(E)xaTrack

Tricephalic Hellkeeper: a tale of a passive backdoor

Tristan Pourcelot (tristan.pourcelot [at] exatrack.com)

We recently found a new passive backdoor targeting Linux and Solaris servers, which can use TCP, UDP or ICMP packets as triggers.

In this article we will dive into BPF in order to assess this malware capabilities :D

http://www.exatrack.com

Content

١.	Introduction to Passive backdoors 2	<u>)</u>
II.	Capabilities	2
III.	Persistance	2
IV.	Protection techniques	2
V.	In memory launch	3
VI.	Process renaming	3
VII.	Timestomping	3
VIII.	Command and Control	3
IX.	BPF filter analysis	3
Х.	Passwords / commands	5
XI.	Encryption6	5
XII.	Aside note: a backdoor's bug6	5
XIII.	Conclusion	7
XIV.	Other publications	7
XV.	Annexes	7
10	DCs - filenames	7
10	DCs - Hashes	7
10	Ccs - Yara rule ٤	3
Li	ist of process names)

I. Introduction to Passive backdoors

Passive backdoors are implants designed to be stealthier than common backdoors, especially by avoiding listening on ports or pinging back to a Command and Control server.

These backdoors are not novel, one of the first sample being cd00r from *phenoelit*, a passive backdoor waiting for a TCP port knocking sequence, dating back to 2000 [1].

Passive backdoors are also used by some of the most advanced attackers, such as *Duqu2* with their portserv.sys driver, the famous *Equation Group* with their Bvp47 or DewDrop implant [2, 3], or the *Turla* threat group with their Uroburos rootkit that we previously analyzed [11].

This kind of backdoor has several advantages:

- They cannot be detected by tools such as netstat because they don't open network ports
- They can be activated by piggybacking on already opened ports (for example, a packet sent to a legitimate HTTP listening port will trigger the backdoor)
- They are inactive unless activated by the attacker.

While not as complex as Bvp47, this backdoor provided the attackers with simple yet powerful capabilities, such as a remote access to the infected systems.

II. Capabilities

This backdoor family uses a BPF filter in order to await a trigger packet, and depending on the received command will either send a ping back, launch a bind shell, or connect a remote shell to the attacker provided IP address.

The implant expects to be launched as the privileged root user, and uses a lock in order to avoid being launched several times. The attacker took special attention to make the lock's filename appears legitimate, and used several of these filenames across the samples. Some of them can be found in the IOCs Annex of this article.

III. Persistence

The binary does not implement any kind of persistence. We assume that the attacker will provide the persistence by modifying configuration files on the targeted system during installation.

IV. Protection techniques

The malware uses several techniques in order to avoid detection, these techniques varying between samples. Some of them relaunched from memory, in order to avoid traces on the file system, while others modified their access and modification timestamp.



V. In memory launch

Some of the samples copied themselves in the /dev/shm folder with a custom filename, before relaunching the copied sample. This technique avoid leaving traces on the target file system, and ensures the binary is completely removed on reboot.

VI. Process renaming

The malware will rename itself using the prctl function with the argument PR_SET_NAME, and a random legitimate-looking name.

These names are hardcoded in the binary, and vary between the samples.

A partial list of used process names can be found in annex of this article.

VII. Timestomping

In some of the samples, a function dubbed set_time was called to alter the access and modification timestamp of the binary using the utimes function. The timestamp used was always set to 0x490a083c (2008-10-30T20:17:16).

VIII. Command and Control

The command and control mechanism of this backdoor is relatively simple, it waits for a trigger packet, then depending on the command, will establish a bind shell, a reverse shell, or send a reply to the attacker.

IX. BPF filter analysis

Berkeley Packet Filter, or BPF, is a technology dating back to the early 90s which was initially designed to filter network packets.

This filtering subsystem is documented in the Linux kernel [5, 6], and is used under the hood by tools such as tcpdump.

This technology was also used by implants such as dewdrop from the Equation toolset [3].

BPF packet filters are implemented using a custom bytecode consisting of about two dozen opcodes, all following the same pattern: {opcode, jt, jf, k}.

Once compiled, such a filter can be instantiated using either libpcap (pcap_setfilter), or the standard library function setsockopt using the so_ATTACH_FILTER option.

In the case of this backdoor, the latter was chosen, and the BPF bytecode is included in binary form in the sample.

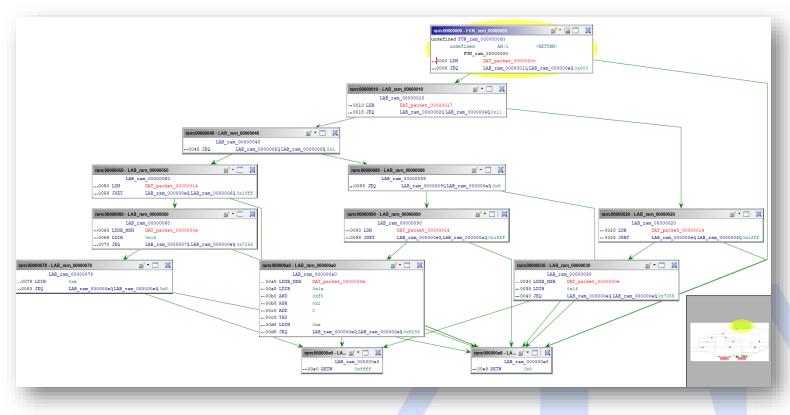
The pseudo-code of the filter installation and packet parsing loop look like this:



```
sock_fprog fprog;
  [...]
  // Simplified copy loop
 memcpy(&stored_data, &filter_bytecode, 0x1e * 8);
  [...]
  fprog.len = 0x1e;
  fprog.filter = &stored data;
  // Create a raw socket
 hSocket = socket(AF PACKET, SOCK RAW, (uint)uVar1);
 if ((hSocket < 1) || (iVar2 = setsockopt(hSocket, 1, SO ATTACH FILTER, &fprog,
sizeof(sock fprog)), iVar2 == -1)) {
   return;
  }
  while( true ) {
    do {
      do {
        memset(received buffer,0,0x200);
        recvfrom(dwCMD,received_buffer,0x200,0,(sockaddr *)0x0,(socklen_t *)0x0);
        //\ensuremath{\,\text{Now}} , parse the packet
  [...]
```

We chose to dump it from the sample, and developed a BPF processor for Ghidra [7] to disassemble it. This processor will be released shortly after some more testing :)

The disassembled graph is the following:



BPF Bytecode disassembly

The BPF bytecode processes the packet in several steps:

1) Filter for IPv4 traffic

@ExaTrack

http://www.exatrack.com

Page 4

- 2) Triage between UDP / TCP or ICMP traffic
- 3) Get the first 2 bytes of packet data
- 4) Check if these 2 bytes looks like a trigger packet (0x5293 for TCP, 0x7255 for UDP and ICMP)
- 5) Return the packet if true

While searching for similar samples, we also found a Solaris variant of this malware. This variant did not use the setsockopt function, but instead relied on libpcap to install its filter. This time the filter could be found in text form, which confirmed our analysis of the BPF byte code:

(udp[8:2]=0x7255) or (icmp[8:2]=0x7255) or (tcp[((tcp[12]&0xf0)>>2):2]=0x5293)

We can deduce from this that the backdoor can be triggered by three different means:

- 1) An UDP packet starting with the bytes 0x7255
- 2) An ICMP packet also starting with the bytes 0x7255

A TCP packet starting with the bytes 0x5293.

X. Passwords / commands

If a packet is matched by the filter, it is returned to the malware process in order to be parsed. The expected format is the following:

```
+---+--+--+--+--+--+--+--+

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

+---+--+--+--+--+--+--+--+--+

| MAGIC |PADDING|PING IP ADDRESS|

+-----+----+

| PORT | PASSWORD (OR COMMAND) |

+-----+---+

| PASSWORD (continued) |
```

The processing function will extract the command string from the packet, and compare it to a set of hardcoded names, such as

• justtryit, justrobot, justforfun, which will establish a bind shell on ports 42391 to 42491;

• socket or sockettop for establishing a reverse shell to an IP address provided in the packet. By default, the implant will send the character 1 (encoded as 0x31) to the IP address provided in the packet.

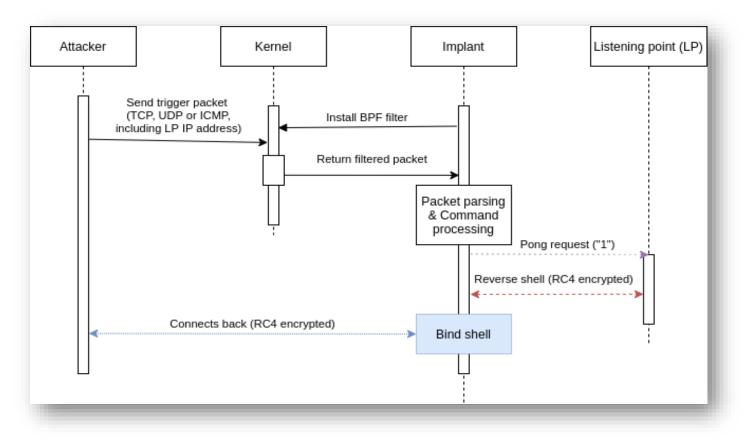
Both the bind shell and the reverse shell processes are renamed (often as /usr/libexec/postfix/master). The attacker is also provided with a clean environment, the following environment variables being set:

- PROMPT
- HISTFILE=/dev/null (this avoids leaving traces in Bash history files)
- MYSQL HISTFILE=/dev/null
- PS1=[\u@\h \W]\\\$
- HOME=/tmp **Or** HOME=/



It should be noted that some of the latest variants of the malware are no longer using keywords such as socket for triggering its capabilities, but are using MD5 hashes. It may imply that the attacker has improved their implant in order to be more resilient or secure.

The whole behavior of the implant can be summarized in the following schema:



Implant behavior

XI. Encryption

If the malware launches the bind shell or the reverse shell, it will encrypt its traffic using the RC4 algorithm. The key used is simply the received command as seen in the previous part (such as socket or justforfun).

XII. Aside note: a backdoor's bug

There is a slight bug in the malware developer code, which will trigger only if the backdoor is launched when the server is receiving a lot of traffic.

This bug was documented in [8], and is due to the fact that the socket can receive traffic before the BPF filter is applied.



XIII. Conclusion

The idea behind this backdoor was neither new nor complicated. However, it provided the attacker with critical capabilities against the targeted networks. It also allowed them to remain stealthy for a long time.

The attacker was regularly updating its toolset with new keywords, improving a little bit its implant with each release by changing their command names, their process names, or their filenames in order to avoid trivial detection.

XIV. Other publications

While finishing this article and a set of slides presented at an event in early May, we noticed an unusual number of rescans of this malware family. This was due to the fact that security researcher @GossiTheDog tweeted a corresponding hash [9].

This malware family seems to be tracked by PwC as *BPFDoor* used by the *Red Menshen* threat actor, and we look forward to their presentation at Troopers 2022 [10]!

XV. Annexes

IOCs - filenames

These filenames can be used as IOCs because, while looking legitimate, they are used only by this malware family.

- /var/run/xinetd.lock
- /var/run/kdevrund.pid
- /var/run/haldrund.pid
- /var/run/syslogd.reboot

IOCs - Hashes

•	07ecb1f2d9ffbd20a46cd36cd06b022db3cc8e45b1ecab62cd11f9ca7a26ab6d		
•	a002f27f1abb599f24e727c811efa36d2d523e586a82134e9b3e8454dde6a089		
•	8b84336e73c6a6d154e685d3729dfa4e08e4a3f136f0b2e7c6e5970df9145e95		
•	76bf736b25d5c9aaf6a84edd4e615796fffc338a893b49c120c0b4941ce37925		
•	96e906128095dead57fdc9ce8688bb889166b67c9a1b8fdb93d7cff7f3836bb9		
•	599ae527f10ddb4625687748b7d3734ee51673b664f2e5d0346e64f85e185683		
•	2e0aa3da45a0360d051359e1a038beff8551b957698f21756cfc6ed5539e4bdb		
•	f47de978da1dbfc5e0f195745e3368d3ceef034e964817c66ba01396a1953d72		
•	3347ddcc909c573a27c157f55d0444954e2b4b749bc65607a9f0319217954ac5		
•	54a4b3c2ac34f1913634ab9be5f85cde19445d01260bb15bcd1d52ebcc85af2c	(Variant	with
embedo	led configuration)		
•	fa0defdabd9fd43fe2ef1ec33574ea1af1290bd3d763fdb2bed443f2bd996d73	(2018 varia	nt)
•	591198c234416c6ccbcea6967963ca2ca0f17050be7eed1602198308d9127c78	(unstrip	oped
variant)			



```
dc8346bf443b7b453f062740d8ae8d8d7ce879672810f4296158f90359dcae3a
```

(Solaris

Variant)

IOCs - Yara rule

```
rule Linux TricephalicImplant {
   meta:
       author = "Exatrack"
       description = "Detect Linux passive backdoors"
        tlp = "WHITE"
        source = "Exatrack"
   strings:
        $str message 01 = "hald-addon-acpi: listening on acpi kernel interface
/proc/acpi/event"
        $str message 02 = "/var/run/haldrund.pid"
        $str message 03 = "/bin/rm -f /dev/shm/%s;/bin/cp %s /dev/shm/%s && /bin/chmod
755 /dev/shm/%s && /dev/shm/%s --init && /bin/rm -f /dev/shm/%s" // in the stack
        $str message 04 = "Cant fork pty"
        $str_hald_05 = "/sbin/iptables -t nat -D PREROUTING -p tcp -s %s --dport %d -j
REDIRECT --to-ports %d"
        $str command 01 = "/sbin/iptables -t nat -A PREROUTING -p tcp -s %s --dport %d -j
REDIRECT --to-ports %d"
        $str command 02 = "/sbin/iptables -I INPUT -p tcp -s %s -j ACCEPT"
        $str command 03 = "/bin/rm -f /dev/shm/%s"
        $str command 04 = "/bin/cp %s /dev/shm/%s"
        $str command 05 = "/bin/chmod 755 /dev/shm/%s"
        $str command 06 = "/dev/shm/%s --init"
        $str_server_01 = "[+] Spawn shell ok."
        $str server 02 = "[+] Monitor packet send."
        $str server 03 = "[-] Spawn shell failed."
        $str server 04 = "[-] Can't write auth challenge"
        $str server 05 = "[+] Packet Successfuly Sending %d Size."
        $str_server_06 = "[+] Challenging %s."
        str server 07 = "[+] Auth send ok."
        $str server 08 = "[+] possible windows"
        $str filter 01 = "(udp[8:2]=0x7255)"
        $str filter 02 = "(icmp[8:2]=0x7255)"
        $str_filter_03 = "(tcp[((tcp[12]&0xf0)>>2):2]=0x5293)"
        $str filter 04 = {15 00 ?? ?? 55 72 00 00}
        $str filter 05 = {15 00 ?? ?? 93 52 00 00}
        $error 01 = "[-] socket"
        $error 02 = "[-] listen"
        $error_03 = "[-] bind"
        $error 04 = "[-] accept"
        $error_05 = "[-] Mode error."
        $error 06 = "[-] bind port failed."
        $error 07 = "[-] setsockopt"
        \ensuremath{\$error}_{08} = "[-] missing -s"
        $error 09 = "[-] sendto"
   condition:
        any of ($str*) or 3 of ($error*)
}
```



Disassembled BPF bytecode // Filter for IPv4 traffic 0 28 00 00 00 0c 00 00 00 ldh [0xc] 8 15 00 00 1b 00 08 00 00 jeq 0x800, +0x0, +0x1b 10 30 00 00 00 17 00 00 00 1db [0x17] // Check for UDP 18 15 00 00 05 11 00 00 00 jeq 0x11, +0x0, +0x5 20 28 00 00 00 14 00 00 00 1dh [0x14] 28 45 00 17 00 ff 1f 00 00 jset 0x1fff, +0x17, +0x0 // Get packet first 2 bytes of data 30 b1 00 00 00 0e 00 00 00 ldxb 4*([0xe]&0xf) 38 48 00 00 00 16 00 00 00 1dh [x+0x16] // Check for trigger marker 0x7255 40 15 00 13 14 55 72 00 00 jeq 0x7255, +0x13, +0x14 // Check for TCP protocol 48 15 00 00 07 01 00 00 00 jeq 0x1, +0x0, +0x7 // Get packet first two bytes 50 28 00 00 00 14 00 00 00 1dh [0x14] 58 45 00 11 00 ff 1f 00 00 jset 0x1fff, +0x11, +0x0 60 b1 00 00 00 0e 00 00 00 ldxb 4*([0xe]&0xf 68 48 00 00 00 16 00 00 00 ldh [x+0x16] // Check for trigger marker 70 15 00 00 0e 55 72 00 00 jeg 0x7255, +0x0, +0xe // Check for TCP packet format 78 50 00 00 00 0e 00 00 00 ldb [x+0xe] 80 15 00 0b 0c 08 00 00 00 jeq 0x8, +0xb, +0xc 88 15 00 00 0b 06 00 00 00 jeq 0x6, +0x0, +0xb 90 28 00 00 00 14 00 00 00 ldh [0x14] 98 45 00 09 00 ff 1f 00 00 jset 0x1fff, +0x9, +0x0 // Get packet data a0 b1 00 00 00 0e 00 00 00 ldxb 4*([0xe]&0xf) 50 00 00 00 1a 00 00 00 ldb [x+0x1a] a8 b0 54 00 00 00 f0 00 00 00 and 0xf0 b8 74 00 00 00 02 00 00 00 rsh 0x2 c0 0c 00 00 00 00 00 00 00 add x c8 07 00 00 00 00 00 00 00 tax d0 48 00 00 00 0e 00 00 00 1dh [x+0xe] // Filter for trigger packet (this time 0x5293) d8 15 00 00 01 93 52 00 00 jeq 0x5293, +0x0, +0x1 e0 06 00 00 00 ff ff 00 00 ret 0xffff e8 06 00 00 00 00 00 00 00 ret 0x0

List of process names

This is a partial list of process names which can be used by the malware to masquerade himself.

- /sbin/udevd -d
- /sbin/mingetty /dev/tty
- /usr/sbin/console-kit-daemon --no-daemon
- hald-addon-acpi: listening on acpi kernel interface /proc/acpi/event

- dbus-daemon --system
- hald-runner
- pickup -l -t fifo -u
- avahi-daemon: chroot helper
- /sbin/auditd -n
- /usr/lib/systemd/systemd-journald
- /usr/lib/systemd/systemd-machined

References

[5]

• [1] cd00r <u>https://packetstormsecurity.com/files/22121/cd00r.c.html</u>

• [2] The Bvp47 - a Top-tier Backdoor of US NSA Equation Group https://www.pangulab.cn/en/post/the_bvp47_a_top-tier_backdoor_of_us_nsa_equation_group/

- [3] Knock Knock! Who's There? An NSA VM <u>https://reverse.put.as/2021/12/17/knock-knock-whos-there/</u>
- [4] Hive BPF <u>https://twitter.com/vx_herm1t/status/1445773123668807680</u>

BPF

OpCodes

https://elixir.bootlin.com/linux/latest/source/include/uapi/linux/bpf_common.h

- [6] BPF filters documentation <u>https://lwn.net/Articles/593476/</u>
- [7] Ghidra <u>https://ghidra-sre.org/</u>
- [8] The wrong way to filter sockets with BPF <u>https://natanyellin.com/posts/ebpf-filtering-done-right/</u>
- [9] Tweet on Red Menshen https://twitter.com/GossiTheDog/status/1522000023092965376/
- [10] Tinker Telco Soldier Spy https://troopers.de/troopers22/talks/7cv8pz/
- [11] Hey Uroburos! What's up ? <u>https://exatrack.com/public/Uroburos_EN.pdf</u>
- Title inspiration / Reverse OST <u>https://www.discogs.com/release/11125255-Kronos-The-</u> <u>Hellenic-Terror</u>