Opening “STEELCORGI”: A Sophisticated APT Swiss Army Knife

yoroi.company/research/opening-steelcorgi-a-sophisticated-apt-swiss-army-knife

January 12, 2021

01/12/2021

Introduction

2020 was a really intense year in terms of APT activities, in fact it brought us new evidence of sophisticated campaigns targeting Enterprises organization across Europe and also Italy. In particular the threat group we track as TH-239, also mentioned as UNC1945 by FireEye security researchers, has been one of the sneakiest.

We discussed some of the new techniques and modus operandi used by this actor in our previous post, revealing how it leverages modern post exploitation tools even in legacy environments such as old Linux-based machines: with the help of a portable virtual machine, TH-239 is able to move part of its arsenal directly into the victim's internal network.

This time we decided to dissect and share intelligence information about another piece of the TH-239 arsenal: a tiny and mysterious tool dubbed “STEELCORGI” on FireEye research. This tool was heavily protected using a novel technique able to make things really difficult to any DFIR Team tackling with TH-239 intrusion, but it’s contents reveal huge surprises and unattended capabilities.

Technical Analysis

One of the most interesting components of the TH-239arsenal is an ELF binary file classified as “STEELCORGI”. The tool is presented in the form of an ELF named with the following md5: 0845835e18a3ed4057498250d30a11b1.

This binary is protected in a very aggressive way, let’s see how.

A Packed ELF

During the analysis we noticed that this ELF was very far from being readable, we extracted a series of elements confirming us that:

- High file dimension (more than 4MB);
- Obfuscated strings;
- Absence of Dynamic and (*.dynsym*) and Static Symbol Tables (*.symtab*);
- Absence of *section-headers* as Anti-reverse engineering Technique;
- High value of entropy > 7.9
- Runtime linking mechanism with *dlopen* and *dlsym*

As the first step, we focused on the static analysis of the sample in order to reconstruct the high level of sophistication and complexity of the packing. At first impact, strings are obfuscated, the binary is dynamically linked but the dynamic symbols table is empty.

Also, the absence of section-headers is an anti-reverse engineering technique adopted in this packer. Another indicator that the binary is packed is the high value of entropy *7.99*, as it is possible to observe in the following picture, on the right we have the whole portion of the ELF binary with compressed data.

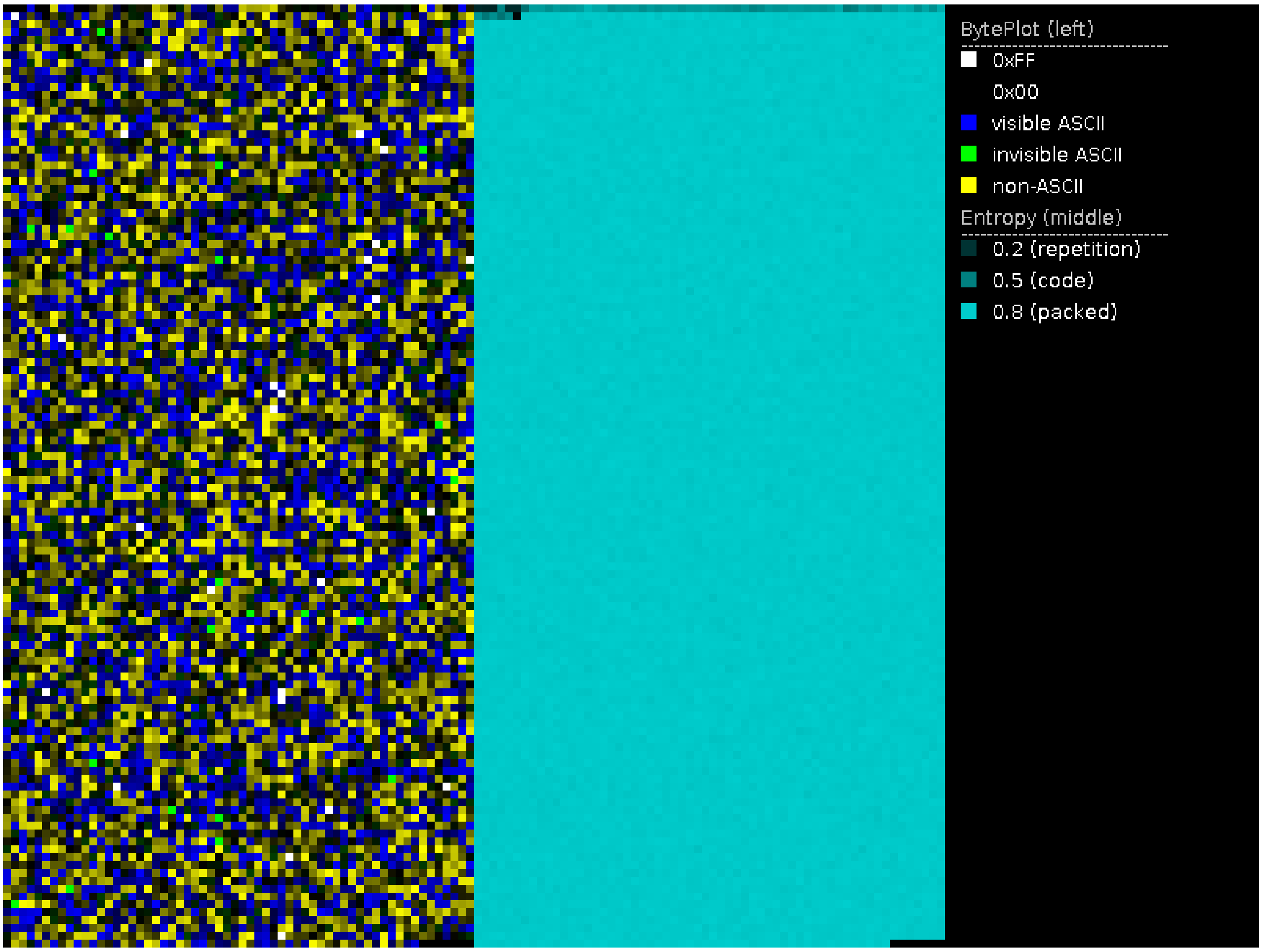
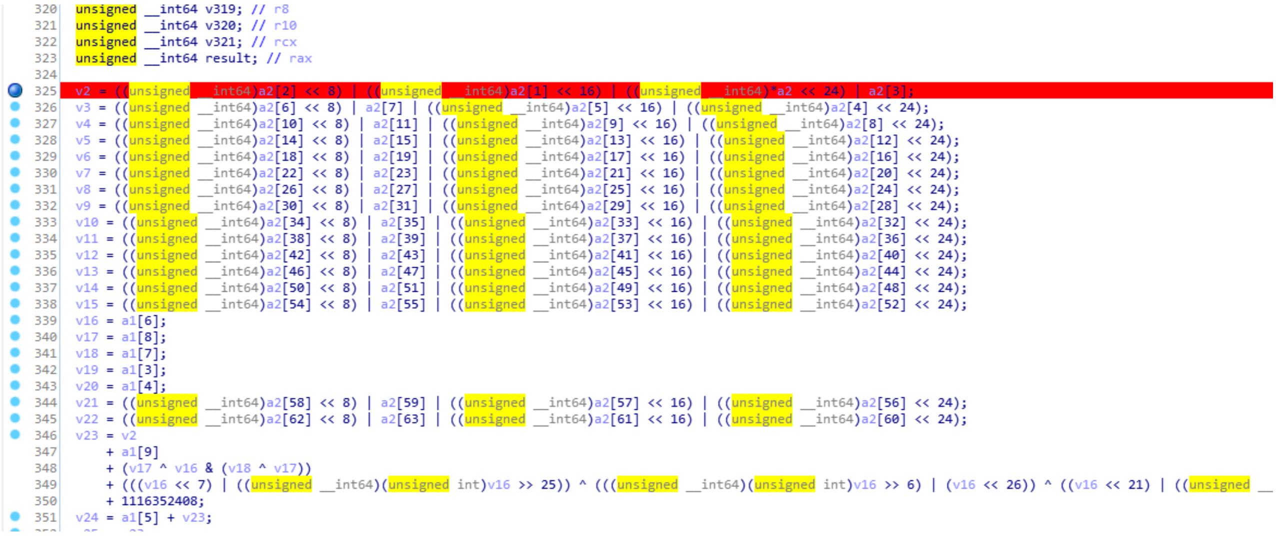


Figure. High entropy section

At this point, we aren't able to retrieve any other information about the packer, so we have to analyze the malicious routines aimed at unpack the sample. During the code inspection, a very long and complex subroutine emerges and it looks like the following screen:



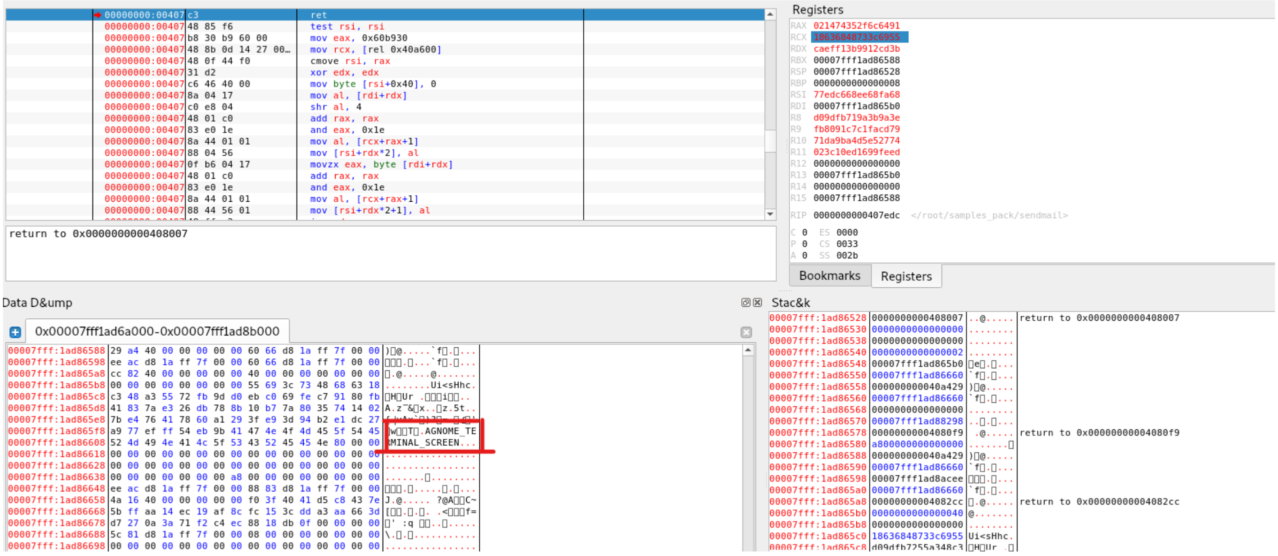
```
unsigned __int64 v319; // r8
unsigned __int64 v320; // r10
unsigned __int64 v321; // rcx
unsigned __int64 result; // rax

v2 = ((unsigned __int64)a2[2] << 8) | ((unsigned __int64)a2[1] << 16) | ((unsigned __int64)*a2 << 24) | a2[3];
v3 = ((unsigned __int64)a2[6] << 8) | a2[7] | ((unsigned __int64)a2[5] << 16) | ((unsigned __int64)a2[4] << 24);
v4 = ((unsigned __int64)a2[10] << 8) | a2[11] | ((unsigned __int64)a2[9] << 16) | ((unsigned __int64)a2[8] << 24);
v5 = ((unsigned __int64)a2[14] << 8) | a2[15] | ((unsigned __int64)a2[13] << 16) | ((unsigned __int64)a2[12] << 24);
v6 = ((unsigned __int64)a2[18] << 8) | a2[19] | ((unsigned __int64)a2[17] << 16) | ((unsigned __int64)a2[16] << 24);
v7 = ((unsigned __int64)a2[22] << 8) | a2[23] | ((unsigned __int64)a2[21] << 16) | ((unsigned __int64)a2[20] << 24);
v8 = ((unsigned __int64)a2[26] << 8) | a2[27] | ((unsigned __int64)a2[25] << 16) | ((unsigned __int64)a2[24] << 24);
v9 = ((unsigned __int64)a2[30] << 8) | a2[31] | ((unsigned __int64)a2[29] << 16) | ((unsigned __int64)a2[28] << 24);
v10 = ((unsigned __int64)a2[34] << 8) | a2[35] | ((unsigned __int64)a2[33] << 16) | ((unsigned __int64)a2[32] << 24);
v11 = ((unsigned __int64)a2[38] << 8) | a2[39] | ((unsigned __int64)a2[37] << 16) | ((unsigned __int64)a2[36] << 24);
v12 = ((unsigned __int64)a2[42] << 8) | a2[43] | ((unsigned __int64)a2[41] << 16) | ((unsigned __int64)a2[40] << 24);
v13 = ((unsigned __int64)a2[46] << 8) | a2[47] | ((unsigned __int64)a2[45] << 16) | ((unsigned __int64)a2[44] << 24);
v14 = ((unsigned __int64)a2[50] << 8) | a2[51] | ((unsigned __int64)a2[49] << 16) | ((unsigned __int64)a2[48] << 24);
v15 = ((unsigned __int64)a2[54] << 8) | a2[55] | ((unsigned __int64)a2[53] << 16) | ((unsigned __int64)a2[52] << 24);
v16 = a1[6];
v17 = a1[8];
v18 = a1[7];
v19 = a1[3];
v20 = a1[4];
v21 = ((unsigned __int64)a2[58] << 8) | a2[59] | ((unsigned __int64)a2[57] << 16) | ((unsigned __int64)a2[56] << 24);
v22 = ((unsigned __int64)a2[62] << 8) | a2[63] | ((unsigned __int64)a2[61] << 16) | ((unsigned __int64)a2[60] << 24);
v23 = v2
    + a1[9]
    + (v17 ^ v16 & (v18 ^ v17))
    + (((v16 << 7) | ((unsigned __int64)(unsigned int)v16 >> 25)) ^ (((unsigned __int64)(unsigned int)v16 >> 6) | (v16 << 26)) ^ ((v16 << 21) | ((unsigned __
    + 1116352408;
v24 = a1[5] + v23;
```

Figure. Part of the decoding routines

It is a particular decoding routine instructed to decrypt some other protected code and strings. The code is a complex succession of logic instructions, like xor, shift, or etc. In the end of the decoding routine, the sample performs a check on the environment variables, looking for a custom one installed by the TH-239 operators.

In fact, the environment variable “MCARCH_” contains the decryption key of the protector wrapper. When the malware retrieves the desidered environment variable, it starts the unpacking routine using the key stored in it and then starts the execution of the real payload.

This approach is a great evasion technique because it avoids the execution of the sample in any environments except the ones where TH-239 operators decide to get in.



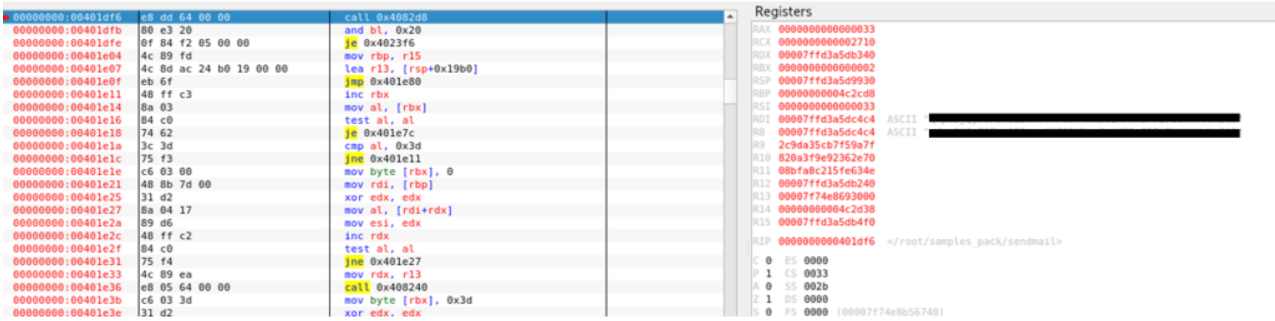
```
00000000:00407 c3                 ret
00000000:00407 48 85 f6           test rsi, rsi
00000000:00407 b8 30 b9 60 00     mov eax, 0x60b930
00000000:00407 48 8b 0d 14 27 00… mov rcx, [rel 0x40a600]
00000000:00407 48 0f 44 f0        cmove rsi, rax
00000000:00407 31 d2              xor edx, edx
00000000:00407 c6 46 40 00        mov byte [rsi+0x40], 0
00000000:00407 8a 04 17           mov al, [rdi+rdx]
00000000:00407 c0 e8 04           shr al, 4
00000000:00407 48 01 c0           add rax, rax
00000000:00407 83 e0 1e           and eax, 0x1e
00000000:00407 8a 44 01 01        mov al, [rcx+rax+1]
00000000:00407 88 04 56           mov [rsi+rdx*2], al
00000000:00407 0f b6 04 17        movzx eax, byte [rdi+rdx]
00000000:00407 48 01 c0           add rax, rax
00000000:00407 83 e0 1e           and eax, 0x1e
00000000:00407 8a 44 01 01        mov al, [rcx+rax+1]
00000000:00407 88 44 56 01        mov [rsi+rdx*2+1], al

return to 0x0000000000408007
```

```
RAX 021474352f6c6491
RCX 18636848733c6955
RDX caeff13b9912cd3b
RBX 00007fff1ad86588
RSP 00007fff1ad86528
RBP 0000000000000008
RSI 77edc668ee68fa68
RDI 00007fff1ad865b0
R8  d09dfb719a3b9a3e
R9  fb8091c7c1facd79
R10 71da9ba4d5e52774
R11 023c10ed1699feed
R12 0000000000000000
R13 00007fff1ad865b0
R14 0000000000000000
R15 00007fff1ad86588

RIP 0000000000407edc  </root/samples_pack/sendmail>

C 0  ES 0000
P 0  CS 0033
A 0  SS 002b
```

```
00007fff:1ad86588 29 a4 40 00 00 00 00 00 60 66 d8 1a ff 7f 00 00 )□@.....`f□.□...
00007fff:1ad86598 ee ac d8 1a ff 7f 00 00 60 66 d8 1a ff 7f 00 00 □□□.□...`f□.□...
00007fff:1ad865a8 cc 82 40 00 00 00 00 00 40 00 00 00 00 00 00 00 □.@.....@.......
00007fff:1ad865b8 00 00 00 00 00 00 00 00 55 69 3c 73 48 68 63 18 ........Ui<sHhc.
00007fff:1ad865c8 c3 48 a3 55 72 fb 9d d0 eb c0 69 fe c7 91 80 fb □H□Ur .□□□i□□..
00007fff:1ad865d8 41 83 7a e3 26 db 78 8b 10 b7 7a 80 35 74 14 02 A.z¯&□x..□z.5t..
00007fff:1ad865e8 7b e4 76 41 78 60 a1 29 3f e9 3d 94 b2 e1 dc 27 {.vAx`□)?□=.□.□'
00007fff:1ad865f8 a9 77 ef ff 54 eb 9b 41 47 4e 4f 4d 45 5f 54 45 □w□□T□.AGNOME_TE
00007fff:1ad86608 52 4d 49 4e 41 4c 5f 53 43 52 45 45 4e 80 00 00 RMINAL_SCREEN...
00007fff:1ad86618 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00007fff:1ad86628 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00007fff:1ad86638 00 00 00 00 00 00 00 a8 00 00 00 00 00 00 00 00 .......□........
00007fff:1ad86648 ee ac d8 1a ff 7f 00 00 88 83 d8 1a ff 7f 00 00 □□□.□.....□.□...
00007fff:1ad86658 4a 16 40 00 00 00 00 00 f0 3f 40 41 d5 c8 43 7e J.@..... ?@A□□C~
00007fff:1ad86668 5b ff aa 14 ec 19 af 8c fc 15 3c dd a3 aa 66 3d [□□.□.□. .<□□□f=
00007fff:1ad86678 d7 27 0a 3a 71 f2 c4 ec 88 18 db 0f 00 00 00 00 □' :q □□..□.....
00007fff:1ad86688 5c 81 d8 1a ff 7f 00 00 08 00 00 00 00 00 00 00 \.□.□...........
00007fff:1ad86698 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

```
00007fff:1ad86528 0000000000408007 ..@..... return to 0x0000000000408007
00007fff:1ad86530 0000000000000000 ........
00007fff:1ad86538 0000000000000000 ........
00007fff:1ad86540 0000000000000002 ........
00007fff:1ad86548 00007fff1ad865b0 □e□.□...
00007fff:1ad86550 00007fff1ad86660 `f□.□...
00007fff:1ad86558 000000000040a429 )□@.....
00007fff:1ad86560 00007fff1ad86660 `f□.□...
00007fff:1ad86568 0000000000000000 ........
00007fff:1ad86570 00007fff1ad88298 ..□.□...
00007fff:1ad86578 00000000004080f9  .@..... return to 0x00000000004080f9
00007fff:1ad86580 a800000000000000 .......□
00007fff:1ad86588 000000000040a429 )□@.....
00007fff:1ad86590 00007fff1ad86660 `f□.□...
00007fff:1ad86598 00007fff1ad8acee □□□.□...
00007fff:1ad865a0 00007fff1ad86660 `f□.□...
00007fff:1ad865a8 00000000004082cc □.@..... return to 0x00000000004082cc
00007fff:1ad865b0 0000000000000040 @.......
00007fff:1ad865b8 0000000000000000 ........
00007fff:1ad865c0 18636848733c6955 Ui<sHhc.
00007fff:1ad865c8 d09dfb7255a348c3 □H□Ur .□
```

Figure. Environment variable lookup



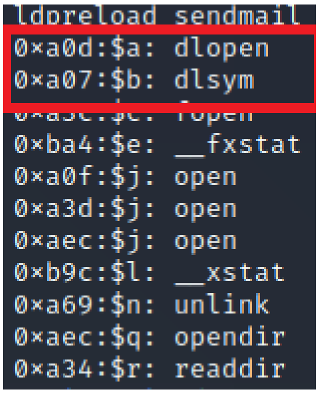
```
00000000:00401df6  e8 dd 64 00 00              call 0x4082d8
00000000:00401dfb  80 e3 20                    and bl, 0x20
00000000:00401dfe  0f 84 f2 05 00 00           je 0x4023f6
00000000:00401e04  4c 89 fd                    mov rbp, r15
00000000:00401e07  4c 8d ac 24 b0 19 00 00     lea r13, [rsp+0x19b0]
00000000:00401e0f  eb 6f                       jmp 0x401e80
00000000:00401e11  48 ff c3                    inc rbx
00000000:00401e14  8a 03                       mov al, [rbx]
00000000:00401e16  84 c0                       test al, al
00000000:00401e18  74 62                       je 0x401e7c
00000000:00401e1a  3c 3d                       cmp al, 0x3d
00000000:00401e1c  75 f3                       jne 0x401e11
00000000:00401e1e  c6 03 00                    mov byte [rbx], 0
00000000:00401e21  48 8b 7d 00                 mov rdi, [rbp]
00000000:00401e25  31 d2                       xor edx, edx
00000000:00401e27  8a 04 17                    mov al, [rdi+rdx]
00000000:00401e2a  89 d6                       mov esi, edx
00000000:00401e2c  48 ff c2                    inc rdx
00000000:00401e2f  84 c0                       test al, al
00000000:00401e31  75 f4                       jne 0x401e27
00000000:00401e33  4c 89 ea                    mov rdx, r13
00000000:00401e36  e8 05 64 00 00              call 0x408240
00000000:00401e3b  c6 03 3d                    mov byte [rbx], 0x3d
00000000:00401e3e  31 d2                       xor edx, edx
```

```
Registers
RAX 0000000000000033
RCX 0000000000002710
RDX 00007ffd3a5db340
RBX 0000000000000002
RSP 00007ffd3a5d9930
RBP 00000000004c2cd8
RSI 0000000000000033
RDI 00007ffd3a5dc4c4  ASCII "[redacted]
R8  00007ffd3a5dc4c4  ASCII "[redacted]
R9  2c9da35cb7f59a7f
R10 820a3f9e92362e70
R11 08bfa8c215fe634e
R12 00007ffd3a5db240
R13 00007f74e8693000
R14 00000000004c2d38
R15 00007ffd3a5db4f0

RIP 0000000000401df6  </root/samples_pack/sendmail>

C 0  ES 0000
P 1  CS 0033
A 0  SS 002b
Z 1  DS 0000
S 0  FS 0000 (00007f74e8b56740)
```

Figure. Environment variable match (redacted)

A Closer Look to the Stub

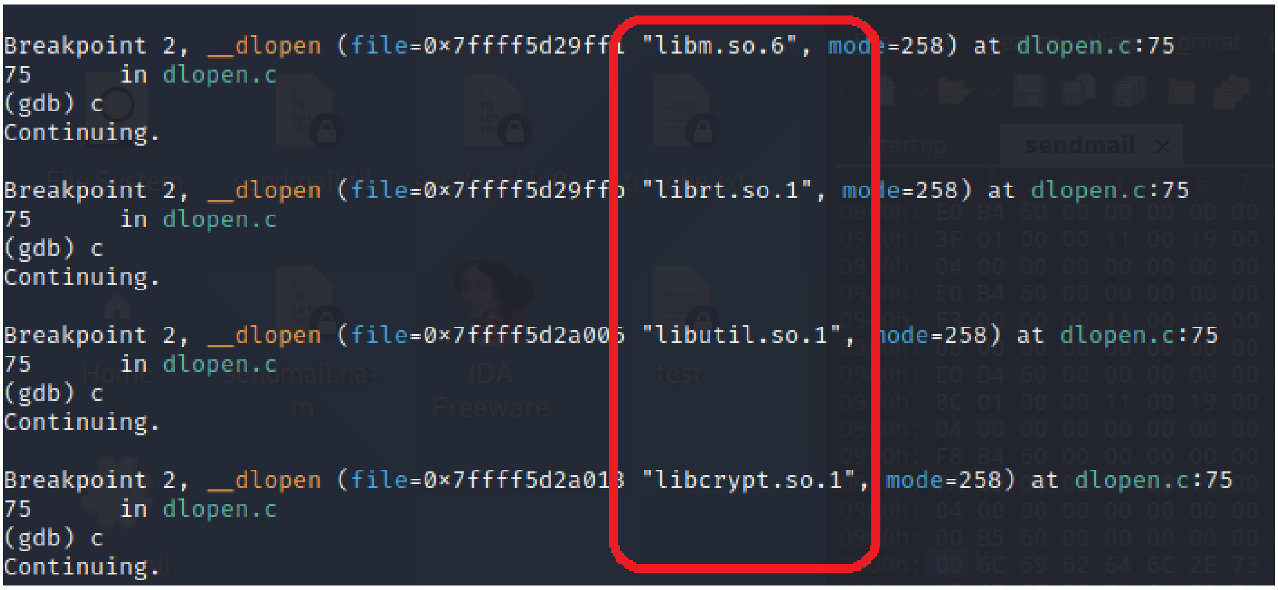
In addition, this packed ELF is matching some suspicious functions usually found in backdoors using the runtime linking techniques. Following are the functions with their relative offset:



```
ldpreload sendmail
0×a0d:$a: dlopen
0×a07:$b: dlsym
0×a3c:$c: fopen
0×ba4:$e: __fxstat
0×a0f:$j: open
0×a3d:$j: open
0×aec:$j: open
0×b9c:$l: __xstat
0×a69:$n: unlink
0×aec:$q: opendir
0×a34:$r: readdir
```

Figure. Packed EFL imports

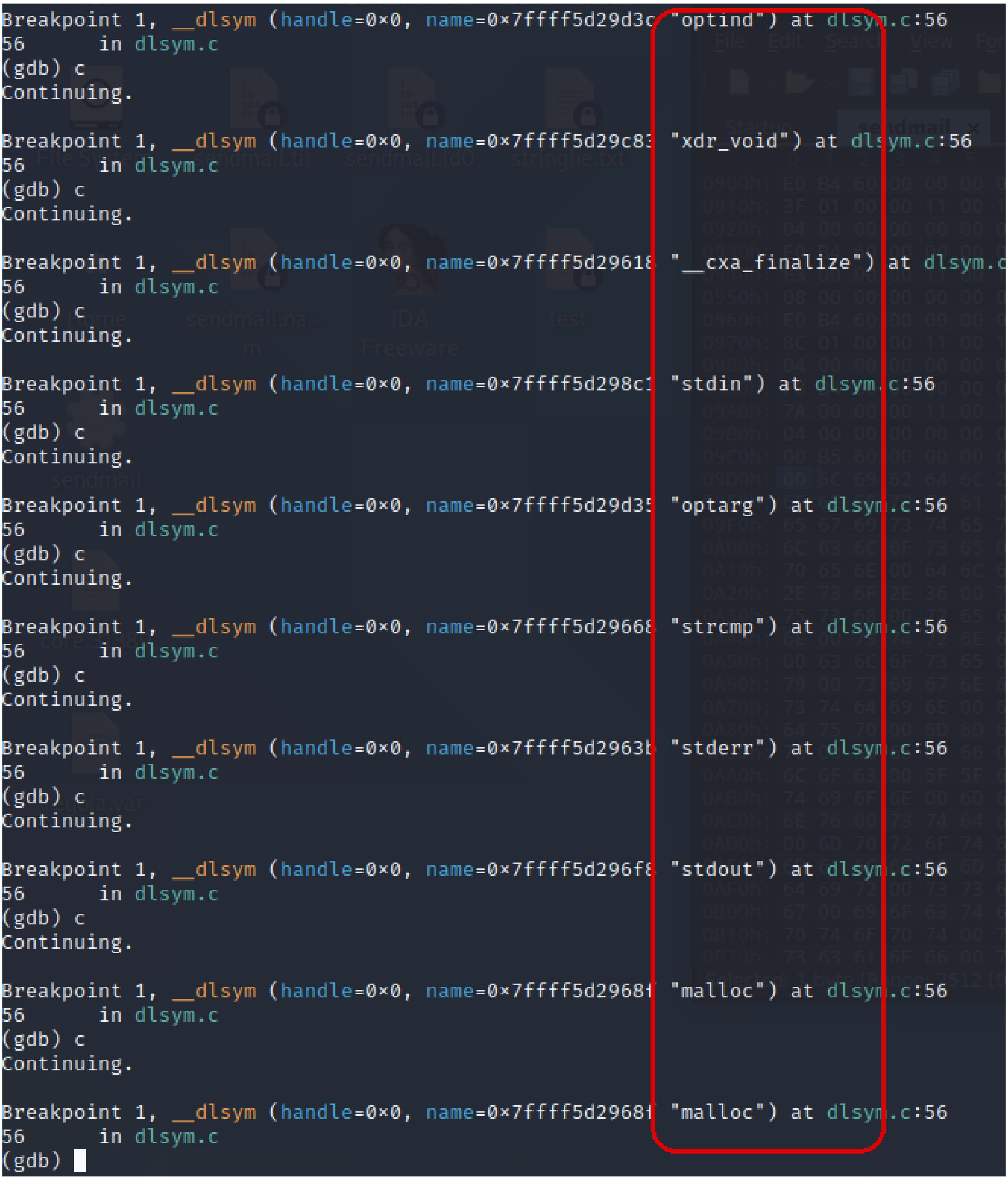
The presence of the *dlopen* and *dlsym* syscalls inside *libdl.so.2* is a clear indicator that this ELF uses a runtime linking mechanism by which hides all the dynamic symbols. The *dlopen()* function loads a shared object into the calling process's address space (the same of *LoadLibrary()* in Windows). The symbol resolution is done by the *dlsym()* syscall which returns the address of the first occurrence of the symbol. Setting a breakpoint on *dlopen()* we are able to know which libraries are loaded at runtime:



```
Breakpoint 2, __dlopen (file=0x7ffff5d29ff1 "libm.so.6", mode=258) at dlopen.c:75
75      in dlopen.c
(gdb) c
Continuing.

Breakpoint 2, __dlopen (file=0x7ffff5d29ffb "librt.so.1", mode=258) at dlopen.c:75
75      in dlopen.c
(gdb) c
Continuing.

Breakpoint 2, __dlopen (file=0x7ffff5d2a006 "libutil.so.1", mode=258) at dlopen.c:75
75      in dlopen.c
(gdb) c
Continuing.

Breakpoint 2, __dlopen (file=0x7ffff5d2a013 "libcrypt.so.1", mode=258) at dlopen.c:75
75      in dlopen.c
(gdb) c
Continuing.
```

Figure. Libraries dynamically loaded by the stub

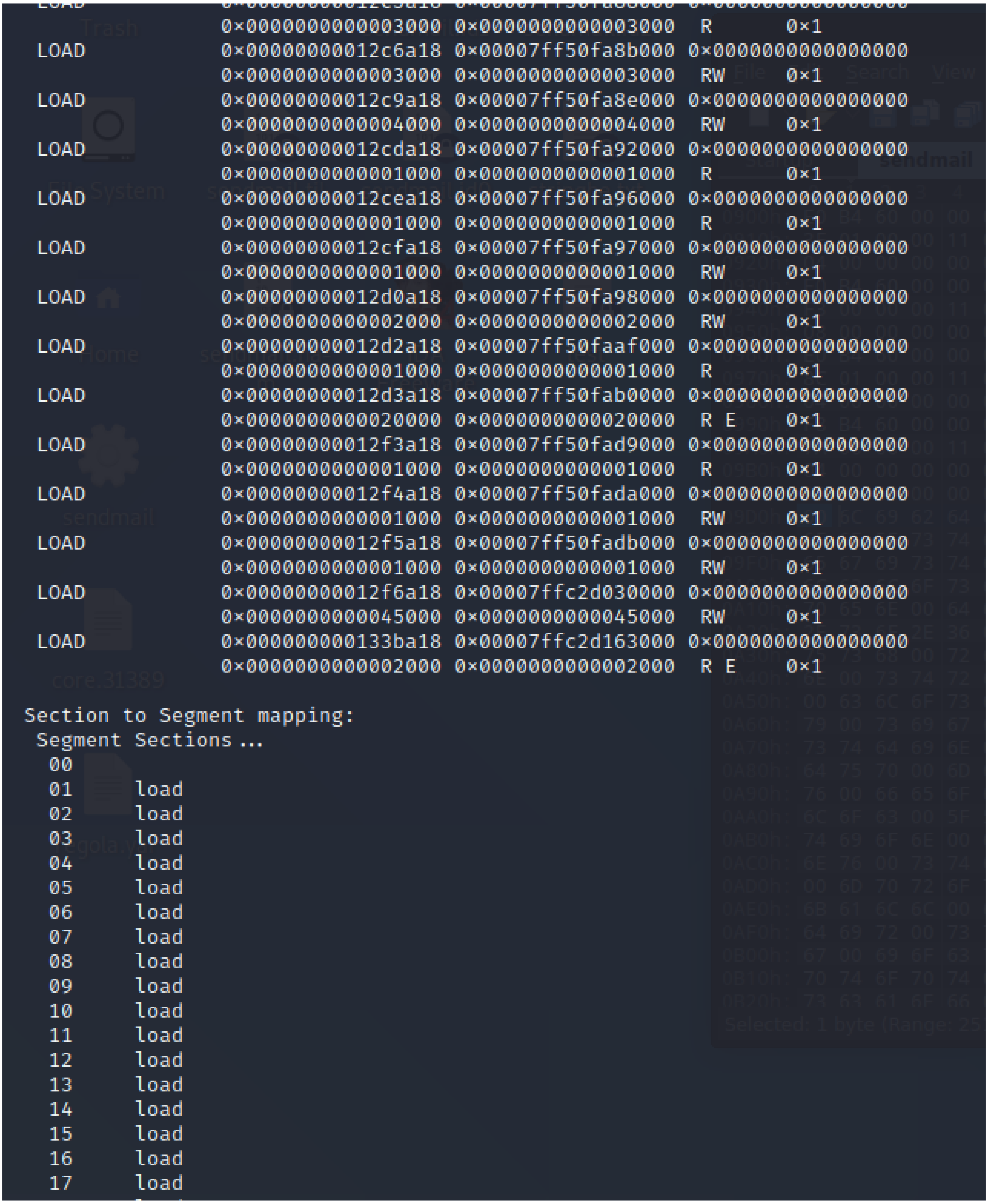
Then, in the same way we dump all the symbol resolved at runtime with the *dlsym()* syscall:



```
Breakpoint 1, __dlsym (handle=0x0, name=0x7ffff5d29d3c "optind") at dlsym.c:56
56      in dlsym.c
(gdb) c
Continuing.

Breakpoint 1, __dlsym (handle=0x0, name=0x7ffff5d29c83 "xdr_void") at dlsym.c:56
56      in dlsym.c
(gdb) c
Continuing.

Breakpoint 1, __dlsym (handle=0x0, name=0x7ffff5d29618 "__cxa_finalize") at dlsym.c
56      in dlsym.c
(gdb) c
Continuing.

Breakpoint 1, __dlsym (handle=0x0, name=0x7ffff5d298c1 "stdin") at dlsym.c:56
56      in dlsym.c
(gdb) c
Continuing.

Breakpoint 1, __dlsym (handle=0x0, name=0x7ffff5d29d35 "optarg") at dlsym.c:56
56      in dlsym.c
(gdb) c
Continuing.

Breakpoint 1, __dlsym (handle=0x0, name=0x7ffff5d29668 "strcmp") at dlsym.c:56
56      in dlsym.c
(gdb) c
Continuing.

Breakpoint 1, __dlsym (handle=0x0, name=0x7ffff5d2963b "stderr") at dlsym.c:56
56      in dlsym.c
(gdb) c
Continuing.

Breakpoint 1, __dlsym (handle=0x0, name=0x7ffff5d296f8 "stdout") at dlsym.c:56
56      in dlsym.c
(gdb) c
Continuing.

Breakpoint 1, __dlsym (handle=0x0, name=0x7ffff5d2968f "malloc") at dlsym.c:56
56      in dlsym.c
(gdb) c
Continuing.

Breakpoint 1, __dlsym (handle=0x0, name=0x7ffff5d2968f "malloc") at dlsym.c:56
56      in dlsym.c
(gdb) 
```

Figure. Syscall invoked during the unpacking

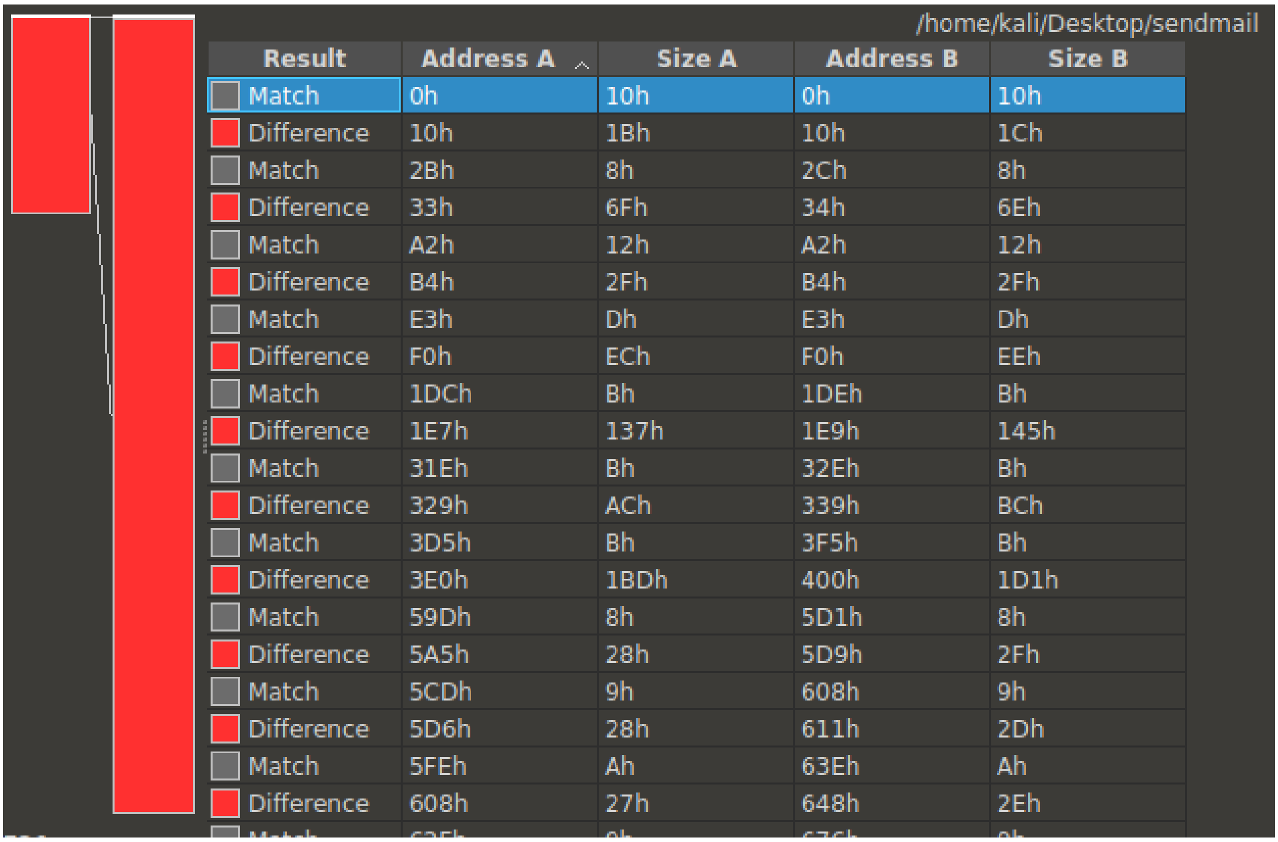
Inspecting the new unpacked memory,we immediately noticed its structure with all the program headers and section headers, then we found all the loaded new segments mapped into Virtual Memory at specific offset:



```
                 0x0000000000003000 0x0000000000003000  R      0x1
  LOAD           0x00000000012c6a18 0x00007ff50fa8b000 0x0000000000000000
                 0x0000000000003000 0x0000000000003000  RW     0x1
  LOAD           0x00000000012c9a18 0x00007ff50fa8e000 0x0000000000000000
                 0x0000000000004000 0x0000000000004000  RW     0x1
  LOAD           0x00000000012cda18 0x00007ff50fa92000 0x0000000000000000
                 0x0000000000001000 0x0000000000001000  R      0x1
  LOAD           0x00000000012cea18 0x00007ff50fa96000 0x0000000000000000
                 0x0000000000001000 0x0000000000001000  R      0x1
  LOAD           0x00000000012cfa18 0x00007ff50fa97000 0x0000000000000000
                 0x0000000000001000 0x0000000000001000  RW     0x1
  LOAD           0x00000000012d0a18 0x00007ff50fa98000 0x0000000000000000
                 0x0000000000002000 0x0000000000002000  RW     0x1
  LOAD           0x00000000012d2a18 0x00007ff50faaf000 0x0000000000000000
                 0x0000000000001000 0x0000000000001000  R      0x1
  LOAD           0x00000000012d3a18 0x00007ff50fab0000 0x0000000000000000
                 0x0000000000020000 0x0000000000020000  R E    0x1
  LOAD           0x00000000012f3a18 0x00007ff50fad9000 0x0000000000000000
                 0x0000000000001000 0x0000000000001000  R      0x1
  LOAD           0x00000000012f4a18 0x00007ff50fada000 0x0000000000000000
                 0x0000000000001000 0x0000000000001000  RW     0x1
  LOAD           0x00000000012f5a18 0x00007ff50fadb000 0x0000000000000000
                 0x0000000000001000 0x0000000000001000  RW     0x1
  LOAD           0x00000000012f6a18 0x00007ffc2d030000 0x0000000000000000
                 0x0000000000045000 0x0000000000045000  RW     0x1
  LOAD           0x000000000133ba18 0x00007ffc2d163000 0x0000000000000000
                 0x0000000000002000 0x0000000000002000  R E    0x1

 Section to Segment mapping:
  Segment Sections...
   00
   01     load
   02     load
   03     load
   04     load
   05     load
   06     load
   07     load
   08     load
   09     load
   10     load
   11     load
   12     load
   13     load
   14     load
   15     load
   16     load
   17     load
```

Figure. Unpacked memory sections

These LOADsegments contain unpacked payload: it has different size than and the number of program-headers and section-headers are also different. The unpacked version have a lot of clear-text LOAD sections that was previously unpacked from memory, the following image summarize the unpacked memory regions (the bar on the right):

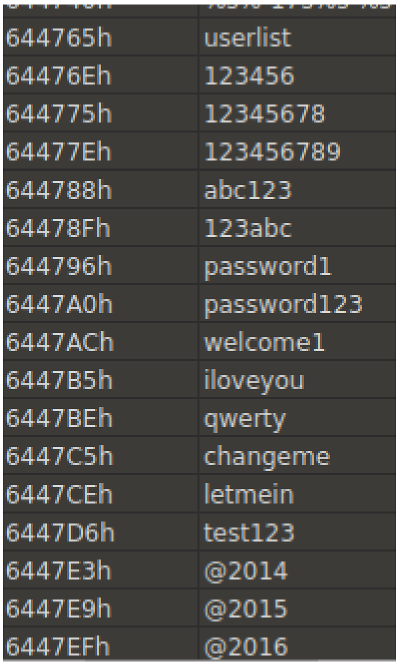


/home/kali/Desktop/sendmail

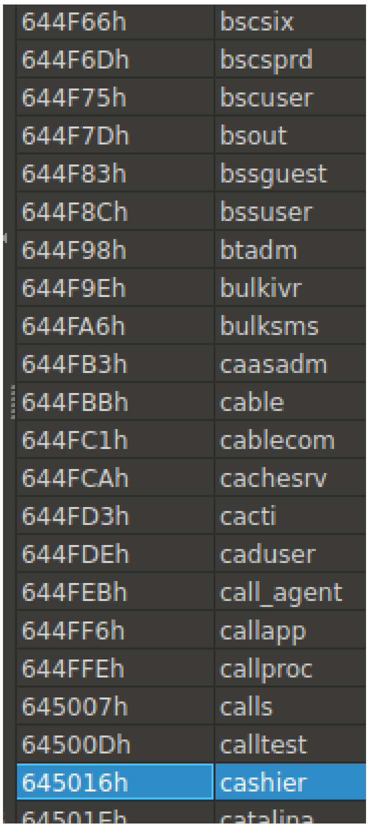
Result	Address A	Size A	Address B	Size B
⬜ Match	0h	10h	0h	10h
🟥 Difference	10h	1Bh	10h	1Ch
⬜ Match	2Bh	8h	2Ch	8h
🟥 Difference	33h	6Fh	34h	6Eh
⬜ Match	A2h	12h	A2h	12h
🟥 Difference	B4h	2Fh	B4h	2Fh
⬜ Match	E3h	Dh	E3h	Dh
🟥 Difference	F0h	ECh	F0h	EEh
⬜ Match	1DCh	Bh	1DEh	Bh
🟥 Difference	1E7h	137h	1E9h	145h
⬜ Match	31Eh	Bh	32Eh	Bh
🟥 Difference	329h	ACh	339h	BCh
⬜ Match	3D5h	Bh	3F5h	Bh
🟥 Difference	3E0h	1BDh	400h	1D1h
⬜ Match	59Dh	8h	5D1h	8h
🟥 Difference	5A5h	28h	5D9h	2Fh
⬜ Match	5CDh	9h	608h	9h
🟥 Difference	5D6h	28h	611h	2Dh
⬜ Match	5FEh	Ah	63Eh	Ah
🟥 Difference	608h	27h	648h	2Eh

Figure. Segment difference

Inspecting all these unpacked regions (in red), we found some dictionaries used by the backdoor for enumeration or brute force. This is very interesting because it shows us the real capabilities and the magnitude of this Kill Chain. More details in the following sections.



644765h	userlist
64476Eh	123456
644775h	12345678
64477Eh	123456789
644788h	abc123
64478Fh	123abc
644796h	password1
6447A0h	password123
6447ACh	welcome1
6447B5h	iloveyou
6447BEh	qwerty
6447C5h	changeme
6447CEh	letmein
6447D6h	test123
6447E3h	@2014
6447E9h	@2015
6447EFh	@2016

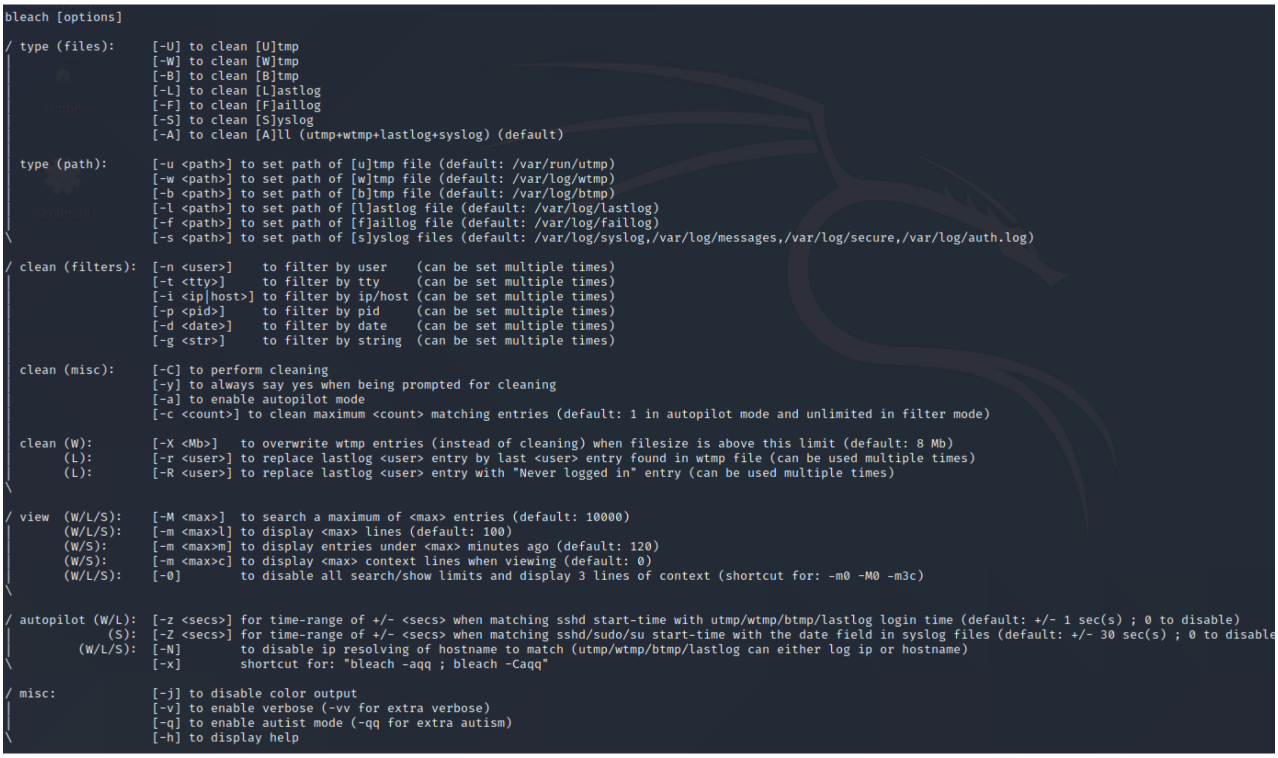


644F66h	bscsix
644F6Dh	bscsprd
644F75h	bscuser
644F7Dh	bsout
644F83h	bssguest
644F8Ch	bssuser
644F98h	btadm
644F9Eh	bulkivr
644FA6h	bulksms
644FB3h	caasadm
644FBBh	cable
644FC1h	cablecom
644FCAh	cachesrv
644FD3h	cacti
644FDEh	caduser
644FEBh	call_agent
644FF6h	callapp
644FFEh	callproc
645007h	calls
64500Dh	calltest
645016h	cashier
64501Eh	catalina

Figure. Wordlists and dictionaries inside the ELF binary

The APT Swiss Army Knife

At this point of the analysis, we want to provide an overview of the capabilities of this malware sample. It is a complete toolset for reconnaissance, lateral movement, exploitation and post exploitation activities. When the toolset is launched, it shows the complete menu with all the possible commands.



```
bleach [options]

/ type (files):     [-U] to clean [U]tmp
|                   [-W] to clean [W]tmp
|                   [-B] to clean [B]tmp
|                   [-L] to clean [L]astlog
|                   [-F] to clean [F]aillog
|                   [-S] to clean [S]yslog
|                   [-A] to clean [A]ll (utmp+wtmp+lastlog+syslog) (default)
|
| type (path):      [-u <path>] to set path of [u]tmp file (default: /var/run/utmp)
|                   [-w <path>] to set path of [w]tmp file (default: /var/log/wtmp)
|                   [-b <path>] to set path of [b]tmp file (default: /var/log/btmp)
|                   [-l <path>] to set path of [l]astlog file (default: /var/log/lastlog)
|                   [-f <path>] to set path of [f]aillog file (default: /var/log/faillog)
\                   [-s <path>] to set path of [s]yslog files (default: /var/log/syslog,/var/log/messages,/var/log/secure,/var/log/auth.log)

/ clean (filters):  [-n <user>]    to filter by user    (can be set multiple times)
|                   [-t <tty>]     to filter by tty     (can be set multiple times)
|                   [-i <ip|host>] to filter by ip/host (can be set multiple times)
|                   [-p <pid>]     to filter by pid     (can be set multiple times)
|                   [-d <date>]    to filter by date    (can be set multiple times)
|                   [-g <str>]     to filter by string  (can be set multiple times)
|
| clean (misc):     [-C] to perform cleaning
|                   [-y] to always say yes when being prompted for cleaning
|                   [-a] to enable autopilot mode
|                   [-c <count>] to clean maximum <count> matching entries (default: 1 in autopilot mode and unlimited in filter mode)
|
| clean (W):        [-X <Mb>]   to overwrite wtmp entries (instead of cleaning) when filesize is above this limit (default: 8 Mb)
|       (L):        [-r <user>] to replace lastlog <user> entry by last <user> entry found in wtmp file (can be used multiple times)
|       (L):        [-R <user>] to replace lastlog <user> entry with "Never logged in" entry (can be used multiple times)
\

/ view  (W/L/S):    [-M <max>]  to search a maximum of <max> entries (default: 10000)
|       (W/L/S):    [-m <max>l] to display <max> lines (default: 100)
|       (W/S):      [-m <max>m] to display entries under <max> minutes ago (default: 120)
|       (W/S):      [-m <max>c] to display <max> context lines when viewing (default: 0)
|       (W/L/S):    [-0]        to disable all search/show limits and display 3 lines of context (shortcut for: -m0 -M0 -m3c)
\

/ autopilot (W/L):  [-z <secs>] for time-range of +/- <secs> when matching sshd start-time with utmp/wtmp/btmp/lastlog login time (default: +/- 1 sec(s) ; 0 to disable)
|             (S):  [-Z <secs>] for time-range of +/- <secs> when matching sshd/sudo/su start-time with the date field in syslog files (default: +/- 30 sec(s) ; 0 to disable
|         (W/L/S):  [-N]        to disable ip resolving of hostname to match (utmp/wtmp/btmp/lastlog can either log ip or hostname)
\                   [-x]        shortcut for: "bleach -aqq ; bleach -Caqq"

/ misc:             [-j] to disable color output
|                   [-v] to enable verbose (-vv for extra verbose)
|                   [-q] to enable autist mode (-qq for extra autism)
\                   [-h] to display help
```

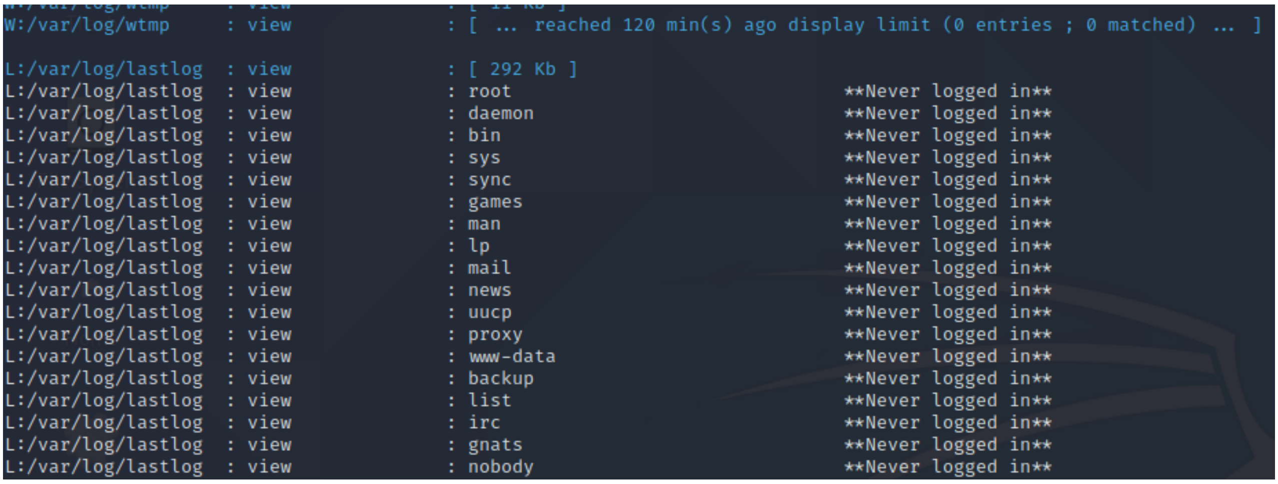
Figure. Malware tool help

One of the sneakiest commands we noticed is the “bleach” one, able to delete all btmp wtmp and btmp logs. The btmp log keeps track of failed login attempts; wtmp gives historical data of utmp and btmp provides the complete picture of users logins at which terminals, logouts, system events and current status of the system, system boot time (used by uptime) etc. It is also able to clean Syslog logs in /var/log/syslog, /var/log/messages, /var/log/secure and /var/log/auth.log or optionally all of them with the “-A” flag (utmp+wtmp+lastlog+syslog) which is the default.

There is also the possibility to apply the so-called “Clean Filters” to clean logs for specific users or ip or according to date etc.

```
 clean (filters):  [-n <user>]    to filter by user    (can be set multiple times)
|                   [-t <tty>]     to filter by tty     (can be set multiple times)
|                   [-i <ip|host>] to filter by ip/host (can be set multiple times)
|                   [-p <pid>]     to filter by pid     (can be set multiple times)
|                   [-d <date>]    to filter by date    (can be set multiple times)
|                   [-g <str>]     to filter by string  (can be set multiple times
```

Is clear that the usage of the “bleach” parameter during an intrusion results in hard times for the DFIR team.



```
W:/var/log/wtmp     : view                 : [ 11 Kb ]
W:/var/log/wtmp     : view                 : [  ...  reached 120 min(s) ago display limit (0 entries ; 0 matched)  ...  ]

L:/var/log/lastlog  : view                 : [ 292 Kb ]
L:/var/log/lastlog  : view                 : root                          **Never logged in**
L:/var/log/lastlog  : view                 : daemon                        **Never logged in**
L:/var/log/lastlog  : view                 : bin                           **Never logged in**
L:/var/log/lastlog  : view                 : sys                           **Never logged in**
L:/var/log/lastlog  : view                 : sync                          **Never logged in**
L:/var/log/lastlog  : view                 : games                         **Never logged in**
L:/var/log/lastlog  : view                 : man                           **Never logged in**
L:/var/log/lastlog  : view                 : lp                            **Never logged in**
L:/var/log/lastlog  : view                 : mail                          **Never logged in**
L:/var/log/lastlog  : view                 : news                          **Never logged in**
L:/var/log/lastlog  : view                 : uucp                          **Never logged in**
L:/var/log/lastlog  : view                 : proxy                         **Never logged in**
L:/var/log/lastlog  : view                 : www-data                      **Never logged in**
L:/var/log/lastlog  : view                 : backup                        **Never logged in**
L:/var/log/lastlog  : view                 : list                          **Never logged in**
L:/var/log/lastlog  : view                 : irc                           **Never logged in**
L:/var/log/lastlog  : view                 : gnats                         **Never logged in**
L:/var/log/lastlog  : view                 : nobody                        **Never logged in**
```

Figure. Bleach parameter execution

However the functionalities and tools embedded in this ELF binary are really wide and this is exactly why we referenced the tool as an APT swiss army knife. Here we sum up a list of the most interesting ones among the enlisting of all the available commands.

```
sendmail [ sun4me | demo | unixcat | nc110 | netcat | netcat-ssl | telnet |
traceroute | traceroute-tcp | traceroute-tcpfin | traceroute-udp | traceroute-icmp
| traceroute-all | sctpscan | sdporn | onesixtyone | snmpgrab | tftpd | ciscopush
| ciscown | ciscomg | HEAD | GET | ssleak | rmiexec | pogo | pogo2 | elogic | Cmd
| backfire | netbackup | netrider | sniff | bleach | nfsshell | mikrotik-client |
sid-force | ssh-user | sshock | ssh | arpmap | ricochet | mac2vendor | ip2country
| ipgen | ipsort | ipcalc | range2class | crunch | words.pl | passgen | passcheck
| getpass | decrypt-cisco | decrypt-vnc | decrypt-cvs | wmon | pmon | lemon | pty
| exec | nsexec | nsexec2 | setns | dumpkcore | dumpmem | pcregrep | xxd | strings
| sstrip | shred | md5sum | sha1sum | sha256sum | compress | uncompress | encrypt
| decrypt | uuencode | uudecode | base64 | whois | whob | resolv | ahost | adig |
axfr | asrv | aspf | periscope | scanip.sh | aliveips.sh | brutus.pl |
enum4linux.pl | snmpcheck.pl | = | _ | . | -? ] [options] [args]
```

```
sendmail [ s4m | demo | ucat | nc110 | nc | ncs | tel | tr | trt | trf | tru | tri
| tra | sctp | sd | sn | sg | tf | ccp | cco | ccg | HEAD | GET | ssleak | rmiexec
| pogo | pogo2 | el | Cmd | bf | nb | nr | sni | clean | nfs | mikro | sid | sshu
| ss | ssh | arp | rick | mac | ip2c | ipg | ips | ipc | r2c | crunch | words | lp
| pcheck | gpass | dec-cisco | dec-vnc | dec-cvs | wmon | pmon | emon | pty | exec
| nsexec | nsexec2 | setns | kcore | dmem | grep | xxd | str | strip | srm | md5 |
sha1 | sha256 | comp | uncomp | enc | dec | uue | uud | b64 | whois | whob | res |
host | dig | axfr | asrv | aspf | scope | scanip | aliveips | brutus | e4l |
snmpcheck | = | _ | . | ? ] [options] [args]
```

The amount of available commands is simply impressive: some are known system utilities, some others are offensive scripts, other ones known hacking tools and other ones mysterious, custom commands.To sum up, we noticed at least four categories of tools embedded in this single ELF binary:

- **Network and Enumeration Tools** such asnetcat, unixcat, netcat-ssl, telnet, traceroute, traceroute-tcp, traceroute-tcpfin, traceroute-udp, traceroute-icmp | traceroute-all, tftpd, HEAD, GET, sniff, nfsshell, ssh, ricochet,axfr, ,whois, scanip, sctpscan, sdporn, rmiexec, arpmap, whois, who, ahost, resolv, adig, axfr, asrv, aspf, periscope, scanip.sh, aliveips.sh, brutus.pl, enum4linux.pl, mikro, ss, sshu, onesixtyone, snmpgrab, snmpcheck, ciscopush, mikrotik-client.

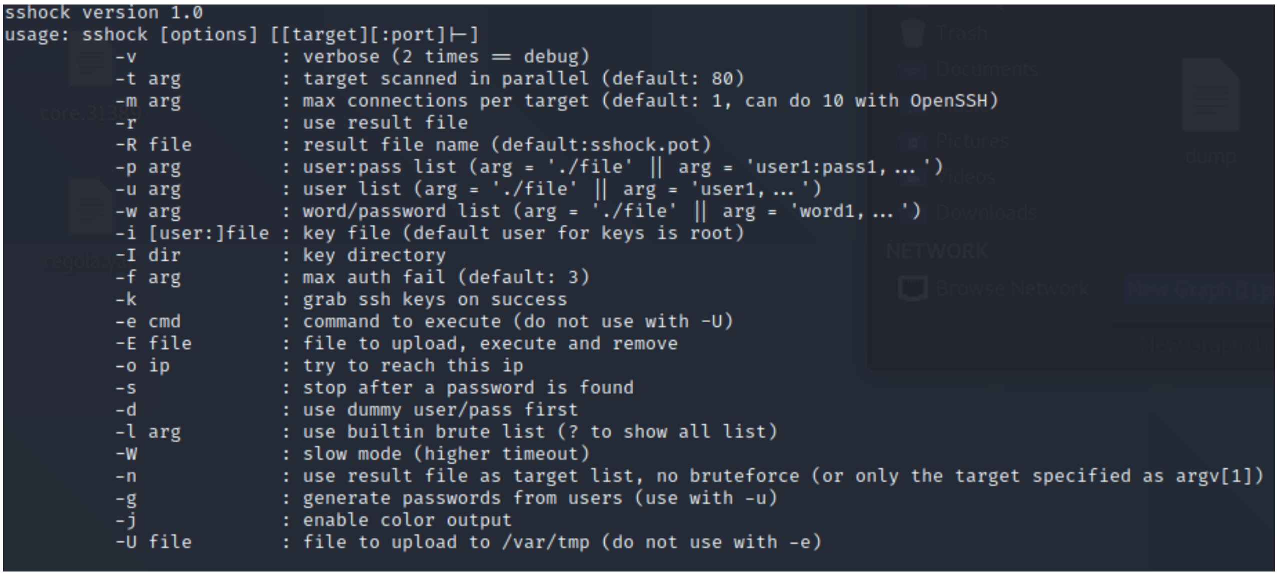
- **Anti-Forensics** tools such asbleach, clean.
- **System Utilities** such asmd5, sha1, mac2vendor, xxd, cmd, netbackup, ip2country, ipgen, ipsort, ipcalc, range2class, crunch, words.pl, passgen, passcheck, getpass, wmon, pmon, pty, exec, nsexec, nsexec2, setns, dumpkcore, dumpmem, pcregrep, strings, sstrip, shred, md5sum, sha1sum, sha256sum, compress, uncompress, encrypt, decrypt, uuencode , uudecode, base64.
- **Escalation and Exploitation** tools like ssleak, decrypt-vpn, pogo, pogo2, sid-force, sshock, decrypt-cisco, decrypt-vnc, decrypt-cvs.

There are tools for enumeration such as arp, dns, active directory, whois, ip enumeration and so on, some network tools and utilities for supporting exploiting and enumerations operations, also some exploitation and decryption tools specifically for CISCO, VNC, CVS and Mikrotik systems.

But some of them require a little deep dive.

SShock

SShock is a tool used to bruteforce SSH logins. In fact it is possible to specify an user list *(-u arg*) and a password list (*-p arg*), as shown in the following figure:



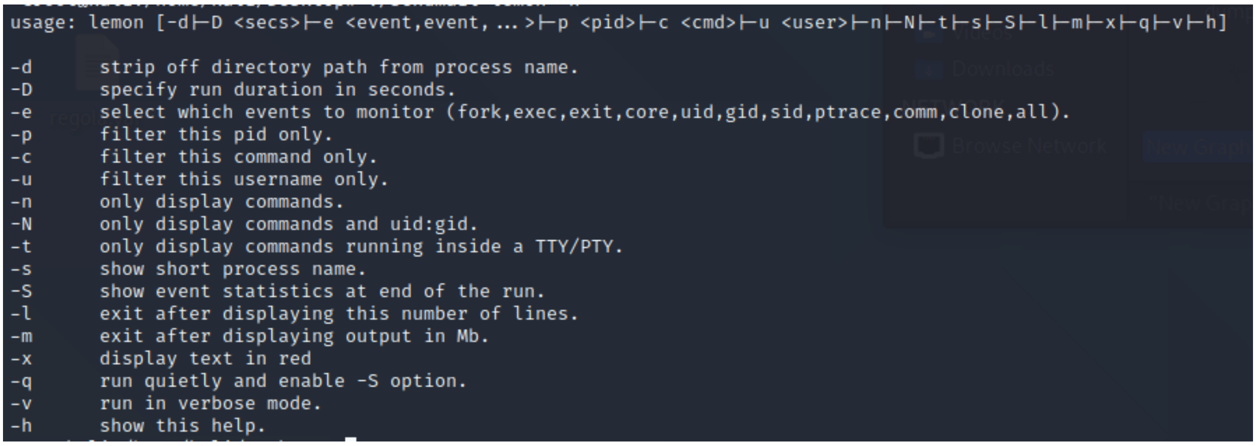
```
sshock version 1.0
usage: sshock [options] [[target][:port]|-]
          -v               : verbose (2 times == debug)
          -t arg           : target scanned in parallel (default: 80)
          -m arg           : max connections per target (default: 1, can do 10 with OpenSSH)
          -r               : use result file
          -R file          : result file name (default:sshock.pot)
          -p arg           : user:pass list (arg = './file' || arg = 'user1:pass1, ... ')
          -u arg           : user list (arg = './file' || arg = 'user1, ... ')
          -w arg           : word/password list (arg = './file' || arg = 'word1, ... ')
          -i [user:]file : key file (default user for keys is root)
          -I dir           : key directory
          -f arg           : max auth fail (default: 3)
          -k               : grab ssh keys on success
          -e cmd           : command to execute (do not use with -U)
          -E file          : file to upload, execute and remove
          -o ip            : try to reach this ip
          -s               : stop after a password is found
          -d               : use dummy user/pass first
          -l arg           : use builtin brute list (? to show all list)
          -W               : slow mode (higher timeout)
          -n               : use result file as target list, no bruteforce (or only the target specified as argv[1])
          -g               : generate passwords from users (use with -u)
          -j               : enable color output
          -U file          : file to upload to /var/tmp (do not use with -e)
```

Figure. SShock help file

Another interesting thing of the tool is the possibility (with the *-E* flag) to specify some input file to upload and execute which will then be removed.

Lemon

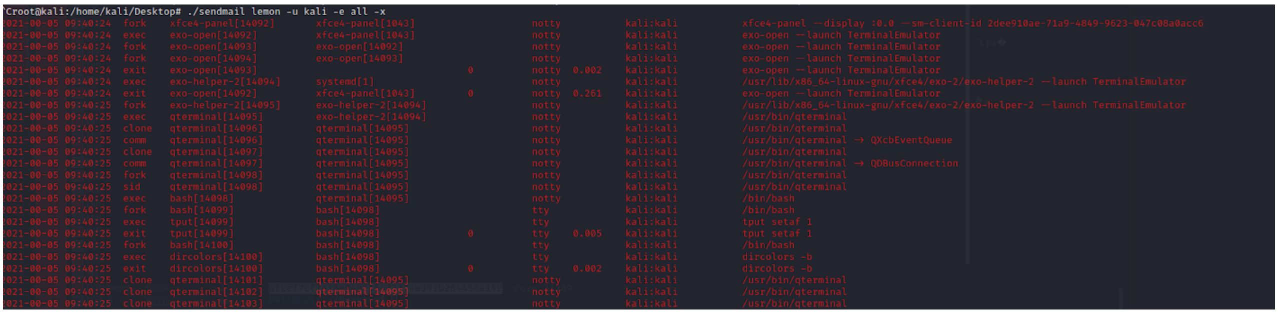
Lemon is a very powerful monitoring utility which is capable of monitoring all system events such as (fork, exec, exit, core etc) of specific processes or users. All monitored events could be filtered with specific switches *(-p, -c, -u*). Below the tool’s help menu is show:



```
usage: lemon [-d|-D <secs>|-e <event,event, ... >|-p <pid>|-c <cmd>|-u <user>|-n|-N|-t|-s|-S|-l|-m|-x|-q|-v|-h]

-d      strip off directory path from process name.
-D      specify run duration in seconds.
-e      select which events to monitor (fork,exec,exit,core,uid,gid,sid,ptrace,comm,clone,all).
-p      filter this pid only.
-c      filter this command only.
-u      filter this username only.
-n      only display commands.
-N      only display commands and uid:gid.
-t      only display commands running inside a TTY/PTY.
-s      show short process name.
-S      show event statistics at end of the run.
-l      exit after displaying this number of lines.
-m      exit after displaying output in Mb.
-x      display text in red
-q      run quietly and enable -S option.
-v      run in verbose mode.
-h      show this help.
```

Figure. Lemon help file

For instance, it is possible to monitor all events related to specific user using the following switches lemon -u <username> -e all, in this case we monitor all system events related to kali user:



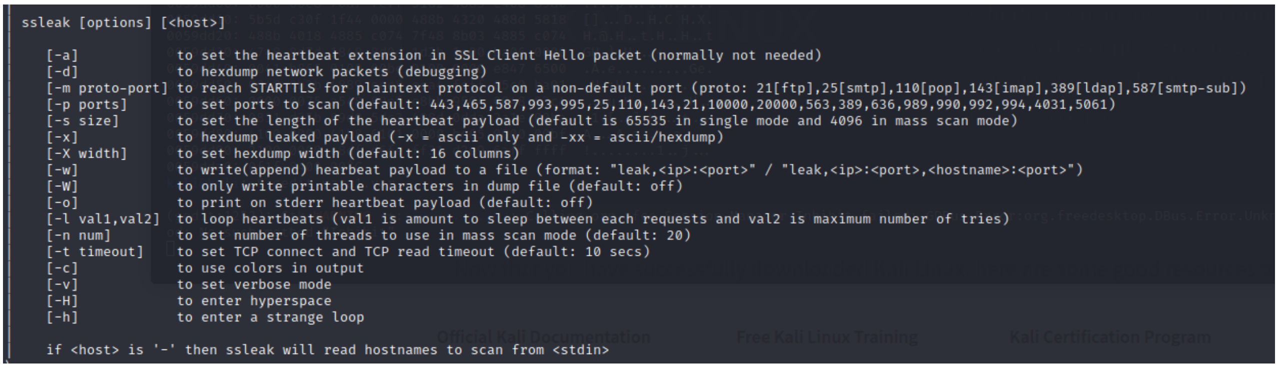
```
Croot@kali:/home/kali/Desktop# ./sendmail lemon -u kali -e all -x
2021-00-05 09:40:24  fork    xfce4-panel[14092]      xfce4-panel[1043]                    notty          kali:kali          xfce4-panel --display :0.0 --sm-client-id 2dee910ae-71a9-4849-9623-047c08a0acc6
2021-00-05 09:40:24  exec    exo-open[14092]         xfce4-panel[1043]                    notty          kali:kali          exo-open --launch TerminalEmulator
2021-00-05 09:40:24  fork    exo-open[14093]         exo-open[14092]                      notty          kali:kali          exo-open --launch TerminalEmulator
2021-00-05 09:40:24  fork    exo-open[14094]         exo-open[14093]                      notty          kali:kali          exo-open --launch TerminalEmulator
2021-00-05 09:40:24  exit    exo-open[14093]                                      0       notty  0.002   kali:kali          exo-open --launch TerminalEmulator
2021-00-05 09:40:24  exec    exo-helper-2[14094]     systemd[1]                           notty          kali:kali          /usr/lib/x86_64-linux-gnu/xfce4/exo-2/exo-helper-2 --launch TerminalEmulator
2021-00-05 09:40:24  exit    exo-open[14092]         xfce4-panel[1043]            0       notty  0.261   kali:kali          exo-open --launch TerminalEmulator
2021-00-05 09:40:25  fork    exo-helper-2[14095]     exo-helper-2[14094]                  notty          kali:kali          /usr/lib/x86_64-linux-gnu/xfce4/exo-2/exo-helper-2 --launch TerminalEmulator
2021-00-05 09:40:25  exec    qterminal[14095]        exo-helper-2[14094]                  notty          kali:kali          /usr/bin/qterminal
2021-00-05 09:40:25  clone   qterminal[14096]        qterminal[14095]                     notty          kali:kali          /usr/bin/qterminal
2021-00-05 09:40:25  comm    qterminal[14096]        qterminal[14095]                     notty          kali:kali          /usr/bin/qterminal -> QXcbEventQueue
2021-00-05 09:40:25  clone   qterminal[14097]        qterminal[14095]                     notty          kali:kali          /usr/bin/qterminal
2021-00-05 09:40:25  comm    qterminal[14097]        qterminal[14095]                     notty          kali:kali          /usr/bin/qterminal -> QDBusConnection
2021-00-05 09:40:25  fork    qterminal[14098]        qterminal[14095]                     notty          kali:kali          /usr/bin/qterminal
2021-00-05 09:40:25  sid     qterminal[14098]        qterminal[14095]                     notty          kali:kali          /usr/bin/qterminal
2021-00-05 09:40:25  exec    bash[14098]             qterminal[14095]                     notty          kali:kali          /bin/bash
2021-00-05 09:40:25  fork    bash[14099]             bash[14098]                          tty            kali:kali          /bin/bash
2021-00-05 09:40:25  exec    tput[14099]             bash[14098]                          tty            kali:kali          tput setaf 1
2021-00-05 09:40:25  exit    tput[14099]             bash[14098]                  0       tty    0.005   kali:kali          tput setaf 1
2021-00-05 09:40:25  fork    bash[14100]             bash[14098]                          tty            kali:kali          /bin/bash
2021-00-05 09:40:25  exec    dircolors[14100]        bash[14098]                          tty            kali:kali          dircolors -b
2021-00-05 09:40:25  exit    dircolors[14100]        bash[14098]                  0       tty    0.002   kali:kali          dircolors -b
2021-00-05 09:40:25  clone   qterminal[14101]        qterminal[14095]                     notty          kali:kali          /usr/bin/qterminal
2021-00-05 09:40:25  clone   qterminal[14102]        qterminal[14095]                     notty          kali:kali          /usr/bin/qterminal
2021-00-05 09:40:25  clone   qterminal[14103]        qterminal[14095]                     notty          kali:kali          /usr/bin/qterminal
```

Figure. Lemon test run

Using this tool it is possible to monitor and track specific user's activities on specific machines (or multiple machines) in order to spot the presence of specific users in some timeframe.

Ssleak

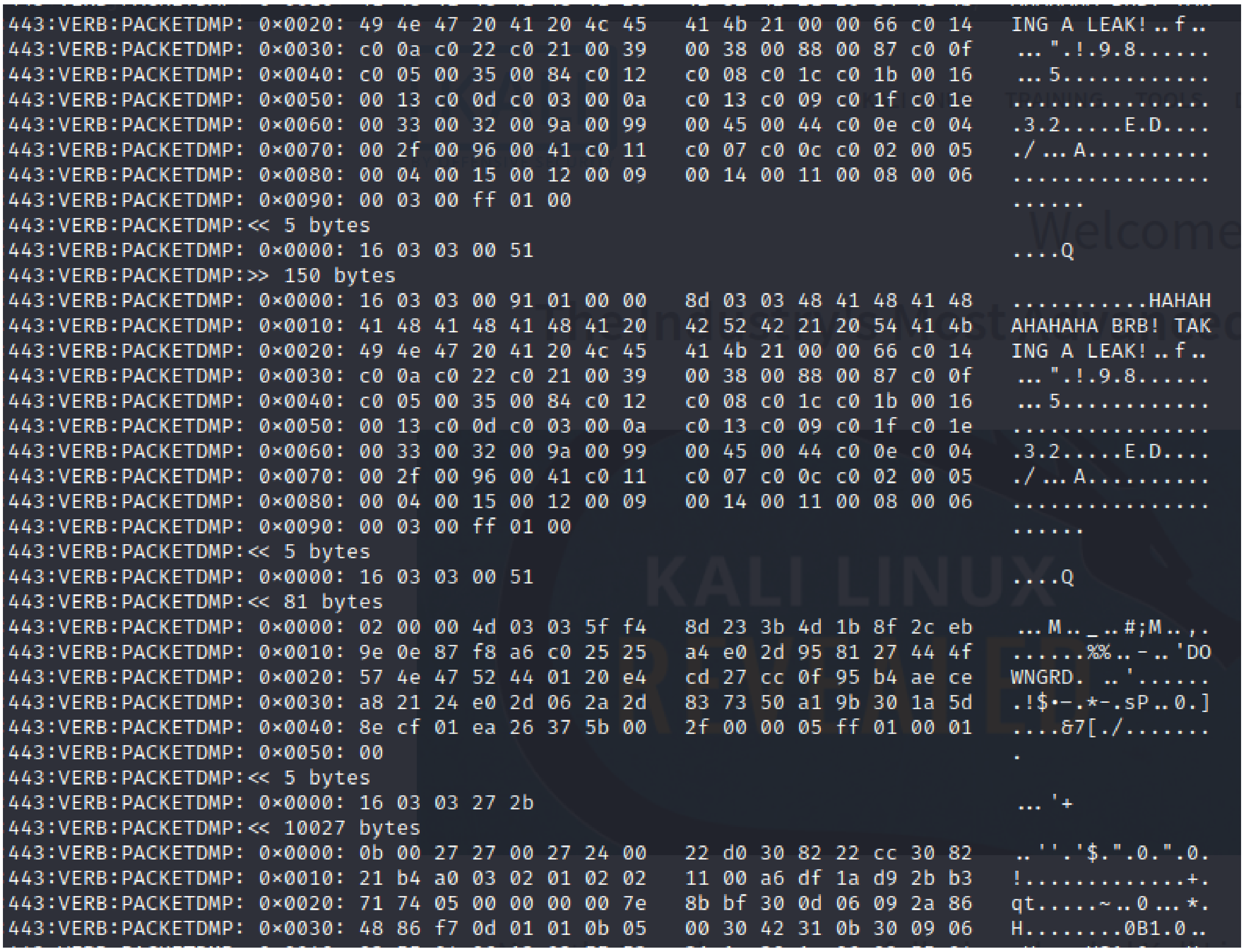
Ssleak is an utility to sniff SSL traffic. It is possible to specify a target and then dump all packets sent to and from in order to leak some information such as the server's certificate, server's canonical names etc.



```
ssleak [options] [<host>]

   [-a]           to set the heartbeat extension in SSL Client Hello packet (normally not needed)
   [-d]           to hexdump network packets (debugging)
   [-m proto-port] to reach STARTTLS for plaintext protocol on a non-default port (proto: 21[ftp],25[smtp],110[pop],143[imap],389[ldap],587[smtp-sub])
   [-p ports]     to set ports to scan (default: 443,465,587,993,995,25,110,143,21,10000,20000,563,389,636,989,990,992,994,4031,5061)
   [-s size]      to set the length of the heartbeat payload (default is 65535 in single mode and 4096 in mass scan mode)
   [-x]           to hexdump leaked payload (-x = ascii only and -xx = ascii/hexdump)
   [-X width]     to set hexdump width (default: 16 columns)
   [-w]           to write(append) hearbeat payload to a file (format: "leak,<ip>:<port>" / "leak,<ip>:<port>,<hostname>:<port>")
   [-W]           to only write printable characters in dump file (default: off)
   [-o]           to print on stderr heartbeat payload (default: off)
   [-l val1,val2] to loop heartbeats (val1 is amount to sleep between each requests and val2 is maximum number of tries)
   [-n num]       to set number of threads to use in mass scan mode (default: 20)
   [-t timeout]   to set TCP connect and TCP read timeout (default: 10 secs)
   [-c]           to use colors in output
   [-v]           to set verbose mode
   [-H]           to enter hyperspace
   [-h]           to enter a strange loop

   if <host> is '-' then ssleak will read hostnames to scan from <stdin>
```

Figure. SSLeak help file

Moreover it is also possible to exploit Heartbleed Vulnerability (CVE-2014-0160) with custom-forged heartbeat packets with a fake length with *-s* switch and print also the hexdump of such leak with *-x* switch.



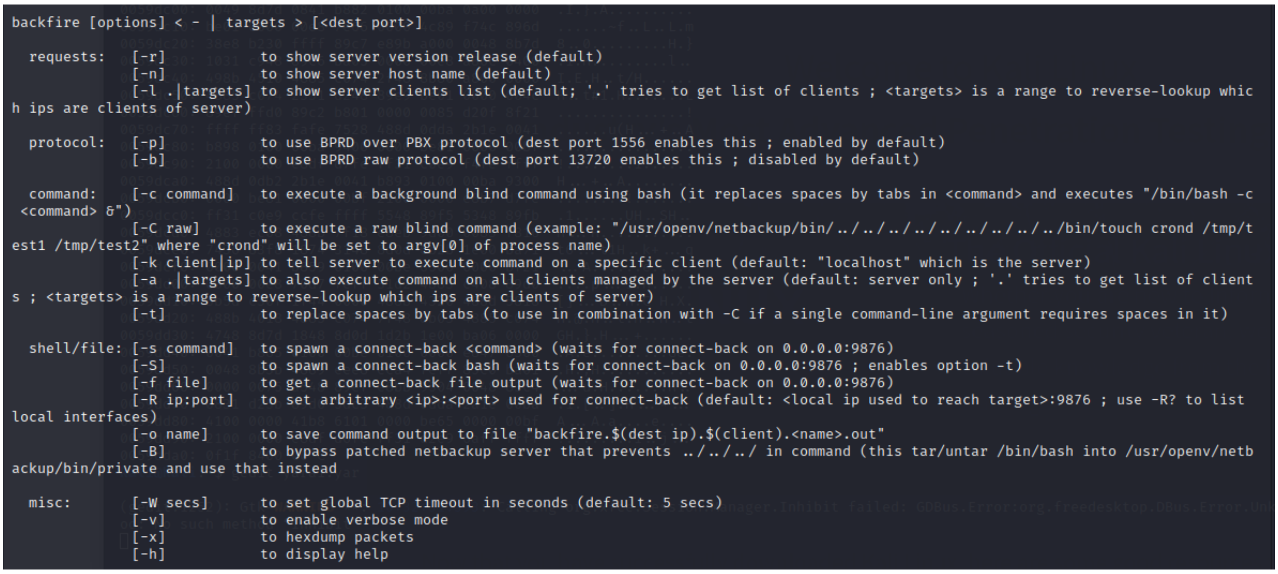
```
443:VERB:PACKETDMP: 0x0020: 49 4e 47 20 41 20 4c 45   41 4b 21 00 00 66 c0 14   ING A LEAK!..f..
443:VERB:PACKETDMP: 0x0030: c0 0a c0 22 c0 21 00 39   00 38 00 88 00 87 c0 0f   ...".!.9.8......
443:VERB:PACKETDMP: 0x0040: c0 05 00 35 00 84 c0 12   c0 08 c0 1c c0 1b 00 16   ...5............
443:VERB:PACKETDMP: 0x0050: 00 13 c0 0d c0 03 00 0a   c0 13 c0 09 c0 1f c0 1e   ................
443:VERB:PACKETDMP: 0x0060: 00 33 00 32 00 9a 00 99   00 45 00 44 c0 0e c0 04   .3.2.....E.D....
443:VERB:PACKETDMP: 0x0070: 00 2f 00 96 00 41 c0 11   c0 07 c0 0c c0 02 00 05   ./...A..........
443:VERB:PACKETDMP: 0x0080: 00 04 00 15 00 12 00 09   00 14 00 11 00 08 00 06   ................
443:VERB:PACKETDMP: 0x0090: 00 03 00 ff 01 00                                   ......
443:VERB:PACKETDMP:<< 5 bytes
443:VERB:PACKETDMP: 0x0000: 16 03 03 00 51                                      ....Q
443:VERB:PACKETDMP:>> 150 bytes
443:VERB:PACKETDMP: 0x0000: 16 03 03 00 91 01 00 00   8d 03 03 48 41 48 41 48   ...........HAHAH
443:VERB:PACKETDMP: 0x0010: 41 48 41 48 41 48 41 20   42 52 42 21 20 54 41 4b   AHAHAHA BRB! TAK
443:VERB:PACKETDMP: 0x0020: 49 4e 47 20 41 20 4c 45   41 4b 21 00 00 66 c0 14   ING A LEAK!..f..
443:VERB:PACKETDMP: 0x0030: c0 0a c0 22 c0 21 00 39   00 38 00 88 00 87 c0 0f   ...".!.9.8......
443:VERB:PACKETDMP: 0x0040: c0 05 00 35 00 84 c0 12   c0 08 c0 1c c0 1b 00 16   ...5............
443:VERB:PACKETDMP: 0x0050: 00 13 c0 0d c0 03 00 0a   c0 13 c0 09 c0 1f c0 1e   ................
443:VERB:PACKETDMP: 0x0060: 00 33 00 32 00 9a 00 99   00 45 00 44 c0 0e c0 04   .3.2.....E.D....
443:VERB:PACKETDMP: 0x0070: 00 2f 00 96 00 41 c0 11   c0 07 c0 0c c0 02 00 05   ./...A..........
443:VERB:PACKETDMP: 0x0080: 00 04 00 15 00 12 00 09   00 14 00 11 00 08 00 06   ................
443:VERB:PACKETDMP: 0x0090: 00 03 00 ff 01 00                                   ......
443:VERB:PACKETDMP:<< 5 bytes
443:VERB:PACKETDMP: 0x0000: 16 03 03 00 51                                      ....Q
443:VERB:PACKETDMP:<< 81 bytes
443:VERB:PACKETDMP: 0x0000: 02 00 00 4d 03 03 5f f4   8d 23 3b 4d 1b 8f 2c eb   ...M.._..#;M..,.
443:VERB:PACKETDMP: 0x0010: 9e 0e 87 f8 a6 c0 25 25   a4 e0 2d 95 81 27 44 4f   ......%%..-..'DO
443:VERB:PACKETDMP: 0x0020: 57 4e 47 52 44 01 20 e4   cd 27 cc 0f 95 b4 ae ce   WNGRD. ..'......
443:VERB:PACKETDMP: 0x0030: a8 21 24 e0 2d 06 2a 2d   83 73 50 a1 9b 30 1a 5d   .!$•-.*-.sP..0.]
443:VERB:PACKETDMP: 0x0040: 8e cf 01 ea 26 37 5b 00   2f 00 00 05 ff 01 00 01   ....&7[./.......
443:VERB:PACKETDMP: 0x0050: 00                                                  .
443:VERB:PACKETDMP:<< 5 bytes
443:VERB:PACKETDMP: 0x0000: 16 03 03 27 2b                                      ...'+
443:VERB:PACKETDMP:<< 10027 bytes
443:VERB:PACKETDMP: 0x0000: 0b 00 27 27 00 27 24 00   22 d0 30 82 22 cc 30 82   ..''.'$.".0.".0.
443:VERB:PACKETDMP: 0x0010: 21 b4 a0 03 02 01 02 02   11 00 a6 df 1a d9 2b b3   !.............+.
443:VERB:PACKETDMP: 0x0020: 71 74 05 00 00 00 00 7e   8b bf 30 0d 06 09 2a 86   qt.....~..0...*.
443:VERB:PACKETDMP: 0x0030: 48 86 f7 0d 01 01 0b 05   00 30 42 31 0b 30 09 06   H........0B1.0..
```

Figure. SSLeak test run

Backfire

Backfire is a tool used to establish and manage connect-back (or reverse) shells. A reverse shell permits to establish a connection between the compromised host (pivot) and the target machine when the target machine is not directly accessible for several reasons. For instance to perform maintenance tasks on hosts behind firewalls or NAT.

As, shown in the following screen, *backfire* provides the execution of such commands (*-c commands*) through a connect-back connection that is possible to spawn with -S flag or with -s <commands>

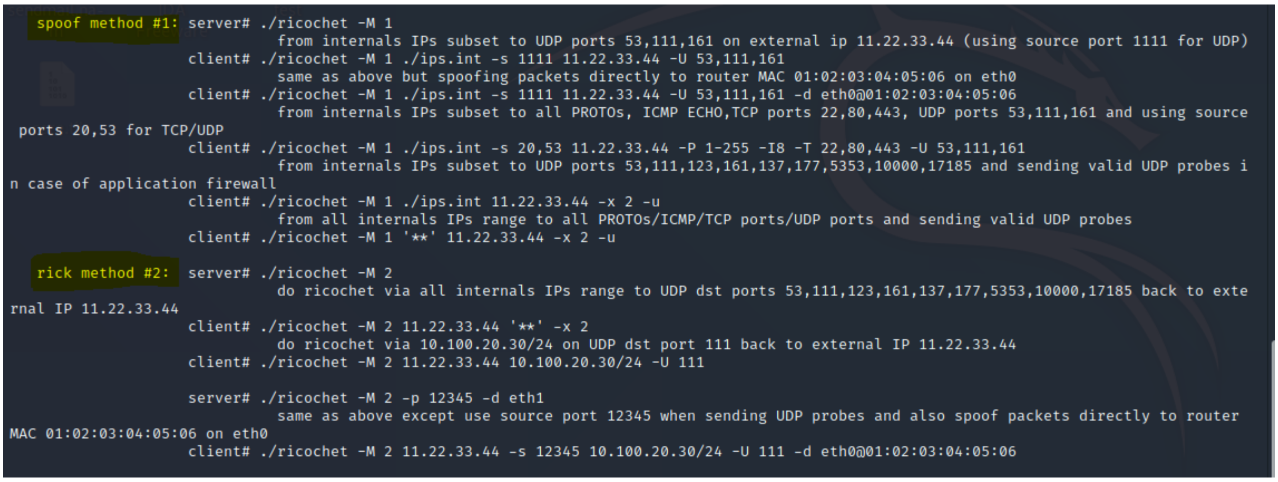


```
backfire [options] < - | targets > [<dest port>]

  requests:   [-r]           to show server version release (default)
              [-n]           to show server host name (default)
              [-l .|targets] to show server clients list (default; '.' tries to get list of clients ; <targets> is a range to reverse-lookup whic
h ips are clients of server)

  protocol:   [-p]           to use BPRD over PBX protocol (dest port 1556 enables this ; enabled by default)
              [-b]           to use BPRD raw protocol (dest port 13720 enables this ; disabled by default)

  command:    [-c command]   to execute a background blind command using bash (it replaces spaces by tabs in <command> and executes "/bin/bash -c
 <command> &")
              [-C raw]       to execute a raw blind command (example: "/usr/openv/netbackup/bin/../../../../../../../../../bin/touch crond /tmp/t
est1 /tmp/test2" where "crond" will be set to argv[0] of process name)
              [-k client|ip] to tell server to execute command on a specific client (default: "localhost" which is the server)
              [-a .|targets] to also execute command on all clients managed by the server (default: server only ; '.' tries to get list of client
s ; <targets> is a range to reverse-lookup which ips are clients of server)
              [-t]           to replace spaces by tabs (to use in combination with -C if a single command-line argument requires spaces in it)

  shell/file: [-s command]   to spawn a connect-back <command> (waits for connect-back on 0.0.0.0:9876)
              [-S]           to spawn a connect-back bash (waits for connect-back on 0.0.0.0:9876 ; enables option -t)
              [-f file]      to get a connect-back file output (waits for connect-back on 0.0.0.0:9876)
              [-R ip:port]   to set arbitrary <ip>:<port> used for connect-back (default: <local ip used to reach target>:9876 ; use -R? to list
local interfaces)
              [-o name]      to save command output to file "backfire.$(dest ip).$(client).<name>.out"
              [-B]           to bypass patched netbackup server that prevents ../../../ in command (this tar/untar /bin/bash into /usr/openv/netb
ackup/bin/private and use that instead

  misc:       [-W secs]      to set global TCP timeout in seconds (default: 5 secs)
              [-v]           to enable verbose mode
              [-x]           to hexdump packets
              [-h]           to display help
```

Figure. Backfire help file

Ricochet

Ricochet is a powerful utility for packet spoofing and FW ACL assessment. The tool can act as a client or a server. The client version permits to forge IP-PROTO/ICMP/UDP/TCP packets in order to test fw ACLs while the server is used to listen for replies coming from the firewall. It is possible to use 2 different methods. One is called *spoof (method #1) to spoof packets* and the other is *rick (method#2*) which stands for "ricochet" used also to spoof the address and port of the outgoing requests:



```
   spoof method #1: server# ./ricochet -M 1
                              from internals IPs subset to UDP ports 53,111,161 on external ip 11.22.33.44 (using source port 1111 for UDP)
                    client# ./ricochet -M 1 ./ips.int -s 1111 11.22.33.44 -U 53,111,161
                              same as above but spoofing packets directly to router MAC 01:02:03:04:05:06 on eth0
                    client# ./ricochet -M 1 ./ips.int -s 1111 11.22.33.44 -U 53,111,161 -d eth0@01:02:03:04:05:06
                              from internals IPs subset to all PROTOs, ICMP ECHO,TCP ports 22,80,443, UDP ports 53,111,161 and using source
 ports 20,53 for TCP/UDP
                    client# ./ricochet -M 1 ./ips.int -s 20,53 11.22.33.44 -P 1-255 -I8 -T 22,80,443 -U 53,111,161
                              from internals IPs subset to UDP ports 53,111,123,161,137,177,5353,10000,17185 and sending valid UDP probes i
n case of application firewall
                    client# ./ricochet -M 1 ./ips.int 11.22.33.44 -x 2 -u
                              from all internals IPs range to all PROTOs/ICMP/TCP ports/UDP ports and sending valid UDP probes
                    client# ./ricochet -M 1 '**' 11.22.33.44 -x 2 -u

   rick method #2:  server# ./ricochet -M 2
                              do ricochet via all internals IPs range to UDP dst ports 53,111,123,161,137,177,5353,10000,17185 back to exte
rnal IP 11.22.33.44
                    client# ./ricochet -M 2 11.22.33.44 '**' -x 2
                              do ricochet via 10.100.20.30/24 on UDP dst port 111 back to external IP 11.22.33.44
                    client# ./ricochet -M 2 11.22.33.44 10.100.20.30/24 -U 111

                    server# ./ricochet -M 2 -p 12345 -d eth1
                              same as above except use source port 12345 when sending UDP probes and also spoof packets directly to router
MAC 01:02:03:04:05:06 on eth0
                    client# ./ricochet -M 2 11.22.33.44 -s 12345 10.100.20.30/24 -U 111 -d eth0@01:02:03:04:05:06
```

Figure. Ricochet help file

Conclusion

The versatility of the "STEELCORGI" tool used by TH-239 is really impressive: all such capabilities embedded in a single, standalone, ready to deploy binary file, potentially enabling the attacker to establish a hidden communication channel, to recon internal network and to step in remote endpoint abusing various techniques. Also, this sort of "swiss army knife" was also heavily protected in a way that could be activated only during an actual intrusion, because the activation key is inoculated into the compromises system directly by the malicious operators, at run time.

All these facts are reminding us how dangerous and slimy an advanced intruder could sneak into the company network: tackling such kinds of threats requires advanced intelligence and analysis capabilities.

Appendix

Indicator of Compromise

Hash:

```
0845835e18a3ed4057498250d30a11b1
```

Yara:

```
rule ELF_packed_STEELCORGI_backdoor_UNC1945{
 meta:
   description = "Yara Rule for packed ELF backdoor of UNC1945"
   author = "Yoroi Malware Zlab"
   last_updated = "2020_12_21"
   tlp = "white"

   category = "informational"

strings:

$s1={4? 88 47 3c c1 6c ?4 34 08 8a 54 ?? ?? 4? 88 57 3d c1 6c}
$s2={0f b6 5? ?? 0f b6 4? ?? 4? c1 e2 18 4? c1 e0 10 4? }
$s3={8a 03 84 c0 74 ?? 3c 3d 75 ?? 3c 3d 75 ?? c6 03 00 4? 8b 7d 00}
$s4={01 c6 89 44 ?? ?? 8b 44 ?? ?? 31 f2 89 74 ?? ?? c1}
$s5={ 4? 89 d8 4? 31 f2 4? c1 e0 13 4? 01 d7 4? }

condition:
   uint32(0) == 0x464c457f and 3 of them
}


rule ELF_unpacked_STEELCORGI_backdoor_UNC1945{
 meta:
   description = "Yara Rule for unpacked ELF backdoor of UNC1945"
   author = "Yoroi Malware Zlab"
   last_updated = "2020_12_21"
   tlp = "white"
   category = "informational"

strings:
$s1="MCARC"
$s2="833fc0088ea41bc3331db60ae2.debug"
$s3="PORA1022"
$s4="server"
$s5="test"
$s6="no ejecutar git-update-server-info"
$s7="dlopen"
$s8="dlsym"
$s9="5d5c6da19e62263f67ca63f8bedeb6.debug"
$s10={72 69 6E 74 20 22 5B 56 5D 20 41 74 74 65 6D 70 74 69 6E 67 20 74 6F 20 67
65 74 20 4F 53 20 69 6E 66 6F 20 77 69 74 68 20 63 6F 6D 6D 61 6E 64 3A 20 24 63
6F 6D 6D 61 6E 64 5C 6E 22 20 69 66 20 24 76 65 72 62 6F 73 65 3B}

condition:
 all of them and #s4>50 and #s5>20
}
```

This blog post was authored by Luigi Martire, Antonio Pirozzi and Luca Mella of Yoroi Malware ZLAB