find a writable directory by repeatedly redirecting output from `echo` to a file in one of several different directories, attempting to read the file using `cat` to see if this succeeded, then removing the file with `rm`. After finding a writable directory in this manner, determining the system's cpu architecture by reading the header for the `echo` system binary using `cat`, and checking for `wget` and `tftp`, the loader downloads a file using `wget` then attempts to execute it. In the publicly released Mirai loader, the downloaded file is a variant of Mirai for the compromised system's cpu architecture. We note that all malware collected by the honeypot was compiled for x86 systems, which is consistent with the x86_64 architecture displayed by Cowrie.

The most common attack pattern follows the same logic as the Mirai source code, but does not use the same commands. Instead of using `echo` to save raw hex to a file, this pattern uses > to create empty files in several directories then attempts to `cd` to those directories if the redirection was successful. Unlike in the original Mirai loader, the attacker in this case does not remove any of the created files. Additionally, this pattern uses `read` to obtain a binary file header if `cat` fails. However, after a writable directory is found and the cpu architecture is determined the loaders are identical. This pattern is far more common than the previous pattern, and accounts for 74.3% of the observed sessions.

Most of the differences internal to each attack pattern are related to the first commands entered by the loader. We define these commands as everything coming before the first > for the most common pattern and everything coming before the first `echo` for the second most common pattern. All loaders generally begin with `enable`, `shell`, and `sh`, in which the attacker attempts to gain access to privileged-mode commands and run a Bourne shell [17]. Minor variations on these commands show up in different loaders, as shown in Table 1. It appears that different attackers use the same loader structure, but make minor modifications to the first commands run by the loader.

TABLE I
INITIAL COMMANDS FOR MOST COMMON ATTACK PATTERN

Occurrences	Command Sequence
10,339	enable, system, linuxshell, shell, sh
92	enable, system, shell, ping, sh, sh
51,867	enable, system, shell, sh
147	enable, system, shell, sh, linuxshell
305	shell, sh
117	system, shell, sh

Every loader following the second pattern then checks the response to an arbitrarily chosen filler keyword which is titled `TOKEN_QUERY` in the Mirai source code [16]. The expected response to this query, titled `TOKEN_RESPONSE` in the Mirai source code, is `TOKEN_QUERY: applet not found`. This is the response generated by the IoT devices targeted by Mirai, which use the Busybox shell. We observed a total of

45 unique choices for this variable, 34 of which showed up in the most common attack pattern and 12 of which showed up in the second attack pattern (1 was used for both). After receiving TOKEN_RESPONSE from the victim device, loaders following the second attack pattern almost always then use `ps` and `cat /proc/mounts` to further fingerprint the system. The first pattern always forgoes these checks.

The IP addresses observed in the most common two attack patterns follow an exponential distribution. In both patterns, the majority of IP addresses were seen under 100 times, but the most common IP addresses were seen upwards of 10,000 times. This is shown in Figure 4, which plots the IP distribution on a logarithmic scale. Although there are a large number of active botnets, a small portion of them account for the majority of malicious file uploads on the honeypot.

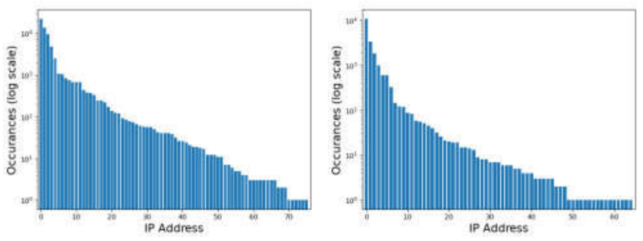

Fig. 4. Frequency of IP addresses for most common two attack patterns, plotted on a logarithmic scale.

These two patterns do not account for every strain of Mirai loader seen on the honeypot. A far less common attack which accounts for just .415% of the dataset (315 sessions from 1 IP), is near identical to the original Mirai source code with substantial modifications to the first few commands. Additionally, Mirai sessions which did not use `echo` or `>` to create files and did not result in an attempted file download are not included in the dataset.

IV. File Analysis

Manual inspection of the binary files downloaded by the two forms of Mirai loader demonstrates a large amount of variation in binaries even for identical loaders. At least 11.9% of the files are packed, with that percentage of files containing "packed with the UPX executable packer," the most popular binary packer among malware authors [18]. Some malware contains specific identifiers, such as "Botnet Made By greek.Helios," "Botnet Made By R2F," and "Edit by Zer0x." 22.1% of the collected files contain the string "/ctrlt/DeviceUpgrade_1," indicative of an attack targeting CVE-2017-17215 [19]. Additionally, 13.1% of the files were compiled for 64 bit systems and 86.9% were compiled for 32 bit systems. To categorize the malware, we use the choice of TOKEN_QUERY, which is apparent in both loaders. In addition to appearing at the beginning of the session for the second attack pattern, for both attacks TOKEN_QUERY is placed after some commands so that successful completion of the command will be followed by TOKEN_RESPONSE. Mirai variants commonly, though not always, use different filler words in their loader. Figure 5 shows the distance matrix for TOKEN_QUERY for the second most common attack pattern, from which it can be seen that similar choices of TOKEN_QUERY in the loader correspond to near-identical loaded malware. Lines separate different choices of TOKEN_QUERY, and shaded cells indicate similarity between files. Distances between malware samples are calculated using ssdeep, a context-triggered piecewise hashing algorithm [20].

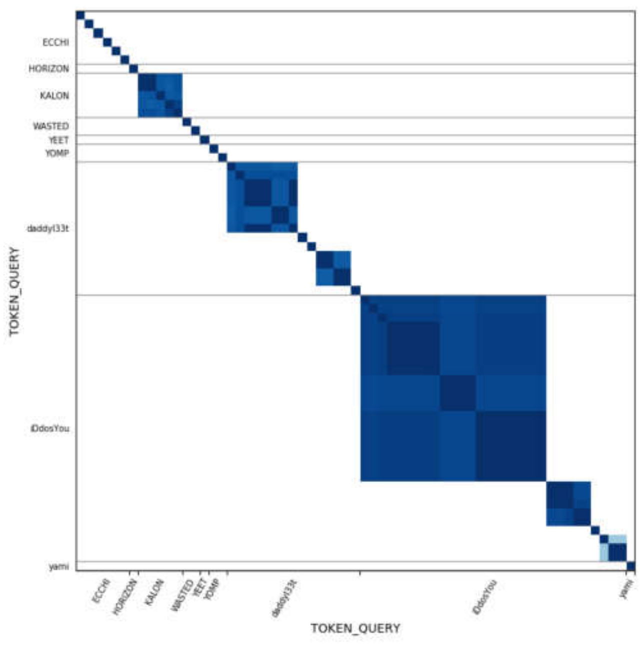

Fig. 5. Distance matrix of ssdeep hashes for Mirai samples from second attack pattern, grouped by choice of TOKEN_QUERY. Darker blues indicate a higher similarity score.

Ssdeep is not robust for detecting functionally identical files which don't have identical bytes or modifications which alter the context triggering [21]. For example, we found through static analysis that the "iDosYou" strain is near identical to the "daddyl33t" strain with a few additional subroutines in the middle of the binary, but ssdeep indicates these files have 0% similarity. However, ssdeep has very good precision [21] and malware considered highly similar are therefore likely to represent very minor modifications of the same code. This can be seen in Figure 5 in that files with different values for TOKEN_QUERY are never detected as similar by ssdeep. For the filler words accounting for the most variance in file hashes, however most of the unique files observed are near identical. For most of the files collected from the "iDosYou" loader, which accounted for the most unique files in the second most common pattern, the only apparent difference is a single IP which varies from sample to sample. This suggests that this same malware is associated with many different loader servers.

To better understand the scope of attacks against the honeypot, each session was reduced to a mapping between IP and the hash of the file downloaded during the session. This analysis was done separately for each attack pattern. The data does

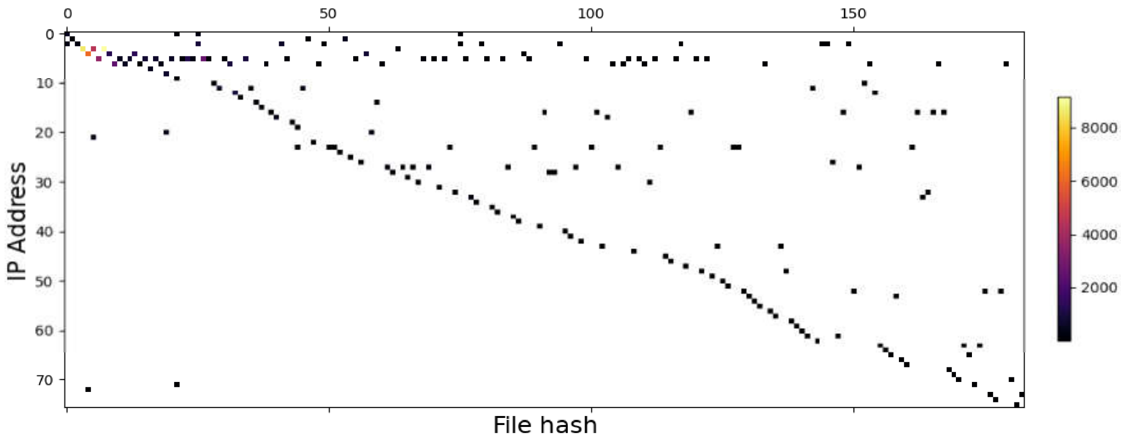

Fig. 6. Mapping between IP addresses and file hashes for the most common attack pattern. White indicates the IP and file never appeared together, otherwise the color ranges from dark purple (IP and hash appeared together few times) to yellow (IP and hash appeared together many times).

not display a one-to-one mapping between IP address and file hash; some IPs download different files in other sessions, and some files are downloaded by multiple IPs. Interestingly, for the most common attack pattern the cardinality of the file set is larger than the cardinality of the IP set. This can be seen in Figure 6, which provides a weighted mapping between IP addresses and files.

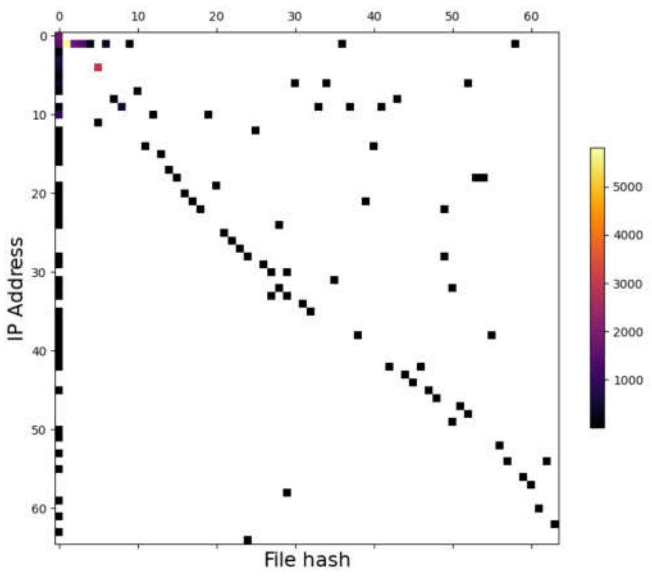

Fig. 7. Mapping between IP addresses and file hashes for the second most common attack pattern. The first column is an empty file hash (no file was downloaded).

It appears common for attackers to change the botnet connected to a loader device, resulting in the same IP attacking the honeypot with several different strains of the Mirai malware. It is far more rare, in contrast, for different IP addresses to download the same malware to the honeypot. While many unique files are presumably minor updates to existing files, it

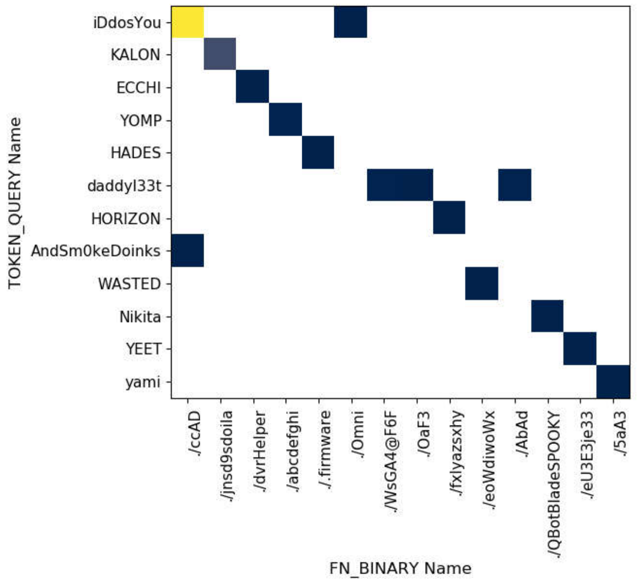

Fig. 8. Mapping between TOKEN_QUERY and FN_BINARY for the second pattern.

is clear that many IPs are downloading highly distinct variants of Mirai. One IP address, for example, downloaded five unique files to the honeypot. Three of the five were signed in ASCII by one of two authors. One was packed with UPX, and two contain strings indicating an attack attempting to exploit CVE-2017-17215. Another IP downloaded 29 unique files, demonstrating similar variation in packing and attack methods. We surmise that attackers deploy different strains over time to expand their botnets further, or different hackers take over the same loader device. We again note that 16 of the 125 unique IPs associated with the two most common attacks launched both the most common attack pattern and the second most common attack pattern, indicating that the same IP addresses

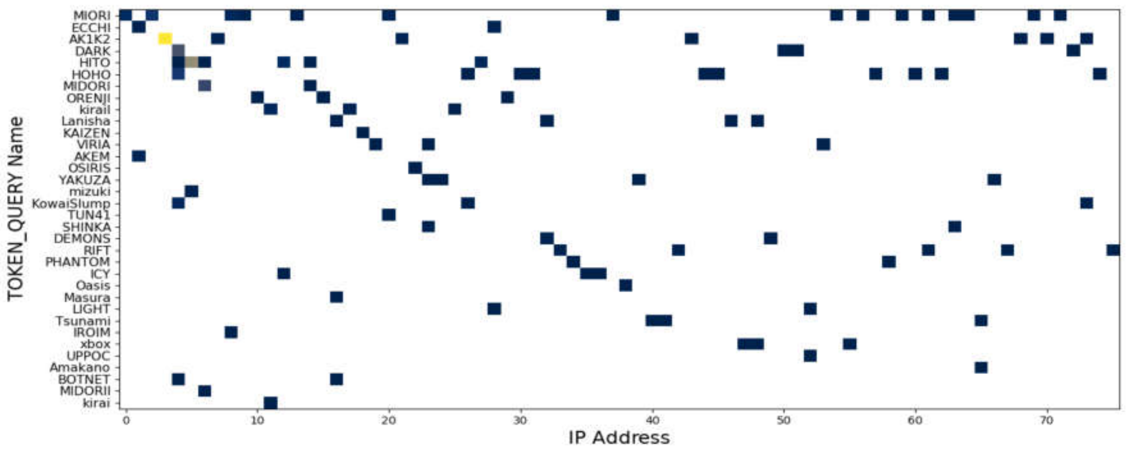

Fig. 9. Mapping between IP address and TOKEN_QUERY variable for the most common attack.

are used for different botnet architectures entirely.

An entirely different relationship between file hash and IP address is apparent in the second most common attack pattern. Unlike the first pattern, in which every session results in a file being downloaded to the system, file downloads frequently fail to be captured in the second attack. This can be seen in the mapping between IP address and file hash, shown in Figure 7. The majority of IP addresses show up in the first column, representing an empty hash (meaning no files were downloaded). Additionally, the number of unique files is no longer greater than the number of unique IPs for this attack.

Another observation from the two Mirai loader attacks is the mapping between `TOKEN_QUERY` and the name of the downloaded binary (`FN_BINARY`), shown in Figure 8. In the leaked Mirai source code, these two variables are defined as "ECCHI" and "dvrHelper" (we note that this is different from the samples of Mirai collected before its release, where "MIRAI" was used rather than "ECCHI" [14]). Mirai variants are sometimes named after their choice for `TOKEN_QUERY`, as we have done in this paper. While the mapping between `TOKEN_QUERY` and `FN_BINARY` is fairly diagonal, there are a few cases where two `FN_BINARY` variables map to the same `TOKEN_QUERY` and vice versa. This seems to indicate that malware is further modified by other attackers, resulting in variants of variants. We also suggest some malware authors chose to use the same `TOKEN_QUERY` variable as existing strains.

The `TOKEN_QUERY` variable can be further used to show that the same IP addresses are launching a wide range of attacks. As shown in Figure 9, there is not a clear relationship between IP address and the choice of this variable. Unlike the mapping between IP address and file hash, which is mostly diagonal, the mapping between IP address and `TOKEN_QUERY` is scattered. Most filler words are associated with a large range of loader server IPs and many IPs are associated with two or more loader strains. This is consistent with our observation from mapping between IP address and file hash that the same IP addresses are launching a variety of different attacks.

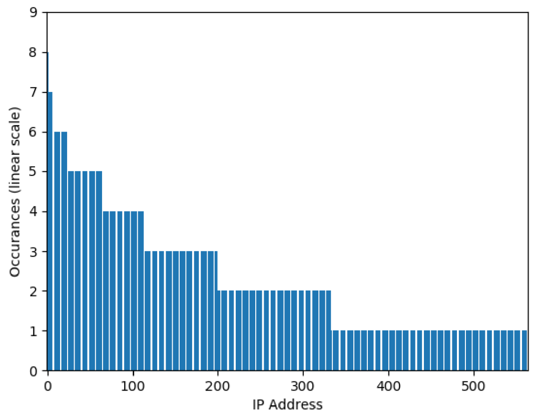

Fig. 10. Frequency of IP addresses for the ssh brute-force attack, plotted on a linear scale.

V. Other Attacks

By removing common Mirai attacks from our dataset, we can gain a better understanding of attacks against the honeypot which are not related to IoT botnets. Accounting for the common three Mirai patterns leaves 1.879% of file-downloading sessions unaccounted for. Most of these are an ssh-based attack which accounted for 1.55% of the unique sessions. This attack was short and simple; an attacker logged in, tried several methods of keeping the server from tracking their bash history, uploaded an ssh public key, then logged out. Due to the sandbox nature of the Cowrie honeypot, the ssh key would be saved but unusable for remote login. Despite only accounting for under 2% of the dataset, this attack had by far the largest number of unique IPs. Of the 1,312 unique sessions, 564 were from unique IPs. This is more than four times the number of IPs observed by the most common two attack patterns, despite those patterns accounting for almost the entire dataset.

The majority of the remaining sessions (.171% of the total dataset; 145 sessions from 9 IPs) appear to be a simplified version of the Mirai loader. The attacker uses the same few initial test commands as the Mirai loader, but instead of performing any further checks using `echo`, `rm`, and output redirection, the attacker simply downloads a bash script that attempts to download binaries compiled for different architectures until one works. The remaining 133 sessions in the dataset vary widely in attack type. In two different attack patterns, accounting for 19 sessions, the attacker logs in using ssh and downloads a perl script which tries to add the honeypot to an IRC-based botnet. In one attack pattern, accounting for 2 sessions, an attacker logs in using ssh and downloads a coin-mining malware. In a few particularly strange sessions, the attacker spends three minutes repeatedly downloading and attempting to run a shell script similar to the one downloaded by the simplified mirai loader.

VI. Conclusion

Mirai remains a major security issue, with botnet attacks on vulnerable telnet ports being by far the most common attack to download or create files. We also observe ssh attacks and older, IRC-based botnet attacks, but these are by far in the minority. We show that the Cowrie honeypot is an effective system for collecting samples of Mirai and Mirai sessions, and identical loader sessions can be found by using simple edit distance between command sequences. From the data collected by our honeypots, we find that a relatively small number of loader devices running an even smaller number of loader code bases are responsible for a wide range of botnet attacks based on Mirai. We conclude that Mirai is directly or indirectly responsible for a vast number of attacks on IoT devices, but can be easily tracked using medium interaction honeypots and sequence matching.

For future work, analysis could be expanded by faking different operating systems using Cowrie to collect a wider range of files. Additionally, more in-depth analysis of the collected files would be needed to gain a better understanding of how much downloaded files vary between strains. While ssdeep shows which files are nearly identical, it is not robust enough for complete analysis and a more flexible solution is necessary. A longer time frame and consideration of sessions which don't result in file downloading would also assist analysis.

References

[1] B. Krebs. (2016, September) Krebsonsecurity hit with record ddos. [Online]. Available: https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/

[2] M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, and J. Schönfelder, "A survey on honeypot software and data analysis," *CoRR*, vol. abs/1608.06249, 2016. [Online]. Available: http://arxiv.org/abs/1608.06249

[3] M. Oosterhof. Cowrie. [Online]. Available: github.com/cowrie/cowrie

[4] G. Wicherski, "Medium interaction honeypots," in *German Honeynet Project*, 2006.

[5] D. Fraunholz, D. Krohmer, S. D. Anton, and H. Dieter Schotten, "Investigation of cyber crime conducted by abusing weak or default passwords with a medium interaction honeypot," in *2017 International Conference on Cyber Security And Protection Of Digital Services (Cyber Security)*, June 2017, pp. 1–7.

[6] I. Koniaris, G. Papadimitriou, and P. Nicopolitidis, "Analysis and visualization of ssh attacks using honeypots," in *Eurocon 2013*, July 2013, pp. 65–72.

[7] I. Vakilinia, S. Cheung, and S. Sengupta, "Sharing susceptible passwords as cyber threat intelligence feed," in *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*. IEEE, 2018, pp. 1–6.

[8] D. Ramsbrock, R. Berthier, and M. Cukier, "Profiling attacker behavior following ssh compromises," in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, June 2007, pp. 119–124.

[9] 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, and Y. Zhou, "Understanding the mirai botnet," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 1093–1110. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

[10] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "Ddos in the iot: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[11] Y. Liu and H. Wang, "Tracking mirai variants." Virus Bulletin, 2018. [Online]. Available: https://www.virusbulletin.com/uploads/pdf/magazine/2018/VB2018-Liu-Wang.pdf

[12] F. Pouget and M. Dacier, "Honeypot-based forensics," in *AusCERT Asia Pacific Information technology Security Conference 2004*, Brisbane, Australia, 2004. [Online]. Available: http://www.eurecom.fr/publication/1417

[13] Virustotal. [Online]. Available: virustotal.com

[14] MalwareMustDie. (2016, August) Mmd-0056-2016 - linux/mirai, how an old elf malcode is recycled.. [Online]. Available: http://blog.malwaremustdie.org/2016/08/mmd-0056-2016-linuxmirai-just.html

[15] G. Kambourakis, C. Kolias, and A. Stavrou, "The mirai botnet and the iot zombie armies," in *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*, Oct 2017, pp. 267–272.

[16] Mirai-source-code. [Online]. Available: https://github.com/jgamblin/Mirai-Source-Code/

[17] S. Edwards and I. Profetis, "Hajime: Analysis of a decentralized internet worm for iot devices," *Rapidity Networks*, October 2016. [Online]. Available: https://security.rapiditynetworks.com/publications/2016-10-16/hajime.pdf

[18] K. A. Roundy and B. P. Miller, "Binary-code obfuscations in prevalent packer tools," *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, p. 4, 2013.

[19] Huawei router hg532 - arbitrary command execution. [Online]. Available: https://www.exploit-db.com/exploits/43414

[20] J. Kornblum, "Identifying almost identical files using context triggered piecewise hashing," *Digital Investigation*, vol. 3, pp. 91–97, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2006.06.015

[21] Y. Li, S. C. Sundaramurthy, A. G. Bardas, X. Ou, D. Caragea, X. Hu, and J. Jang, "Experimental study of fuzzy hashing in malware clustering analysis," in *8th Workshop on Cyber Security Experimentation and Test (CSET 15)*. Washington, D.C.: USENIX Association, 2015. [Online]. Available: https://www.usenix.org/conference/cset15/workshop-program/presentation/li