Mirai Botnet Detection: A Study in Internet Multi-resolution Analysis for Detecting Botnet Behavior

Sarah Khoja, Antonina Serdyukova, Khadeza Begum, Joonsang Choi

May 14, 2017

Project Summary

Botnets are by no means a recent attack vector, but, as Mirai's recent attack on Dyn showed, they still command attention. While there is a significant amount of research on botnet detection, behavior and prevention, there is still room for new botnets to create chaos. Mirai is one such botnet which targeted the Internet of Things (IoT) devices. It was able to exploit some major vulnerabilities such as default passwords and unprotected ports. It may seem surprising that Mirai's trick card worked, especially considering the Carna botnet [CARNA] which displayed almost the same exact vulnerabilities on networked devices. However, this simply displays the lack of security knowledge, or the obliviousness when it comes to IoT's. In this contribution, we simulate the botnet in order to determine any identifying characteristics or behaviors, which will allow for a situational awareness report.

Motivation for the Project

A botnet is a collection of infected devices, or network of robots as its name suggests. Generally, vulnerable targets are located, and the malware is loaded onto the machine which then joins the botnet. The machine then sends pings back to the command and control center, or CNC, which is controlled by the botmaster, or the main attacker. The CNC is used to send attacks and other commands to all the bots, and maintain the botnet. The attacker will gather a large net of devices, and then use the combined force of all the bots to send torrents of packets to a specified target. This onslaught

usually clogs the links to the point where legitimate service is denied, because the target cannot handle the requests, thus resulting in what is known as a denial of service attack. The target systems usually end up failing, and will suffer continual denial of service until the attackers are stopped or back down. There are of course variations of botnet architectures, setups and types of attacks and obfuscations.

Mirai was not an overly complicated botnet, rather it leveraged the large number of vulnerable, and easily exploited IoT's to garner enough strength to carry out a large scale distributed denial of service attack (DDoS). Initially Mirai attacked a security journalist's website, and then moved on to a larger target- Dyn, a popular internet provider that supports a large part of the internet backbone newmanwired. The traffic from the IoT bots effectively denied millions of people access to internet services. While the attacks were mitigated quickly, it did not deter from the fact that this botnet caused an unsettling ripple in the increasingly connected world we now occupy. This is more than enough to motivate research efforts in order to efficiently and effectively detect and prevent further DDoS attacks.

Though the technological age introduces many conveniences in the form of devices (some estimates indicate about 21 billion IoT devices by 2020 [2]), the security aspect of said devices are below par. Mirai used this to its advantage, and its modus operandi is as follows: It scans, attempts a brute force entry point, reports back to an element of the botnet, compromises the IoT device, and the cycle continues. Initially the CNC is set up, and the

scanning process begins from within the CNC. Once a susceptible machine is found, the bruting process begins, where the CNC bot attempts to login using a hard-coded list of login usernames and passwords. If the bruting is successful, the initial CNC bot reports the information to the scanListen server, which maintains a log of the login credentials. The scanListen then connects to the loader, who will load Mirai onto the bruted device, which takes over the device and forces it to repeat the process of: scan, brute, scanListen, scan and so on [1].

Mirai is an important case to study for a variety of reasons. First, the source code is publicly available, meaning that there will be variations and continual usage of this malware [1]. Secondly, there is the Mirai architecture to be considered. There are different foundations upon which botnets are built, for instance an IRC botnet which has a centralized architecture. The IRC botnet has a distinct signature, can easily be detected and has a single point of failurezhao-botnet-detection. Peer-to-peer botnets are built with an architecture similar to P2P sharing, and thus does not suffer from single points of failure zhao-botnet-detection. However, P2P botnets do suffer from latency in the CNC transmission, and this leads to an inhibition of the bots' synchronization zhao-botnet-detection. Mirai uses several compromised computers from mid-sized businesses as its CNC servers, and it avoids detection by changing its location three times as much as other IoT botnets [2]. In this contribution we hope to determine what, if any, Mirai fingerprints exist.

Approach

Mirai's traffic was captured by Impact, from the time period of June to December 2016. This data set includes the scanning traffic that infected Mirai bots send out to the network. Impact's darknet acted as a sinkhole, which pulled in all the SYN scanning traffic [3]. This was not ideal data, as it did not have any other communications, aside from scan data. There was no attack data, or any scanListen/loader communication, or even simple CNC communication. This left us with a very one sided view of the botnet's attack methods. However, this does not mean this was completely useless data. In fact, the scan data could instead be used to look for Mirai fingerprints.

Normally, when a packet is sent out, the operating system is in charge of setting up the headers, like the TCP and IP header. Mirai, on the other hand, when it sends out scan packets, takes the reigns and sets the headers up in a specific way. One example of a hardcoded Mirai fingerprint is the sequence number in the TCP header. Mirai shifts the destination IP address until it is in integer form, rather than dotted notation, and uses that as the sequence number when sending a scan packet to that destination. This is how Impact was able to gather a corpus of scan data, by filtering for this fingerprint [3].

We intend to use tools and scripts like p0f, the passive OS fingerprinting tool, to determine if the operating systems behind the infected devices are detectable. This will allow us to know if Mirai behaves differently based on the different operating system that it occupies. Impact's dataset can be

analyzed by examining specific header fields like TTL, window size, id, and others.

We would also like to use packet analysis tools like ARGUS [4] or SiLK SiLK on the scan data. This would help us visualize whether any useful patterns exist. This could lead to some sort of data binning by time or frequency, to locate any deeper identifying factors that Mirai enables. The data needs to be analyzed using tools that will allow for making interconnections within the set. SiLK [10]is one such set of tools that allow for network traffic collection and analysis. The tools are highly useful for enabling analysts to quickly and efficiently query large traffic databases.

Another approach to have a comprehensive review of Mirai would be to set up the actual botnet. This will be done within the confines of DETER, the experimental sandbox setup. Certain measures would have to be taken to ensure that Mirai did not escape, because it is a highly aggressive scanner. This would mean setting up certain firewall rules, changing Mirai's source code, and of course setting up an internal DNS server, because DETER's setup is not connected to the internet.

Due to the fact that there is not that much information within the scan data, at first glance, we attempt to gather all the different types of Mirai communication, and attempt a circular analysis. In essence, this contribution attempts to replicate the results, and compare with what already exists.

Aside from the Impact data, and the data collected on Deter, we collected data from a home network router. This is important because it allowed us

to see how Mirai was acting in real time, and what variations of it exist in the wild. We hope to use data from a wide variety of sources, to get a more complete understanding of this botnet.

Expected Outcomes

We expect to deliver visuals of the Mirai dataset. We hope to depict characteristics in the data that can successfully identify the early stages of a Mirai botnet attack. This could include packet transmission from the botmaster to the zombie bots, such as CNC messages, or zombie bots scanning for more victims to conscript to the army.

Outcomes

When using p0f, we initially ran into some issues with the third version, which did not seem to be compatible with zipped files, and because of time issues, we were forced to use the second version of p0f. Running p0f, with the packaged fingerprint database list was entirely unsuccessful, as every single IP address in the scan data pcaps returned with "Unknown". This makes sense, as Mirai was overriding the onboard operating system, and making its own header fields. The next approach was to define our own fingerprint, and label it as Linux MiraiInfected. The contents of this fingerprint was the result of a careful perusal of the scan data. p0f dictates that the fingerprint should be listed in the following form: "wwww:ttt:D:ss:000...:QQ:OS:Details". The first field is for window size, then time to live, then whether the don't fragment bit is set or not, followed by the overall SYN packet size, then the

option value, and lastly the operating system and specific version/details of it. The scan data indicated that window sizes were almost unique, and apparently random. Mirai's source code supported this theory as it was using a random function to assign the window size. Therefore, we left the window size with an arbitrary value, and moved onto ttl. The ttl was interesting because, using ARGUS, we realized that there was a definite pattern. When the data was binned and organized by ttl, we noted that the times were centralized around 46. A great majority of the packets fell into that bucket, whereas the ttl's of 33 and 200 held the few outliers. Some research later suggested that Linux systems have a ttl set to 64, and knowing that many IoT devices have Linux onboard, we chose 64 for the ttl of choice greene-iot. The don't fragment flag was not set for any of the packets, and this is normal as packets usually don't have this flag set. The length of the packets were either 40 or 44, almost entirely. The latter size discrepancy was because some packets had the MSS option filled out, while others didn't. There were far more packets that did not have the MSS option, than those with it. The packets did not have any options set. We specified for the operating system to be called MiraiInfected, and re-ran p0f. This time, nearly all the packets fell into either MiraiInfected packet size 44 or MiraiInfected packet size 40. These results suggested more work needed. Because all of the packets keep falling into the same operating system fingerprint, we needed another identifier that would differentiate between different operating systems.

We decided to work with ARGUS and SiLK as well. As mentioned ear-

lier, ARGUS helped us identify the normalized distribution of the ttl values. However, when we binned the data by time, we found no significant results. Though we binned by time, there was approximately the similar amounts of packet in each time bin. The packets in each bin increased as the pcaps got closer to the attack date.

Within the experiment on DETER, we were able to build Mirai and it's scanListen and loader components. Because DETER is a closed network, we were not able to access the internet at all. But, Mirai used Google's DNS server to communicate and scan. As a result, we had to switch this, and build a DNS server within the network and have it route the communications. Once all the cross compilers, and other dependencies were installed, we were able to build Mirai and began the attack phases. We were able to capture data that displayed the attacker logging onto the CNC and accessing the bots, and sending attack commands to the bots. The recreation of the botnet also enabled the capture of scan data and loader data. The scan data is important because we can use it for circular analysis. We were able to put the scan data back into p0f and measure whether p0f could detect Mirai enabled packets. Any packet that was sent from Mirai during the scan process returned as "Unknown". This was important to note, it supported our earlier findings.

We also used the DETER setup to create an vulnerable machine. This target node was running telnet and had a user with one of the login credentials that was in Mirai's password list. Once we began the scanning process, we began capturing packets where the CNC initial bot was attempting to brute

force its way into the vulnerable target. Analysis of these pcaps revealed a better picture of what Mirai does as it runs. For instance, we were able to replicate the syn scan. The destination address was converted into the sequence number. This time, however, we were able to pick up the target machine's response. Before we setup telnet to listen on port 23, to allow Mirai to brute force the password, the target machine refused to accept the CNC's SYN packet. Every time the CNC would send a SYN scan, the target would reset the connection. Once we enabled telnet, we noticed that the target node began sending SYN ACK packets, attempting to talk back to the CNC. Generally, the CNC kept using the same sequence number, but in some cases it used a new, apparently random number. And when the CNC began pushing data, or brute forcing the password, the sequence numbers changed to indicate how to later compose the byte stream with all the received segments.

In order to collect data from the our local routers, we ran iptables to ensure that packets that were dropped would be logged. Using the same method Impact used, we were able to filter out only the Mirai scans, which suggested that there are still strains of Mirai that maintain the sequence number fingerprint. Using this filtered log, we then found the frequencies of the ttl fields, which again was clustered around the mid-50's, which is a little higher than what was found in Impact's data (around 40's). We also filtered out the window sizes, and the data indicated that a large percentage of the packets had a unique window size. There were some anomalies, where

a window size was repeated, but it was far and few in between. Interestingly enough, the frequencies corresponded in the following way, for instance, the source ip v.x.y.z sent 30 SYN packets, and the same log contained a window size of 62240 that appeared 30 times. This pattern appeared in a constant fashion. This could suggest that if a source ip sends multiple packets to one destination, it sends the same packet repeatedly. The size of the packets were distributed equivalently to what was seen in Impact's data, with the greater portion being 40, and a smaller fraction being 44. An interesting aspect was the destination ports; the logs indicated that Mirai was targeting 23, 2323, but also new ports like 80, 81, 2222, 23231, which is one of the obvious signs of new strains of Mirai.

Conclusion and Further Endeavors

The approaches so far were successful in providing a baseline starting point. We have determined that some fingerprints exist, such as the sequence number and the length of the packet. However, any deeper fingerprints remained hidden. The time binning did not work, nor did the passive os fingerprinting. While we were initially unsuccessful, this is by no means a dead end, as we still have further options to explore.

One future path of exploration would be in the frequency domain, or the frequency of occurance. For instance, this can be extrapolated to the passive operating systems fingerprinting. If we find that certain IP addresses are communicating at different frequencies, then this could be a resulting effect

of the different operating systems behind the IP addresses. One attempt would be to locate the "top talkers" using ARGUS, over the course of some time period, and then extracting all of the instances of the corresponding SYN packets. This data could be used to calculate the frequency of packets over the time domain, which would then involve Fourier or wavelet transformations wavelets. Any patterns or outputs from this data would give us another view point of the patterns in the undercurrents of Mirai.

Another option would be to consider machine learning. To reach some conclusive statement that xyz traffic is actually Mirai traffic, we can use the Impact dataset, combined with normal traffic in order to train a model and determine which is malicious and which is benign traffic. An algorithm would be trained on normalized traffic from Windows, Linux and iOS operating systems; the training will be done over their respective classifiers. Once the model is trained, it can be used on the scan traffic data, and we can attempt operating system detection.

As mentioned earlier, the current available process for gathering Mirai data is to filter by sequence number, if it matches the destination IP address. This is the main fingerprint which was used to collect the Impact data, as well as my router data. The DETER lab data did not suggest otherwise, rather it supported this fingerprint. However, it must be noted that newer versions of Mirai may not keep the sequence number fingerprint, and this would mean any efforts to filter for this new version would fail using the aforementioned method. Therefore, we are currently biasing the newer versions, should they

exist, out of the picture.

Ethics [11]

- Respect For Persons: This study uses a dataset with IP addresses that have been anonymized using cryptopan.
- Beneficence: We are minimizing the harm by anonymizing the IP addresses, but at the same time, we preserve the prefixes which allow for the data to be meaningful. Furthermore, we will run all instances of the botnet within a closed system on DETER, which will minimize the effects of the malware, while allowing the data to be collected.
- Respect for Law and Public Interest: Given the procedure stated above, this research does not attempt to go beyond the limits of the law. The malware will not be used in any public networks.

References

[1] A. Senpai, "Mirai forum post," 2016.

[2] L. H. Newman, "The botnet that broke the internet isn't going away," https://www.wired.com/2016/12/botnet-broke-internet-isnt-going-away/.

[3] IMPACT. (2016) Mirai scanning 2016. [Online]. Available: https://impactcybertrust.org/dataset$_v iew?idDataset = 717$

[4] C. Bullard. (2016) Argus. [Online]. Available: https://qosient.com/argus/

[5] P. Barford, C. Partridge, and W. Willinger, "Internet multi-resolution analysis: A vision and framework in support of representing, analyzing, and visualizing internet measurements."

[6] MATLAB, "Understanding wavelets, part 1: What are wavelets," https://www.youtube.com/watch?v=QX1-xGVFqmw, August 2016.

[7] T.-F. Yen and M. K. Reiter, "Traffic aggregation for malware detection," in *Proceedings of the 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, ser. DIMVA '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 207–227. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-70542-$0_1$1

[8] J. Wiens, "Ids deconstructed," February 2006.

[9] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment*, ser. IMW '02. New York, NY, USA: ACM, 2002, pp. 71–82. [Online]. Available: http://doi.acm.org/10.1145/637201.637210

[10] CERT/NetSA at Carnegie Mellon University, "SiLK (System for Internet-Level Knowledge)," [Online]. Available: http://tools.netsa.cert.org/silk., [Accessed: July 13, 2009].

[11] D. Dittrich and E. Kenneally, "The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research," U.S. Department of Homeland Security, Tech. Rep., Aug 2012.