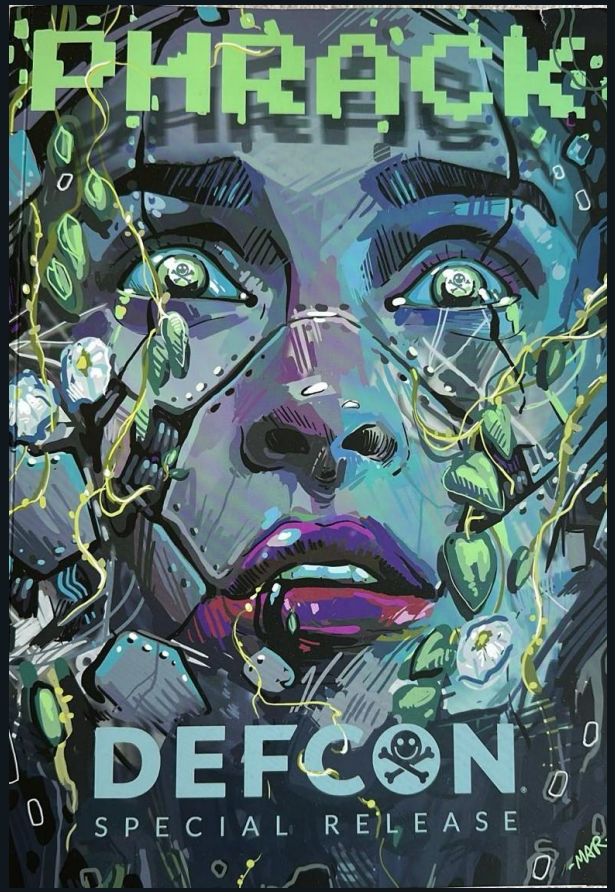
OCTOBER 9TH 2025

Sandfly Security®

Cold Incident Response Conference

Linux Stealth Rootkit Hunting


PHRACK
DEFCON
SPECIAL RELEASE

Common Stealth Rootkit Features

- Hides Module
- Hides Files
- Hides Directories
- Hides Processes
- Magic Packet Activation
- Evasive Backdoor

Stealth Rootkits are Juggling Hand Grenades

+ Can hide from casual detection
+ Can evade some focused detection efforts
+ Have stealthy backdoor capabilities

- Cause system stability impacts
- Fragile and tied to specific kernel versions
- Obvious once seen

Make Hiding Become a Liability

Principles of Linux Rootkit Hunting

Principles of Linux Rootkit Hunting

Data leaks

Inconsistent Answers

System Impacts

Locating Initialization Files

Module Initialization

```
root@sandflysecurity-victim:~/tomcat20250414_rootkit_linux2345/install# ./install.sh
Installing

.--ok--

root@sandflysecurity-victim:~/tomcat20250414_rootkit_linux2345/install#
```

Creates persistence and inserts module:

`/usr/lib64/tracker-fs` (module)
`/usr/include/tracker-fs/tracker-efs` (backdoor)
`/etc/init.d/tracker-fs`
`/etc/rc2.d/S55tracker-fs`
`/etc/rc3.d/S55tracker-fs`
`/etc/rc5.d/S55tracker-fs`

Locating Initialization Files

```
root@sandflysecurity-victim:~# systemctl status tracker-fs.service
● tracker-fs.service
     Loaded: loaded (/etc/init.d/tracker-fs; generated)
     Active: active (exited) since Tue 2025-09-30 02:35:51 UTC; 39min ago
       Docs: man:systemd-sysv-generator(8)
    Process: 691 ExecStart=/etc/init.d/tracker-fs start (code=exited, status=0/SUCCESS)
        CPU: 27ms

Sep 30 02:35:51 sandflysecurity-victim systemd[1]: Starting tracker-fs.service...
Sep 30 02:35:51 sandflysecurity-victim systemd[1]: Started tracker-fs.service.
root@sandflysecurity-victim:~#
```

```
systemctl list-units --state=exited
systemctl status tracker-fs.service
```

Ask `systemd` what init scripts it has run.

Leaks path: `/etc/init.d/tracker-fs`

Initialization File Investigation

```
root@sandfly-victim:/etc/init.d#
root@sandfly-victim:/etc/init.d# ls -al
total 124
drwxr-xr-x  2 root root 4096 Aug 13 22:20 .
drwxr-xr-x 92 root root 4096 Aug 13 22:10 ..
-rwxr-xr-x  1 root root 3740 Feb 23  2022 apparmor
-rwxr-xr-x  1 root root 2920 May 22 20:40 apport
-rwxr-xr-x  1 root root 1232 Nov 22  2021 console-setup.sh
-rwxr-xr-x  1 root root 3062 Mar 17  2021 cron
-rwxr-xr-x  1 root root  937 Jan 13  2022 cryptdisks
-rwxr-xr-x  1 root root  896 Jan 13  2022 cryptdisks-early
-rwxr-xr-x  1 root root 3152 Jun 28  2021 dbus
-rwxr-xr-x  1 root root  985 Dec  2  2022 grub-common
-rwxr-xr-x  1 root root 1748 Feb 20  2022 hwclock.sh
-rwxr-xr-x  1 root root 2638 Oct 30  2021 irqbalance
-rwxr-xr-x  1 root root 1503 Jan 19  2022 iscsid
-rwxr-xr-x  1 root root 1479 Jul 24  2021 keyboard-setup.sh
-rwxr-xr-x  1 root root 2044 Jan  8  2021 kmod
-rwxr-xr-x  1 root root  695 May 19  2021 lvm2
-rwxr-xr-x  1 root root  586 May 19  2021 lvm2-lvmpolld
-rwxr-xr-x  1 root root 2433 Jan 19  2022 open-iscsi
-rwxr-xr-x  1 root root 1846 Nov 30  2023 open-vm-tools
-rwxr-xr-x  1 root root 1386 Feb 23  2022 plymouth
-rwxr-xr-x  1 root root  760 Feb 23  2022 plymouth-log
-rwxr-xr-x  1 root root  959 Feb 25  2022 procps
-rwxr-xr-x  1 root root 4417 Oct 11  2022 rsync
-rwxr-xr-x  1 root root 1222 Feb 18  2021 screen-cleanup
-rwxr-xr-x  1 root root 4060 Mar 13  2024 ssh
-rwxr-xr-x  1 root root 6871 Mar  8  2022 udev
-rwxr-xr-x  1 root root 2083 Sep 19  2021 ufw
-rwxr-xr-x  1 root root 1391 Feb 19  2021 unattended-upgrades
-rwxr-xr-x  1 root root 1306 Apr  9  2024 uuidd
```

Where is `/etc/init.d/tracker-fs`?

```
root@sandfly-victim:/etc/init.d# stat tracker-fs
  File: tracker-fs
  Size: 101             Blocks: 8          IO Block:
Device: fc01h/64513d    Inode: 17          Links: 1
Access: (0755/-rwxr-xr-x)  Uid: (    0/    root)   Gi
Access: 2025-08-13 22:20:21.968088601 +0000
Modify: 2025-08-13 22:20:21.956087701 +0000
Change: 2025-08-13 22:20:21.956087701 +0000
 Birth: 2025-08-13 22:20:21.956087701 +0000
root@sandfly-victim:/etc/init.d#
```

Why does `stat` show it's there?

Use `ls`, `cat`, `stat`, `file`, `dd`, to test for presence as well.

Initialization File Investigation

```
root@sandflysecurity-victim:~# cat /etc/init.d/tracker-fs
#!/bin/bash
#
case "$1" in
'start')
        /sbin/insmod /usr/lib64//tracker-fs
        ;;
'stop')
        ;;
esac
exit 0
root@sandflysecurity-victim:~#
```

`cat /etc/init.d/tracker-fs`

`/usr/lib64/tracker-fs` is our kernel module.

World's Fastest Rootkit Confirmation

```
root@sandflysecurity-victim:~/test# touch testing
root@sandflysecurity-victim:~/test# touch tracker-fs
root@sandflysecurity-victim:~/test# ls -al
total 8
drwxr-xr-x  2 root root 4096 Oct  1 22:32 .
drwx------ 10 root root 4096 Oct  1 22:30 ..
-rw-r--r--  1 root root    0 Oct  1 22:32 testing
root@sandflysecurity-victim:~/test#
```

tracker-fs where are you!?

Make a filename with the suspect name (`tracker-fs`) being hidden.

Do you see it with directory listing afterwards? Rootkit active if not.

Decloaking Hidden File Data

Decloaking Hidden File Data

```
root@sfly-d-reptile:/# cat /etc/modules
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.

# malicious content below

root@sfly-d-reptile:/#
```

Data invisible from system commands.

Some rootkits (like Reptile) will hide data inside files to evade detection.

This file (`/etc/modules`) has data hidden by a rootkit.

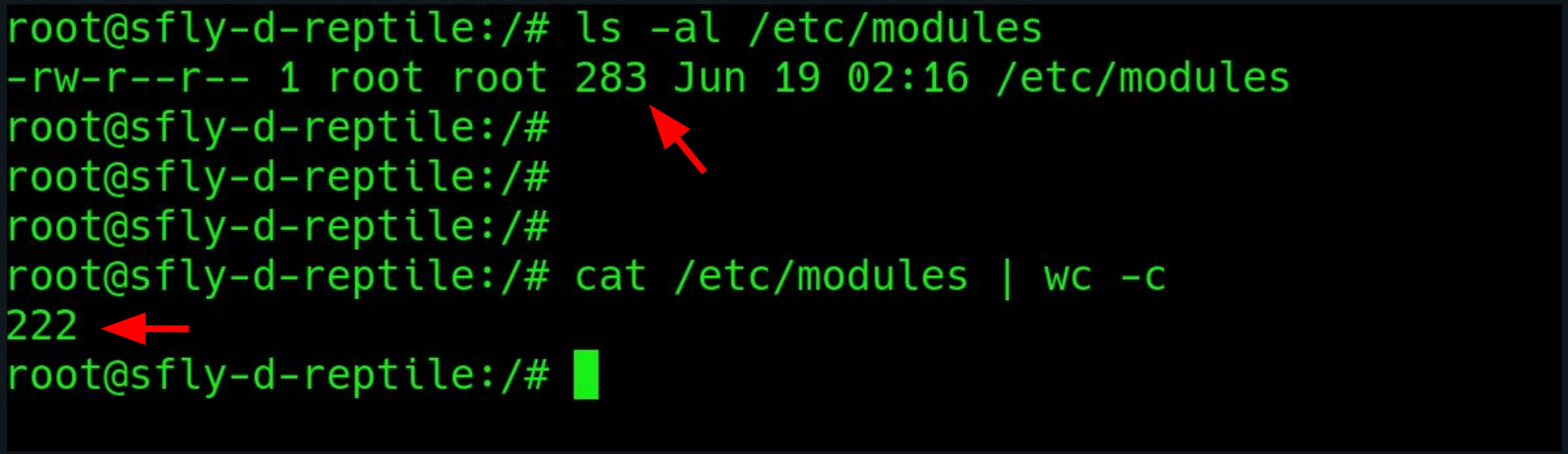
Decloaking Hidden File Data

```
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.

# malicious content below
~
~
~
~
~
~
~
~
~
~
~
~
~
"/etc/modules" 7L, 222C                                    1,1           All
```

Hides data even inside editors!

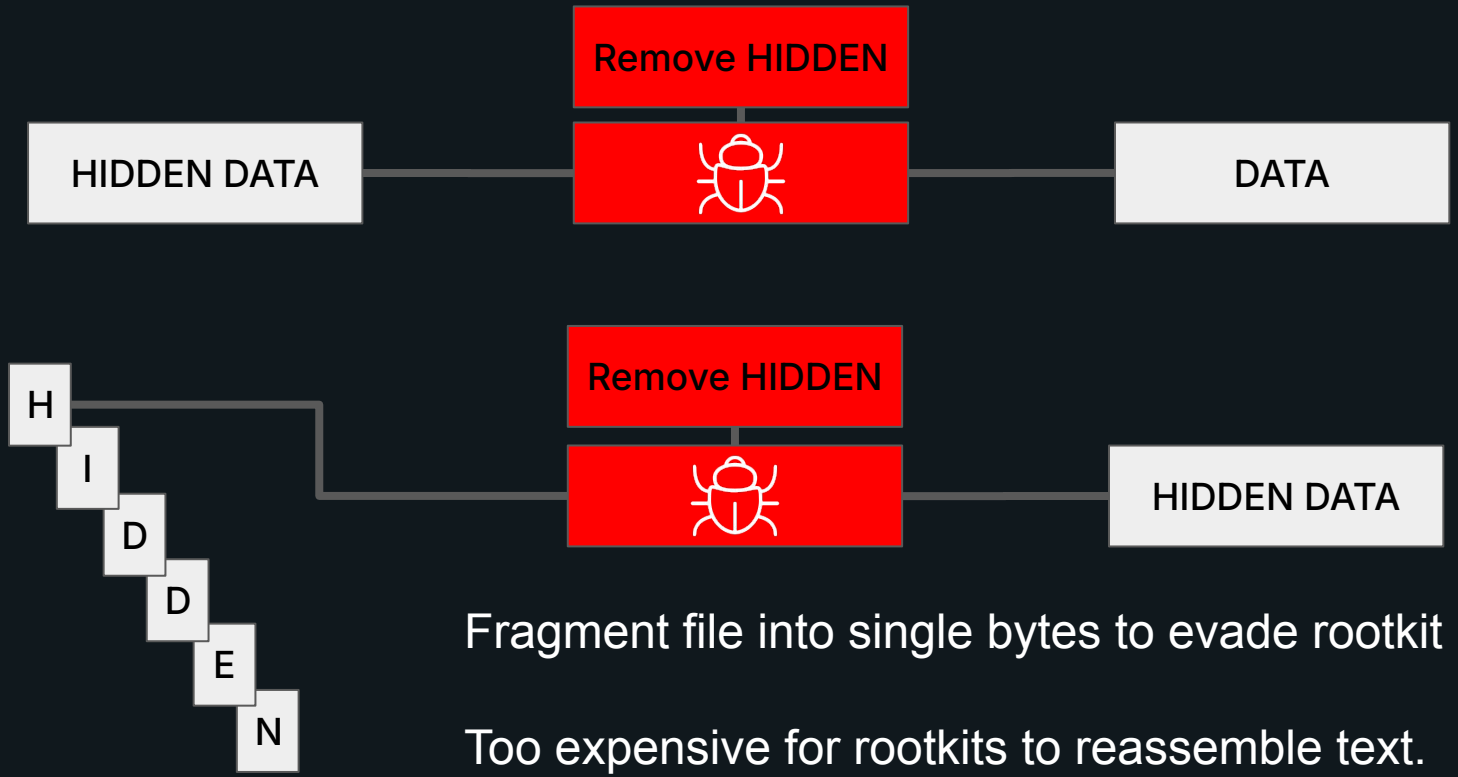
Byte Count Mismatch



`ls -al /etc/modules` (shows 283 bytes)

`cat /etc/modules | wc -c` (shows only 222 bytes!)

Single Byte Read Decloaking



Fragment file into single bytes to evade rootkit buffer.

Too expensive for rootkits to reassemble text.

Single Byte Read Decloaking

```
root@sfly-d-reptile:/# grep . /etc/modules --color=never
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# malicious content below
<reptile>
#this is hidden malicious file content
#</reptile>
root@sfly-d-reptile:/#
```

Single byte file read with `grep` to evade rootkit scrubbing:

```
grep . /path/to/file
```

Single Byte Read Decloaking

```
root@sfly-d-reptile:/# dd count=10000 bs=1 if=/etc/modules 2>/dev/null
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.

# malicious content below
#<reptile>
#this is hidden malicious file content
#</reptile>
root@sfly-d-reptile:/#
```

Single byte file read with `dd` to evade rootkit scrubbing:

```
dd count=10000 bs=1 if=/path/to/file 2>/dev/null
```

Tainted Kernels

Kernel Taint Legitimate Modules

```
ubnt@ubnt:~$ grep "(" /proc/modules
cvm_ipsec_kame 36927 0 - Live 0xffffffffc0a1b000 0xffffffffc0a19000 (O)
cavium_ip_offload 170013 0 - Live 0xffffffffc097b000 0xffffffffc0974000 (PO)
ubnt_nf_app 10236 1 cavium_ip_offload, Live 0xffffffffc0932000 0xffffffffc0930000 (PO)
tdts 554910 2 cavium_ip_offload,ubnt_nf_app, Live 0xffffffffc08ba000 0xffffffffc0899000 (PO)
ubnt_platform 285311 0 - Live 0xffffffffc0816000 0xffffffffc080c000 (PO)
ubnt@ubnt:~$
```

`grep "(" /proc/modules`

Common Taint Types

P - Proprietary
O - Out of Tree
E - Unsigned Module (DANGER)

Kernel Taint Check


```
cat /proc/sys/kernel/tainted
grep "(" /proc/modules
```

Where is my tainted module?

Kernel Taint Check Script

```
root@sandflysecurity-victim:~# ./kernel-chktaint.sh
Kernel is "tainted" for the following reasons:
 * unsigned module was loaded (#13)
For a more detailed explanation of the various taint flags see
 Documentation/admin-guide/tainted-kernels.rst in the Linux kernel sources
 or https://kernel.org/doc/html/latest/admin-guide/tainted-kernels.html
Raw taint value as int/string: 8192/'G                E      '
root@sandflysecurity-victim:~#
```

`kernel-chktaint.sh`

Shows unsigned module (taint type “E”) is present.

Again, why not showing under `/proc/modules` as type (E)?

Kernel Taint In Logs

```
root@sandflysecurity-victim:~# grep taint /var/log/kern.log
Sep 30 01:41:24 sandlfysecurity-victim kernel: [   24.354953] vmwfxs: module verification failed
 key missing - tainting kernel
```

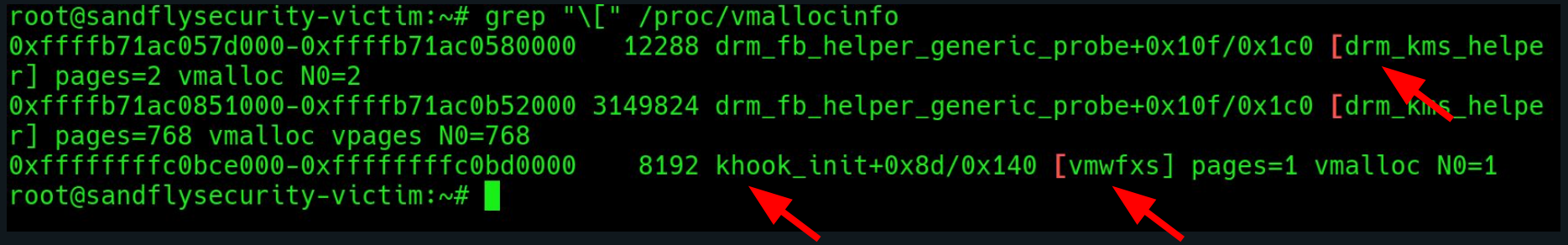
```
grep taint /var/log/kern.log
dmesg | grep taint
```

```
Sep 30 02:35:35 sandflysecurity-victim kernel: ISO 9660 Extensions: RRIP_1991A
Sep 30 02:35:51 sandflysecurity-victim kernel: vmwfxs: module verification failed: signature and/or required key
Sep 30 02:35:52 sandflysecurity-victim kernel: loop3: detected capacity change from 0 to 8
Sep 30 02:35:52 sandflysecurity-victim kernel: kauditd_printk_skb: 29 callbacks suppressed
```

```
journalctl -t kernel
journalctl -t kernel | grep taint
journalctl -t kernel | grep signature
```

Module “vmwfxs” is a person of interest.

Kernel Module Decloak



`grep "\[" /proc/vmallocinfo`

Shows some kernel modules in `[brackets]`:

`drm_kms_helper`

`vmwfxs`

`vmwfxs` also hows `khook_init` which is a known kernel hook library call.

Kernel Module Decloak

```
root@sandflysecurity-victim:~# grep drm_kms_helper /proc/modules
drm_kms_helper 315392 3 virtio_gpu, Live 0xffffffffc05c5000
syscopyarea 16384 1 drm_kms_helper, Live 0xffffffffc04f8000
sysfillrect 20480 1 drm_kms_helper, Live 0xffffffffc04f2000
sysimgblt 16384 1 drm_kms_helper, Live 0xffffffffc04ed000
fb_sys_fops 16384 1 drm_kms_helper, Live 0xffffffffc044c000
cec 65536 1 drm_kms_helper, Live 0xffffffffc0541000
drm 622592 3 virtio_gpu,drm_kms_helper, Live 0xffffffffc0454000
root@sandflysecurity-victim:~#
root@sandflysecurity-victim:~#
root@sandflysecurity-victim:~#
root@sandflysecurity-victim:~#
root@sandflysecurity-victim:~# grep vmwfxs /proc/modules
root@sandflysecurity-victim:~#
```

`grep <kernel_module_name> /proc/modules`

We should see the module names in `[brackets]` from `/proc/vmallocinfo`.

Why isn't `vmwfxs` here?

Kernel Module Decloak

```
root@sandflysecurity-victim:~# ./sandfly-kernel-module-decloak.sh
Linux Loadable Kernel Module (LKM) rootkit check 1.0.
Copyright (c)2025 Sandfly Security - Agentless Linux Security - https://www.sandflysecurity.com
Checking for hidden Linux kernel modules.


*** WARNING ***
Kernel module 'vmwfxs' is active and hiding.
The /proc/vmallocinfo entry showing it is loaded is the following:

0xffffffffc0bce000-0xffffffffc0bd0000    8192 khook_init+0x8d/0x140 [vmwfxs] pages=1 vmalloc N0=1

*** WARNING: This system has hidden kernel modules and may be operating a rootkit.

root@sandflysecurity-victim:~#
```

Free script to do show this type of hidden module (see links).

`sandfly-kernel-module-decloak.sh`

Kernel Module Investigation

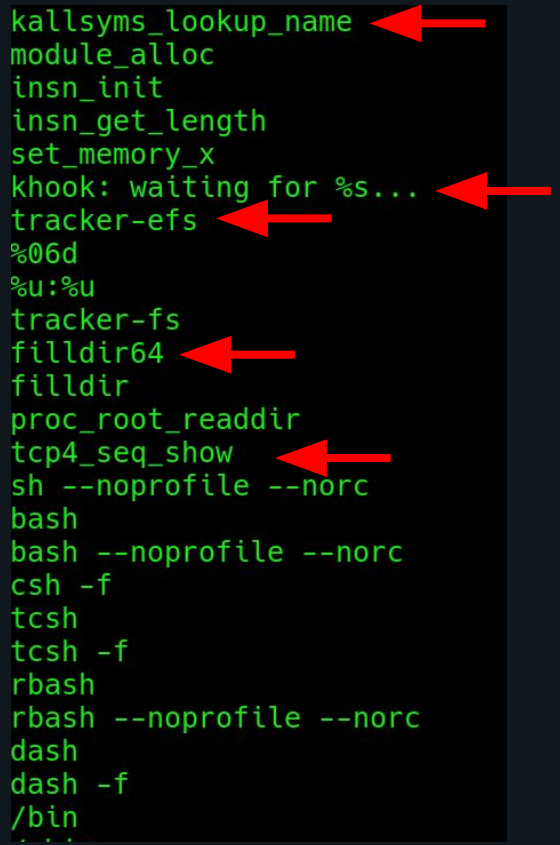
Hidden Module Data

```
root@sandflysecurity-victim:~# file /usr/lib64/tracker-fs
/usr/lib64/tracker-fs: ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), BuildID[sha1]=a
f5ba59cd, not stripped
root@sandflysecurity-victim:~# strings /usr/lib64/tracker-fs
^cXv
Linux
Linux
ATSL
D$0H
[A\]
D$ H
```

```
file /usr/lib64/tracker-fs
strings /usr/lib64/tracker-fs
```

Recommend copying binary to isolated system before doing these actions!

Hidden Module Data



Hooking system calls (kallsyms, khook)

Backdoor binary name (tracker-efs)

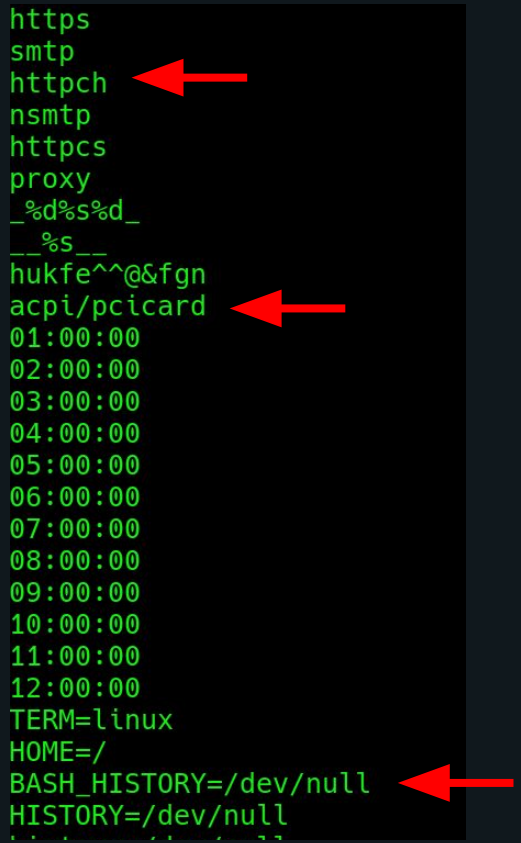
Hiding directory entries (filldir/readdir)

TCP concealment (tcp4_seq_show)

Shells for backdoor (bash, etc.)

Various paths

Hidden Module Data

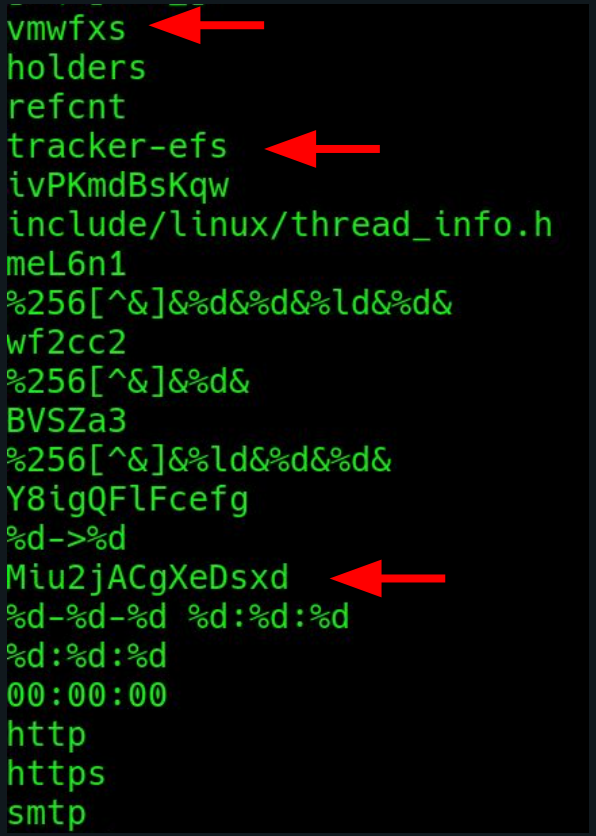


Various protocols

Control socket (/proc/acpi/pcicard)

Shell anti-forensics

Hidden Module Data



Module name (vmwfxs)

Backdoor process name (tracker-efs)

Various passwords (Miu2jACgXeDsxd)

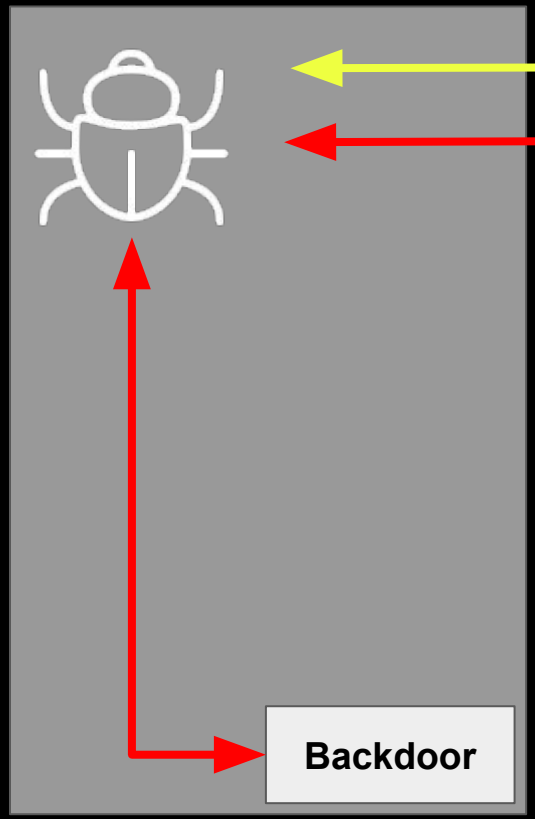
Backdoor Investigation

The Backdoor

+ Activates on any port with magic packet
+ Enables lateral movement or shell access
+ Has anti-forensics features

- Incomplete hiding exposes it
- Can recover binary when running
- Pressing attack always compromises stealth

Magic Packet Backdoor Activation



1) Sends magic packet with TCP Sequence and IP-ID set
2) Magic packet on any port activates it
3) Backdoor spawns (pressing attack)
4) Redirects TCP stream to backdoor
5) Traffic is encrypted to hide

Backdoor Features

```
root@sandflysecurity-attacker:~# ./tcat -H [redacted] -P 12345
Start Login Check......Connect OK

master pid: 1075, master port: 5034
Login Check OK
#CMD#> help

 shell  -- start shell
 upload <local path>|<remote path>
        -- upload local file to remote file, only single file
 download <remote path>|<local path>
        -- download remote file to local file, only single file
 nup, upload file, -h for more help
 ndown, download file, -h for more help
 trans, trans next machine, -h for more help
 back   -- back to pre-machine
 socks5 -o -p port, socks proxy, -h for more help
 inc     -- echo connection info
 exit   -- close & exit

#CMD#> shell -h
```

Backdoor Features

```
#CMD#> shell -h

        -s, --single : Single Command Exec Mode, No Shell
        -b <bash filepath>, --bash : Input Bash FilePath, default: /bin/sh --noprofile --
        -e <func num>, --env : func num: 0 -- no env, 1 -- use default env, 2 -- self def
        -i <self defined env>, --input : input self defined env
        default env: TERM=linux||HOME=/||PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bi
ory=/dev/null||HISTFILE=/dev/null||TMOUT=0
        -m <multiple>, --multi : shell input buffer size = 8K * multiple, default multipl
        -t <time>, --time : set packet delay time, default time: 0 us
                such as input 100KB data, just set -m 32 -t 500000
#CMD#> shell -b /bin/sh -t 500000
# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  lost+found  media  mnt  opt  proc
# whoami
root
#
```

Time delay to perhaps evade state-based network detection rules?

Backdoor Features

```
#CMD#> trans -h

        -x, --mod : select trans module, 1 - straight, 2 - normal proxy, 3 - socks proxy, 4 - li
        -H, --host : remote host
        -P, --port : remote port
        -m, --protocol : main protocol, such as: tcp, http, ssl, https, smtp
        -p, --password : password
        -e, --eth : ethernet interface
        -k, --knock : knock protocol, just for proxy trans, 0 - TCP, 1 - HTTP GET, 2 - HTTP POST
        -i, --sH : socks server host
        -o, --sP : socks server port
        -u, --su : socks server username
        -a, --sp : socks server user password
#CMD#>
```

Backdoor can chain to other rootkits with magic packet knocking

Backdoor Hidden Process Decloaking

```
root@sandflysecurity-victim:~/sandfly-processdecloak# ./sandfly-processdecloak
sandfly-processdecloak Version 1.0.6
Copyright (c) 2020-2022 Sandfly Security
Agentless Security for Linux  - www.sandflysecurity.com

Decloaking hidden Process IDs (PIDS) on Linux host.
Found hidden PID: 1075 with name: tracker-efs
Found hidden PID: 1076 with name: bash
root@sandflysecurity-victim:~/sandfly-processdecloak#
root@sandflysecurity-victim:~/sandfly-processdecloak#
root@sandflysecurity-victim:~/sandfly-processdecloak# ls -al /proc | grep 1075
root@sandflysecurity-victim:~/sandfly-processdecloak#
root@sandflysecurity-victim:~/sandfly-processdecloak# ls -al /proc | grep 1076
root@sandflysecurity-victim:~/sandfly-processdecloak#
```

PID brute force with `sandfly-processdecloak` finds hidden processes.

Can't see processes with direct listing of `/proc`.

Backdoor Hidden Network Port

```
root@sandflysecurity-victim:~# lsof -nP -iTCP -sTCP:LISTEN
COMMAND   PID            USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
systemd-r 534 systemd-resolve   14u  IPv4  18031      0t0  TCP 127.0.0.53:53 (LISTEN)
sshd      701            root    3u  IPv4  18956      0t0  TCP *:22 (LISTEN)
sshd      701            root    4u  IPv6  18969      0t0  TCP *:22 (LISTEN)
root@sandflysecurity-victim:~#
```

```
root@sandflysecurity-victim:~# ss -tulnp
Netid  State    Recv-Q   Send-Q     Local Address:Port     Peer Address:Port  Process
udp    UNCONN   0        0          127.0.0.53%lo:53           0.0.0.0:*      users:(("systemd-resolve",pid=534
tcp    LISTEN   0        128              0.0.0.0:22           0.0.0.0:*      users:(("sshd",pid=701,fd=3))
tcp    LISTEN   0        10               0.0.0.0:5034         0.0.0.0:*
tcp    LISTEN   0        4096       127.0.0.53%lo:53           0.0.0.0:*      users:(("systemd-resolve",pid=534
tcp    LISTEN   0        128                 [::]:22              [::]:*      users:(("sshd",pid=701,fd=4))
root@sandflysecurity-victim:~#
```

`lsof` is not showing the hidden listener, but `ss` does:

```
ss -tulnp
```

Backdoor Hidden Process Investigation

```
root@sandflysecurity-victim:~/sandfly-processdecloak# ls -al /proc/1075
total 0
dr-xr-xr-x   9 root root 0 Oct  1 00:20 .
dr-xr-xr-x 149 root root 0 Sep 30 23:52 ..
-r--r--r--   1 root root 0 Oct  1 00:32 arch_status
dr-xr-xr-x   2 root root 0 Oct  1 00:32 attr
-rw-r--r--   1 root root 0 Oct  1 00:32 autogroup
-r--------   1 root root 0 Oct  1 00:32 auxv
-r--r--r--   1 root root 0 Oct  1 00:32 cgroup
--w-------   1 root root 0 Oct  1 00:32 clear_refs
-r--r--r--   1 root root 0 Oct  1 00:32 cmdline
-rw-r--r--   1 root root 0 Oct  1 00:32 comm
-rw-r--r--   1 root root 0 Oct  1 00:32 coredump_filter
-r--r--r--   1 root root 0 Oct  1 00:32 cpu_resctrl_groups
-r--r--r--   1 root root 0 Oct  1 00:32 cpuset
lrwxrwxrwx   1 root root 0 Oct  1 00:32 cwd -> /
-r--------   1 root root 0 Oct  1 00:32 environ
lrwxrwxrwx   1 root root 0 Oct  1 00:32 exe -> /usr/include/tracker-fs/tracker-efs
dr-x------   2 root root 0 Oct  1 00:20 fd
```

We can see process directory if we know the cloaked PID.

Backdoor Hidden Process Anti-Forensics

```
root@sandflysecurity-victim:/proc/1075# strings environ
TERM=linux
HOME=/
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin
BASH_HISTORY=/dev/null
HISTORY=/dev/null
history=/dev/null
HISTFILE=/dev/null
TMOUT=0
root@sandflysecurity-victim:/proc/1075#
```

`strings /proc/PID/environ`

Evading command history is highly suspicious!

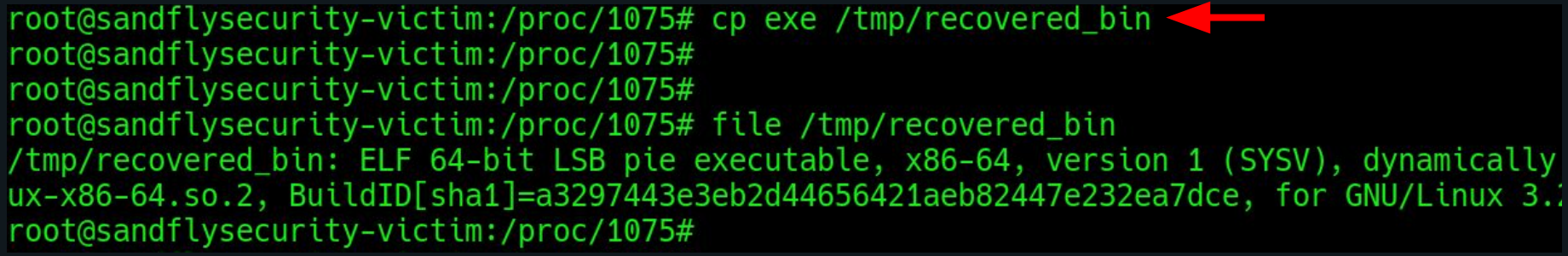
Pressing the attack on host exposes the entire rootkit in multiple ways.

Backdoor Hidden Process Binary Recovery

```
root@sandflysecurity-victim:~/sandfly-processdecloak# ls -al /proc/1075
total 0
dr-xr-xr-x   9 root root 0 Oct  1 00:20 .
dr-xr-xr-x 149 root root 0 Sep 30 23:52 ..
-r--r--r--   1 root root 0 Oct  1 00:32 arch_status
dr-xr-xr-x   2 root root 0 Oct  1 00:32 attr
-rw-r--r--   1 root root 0 Oct  1 00:32 autogroup
-r--------   1 root root 0 Oct  1 00:32 auxv
-r--r--r--   1 root root 0 Oct  1 00:32 cgroup
--w-------   1 root root 0 Oct  1 00:32 clear_refs
-r--r--r--   1 root root 0 Oct  1 00:32 cmdline
-rw-r--r--   1 root root 0 Oct  1 00:32 comm
-rw-r--r--   1 root root 0 Oct  1 00:32 coredump_filter
-r--r--r--   1 root root 0 Oct  1 00:32 cpu_resctrl_groups
-r--r--r--   1 root root 0 Oct  1 00:32 cpuset
lrwxrwxrwx   1 root root 0 Oct  1 00:32 cwd -> /
-r--------   1 root root 0 Oct  1 00:32 environ
lrwxrwxrwx   1 root root 0 Oct  1 00:32 exe -> /usr/include/tracker-fs/tracker-efs
dr-x------   2 root root 0 Oct  1 00:20 fd
```

It’s extremely easy to recover a process binary, even if deleted from disk.

Backdoor Hidden Process Binary Recovery

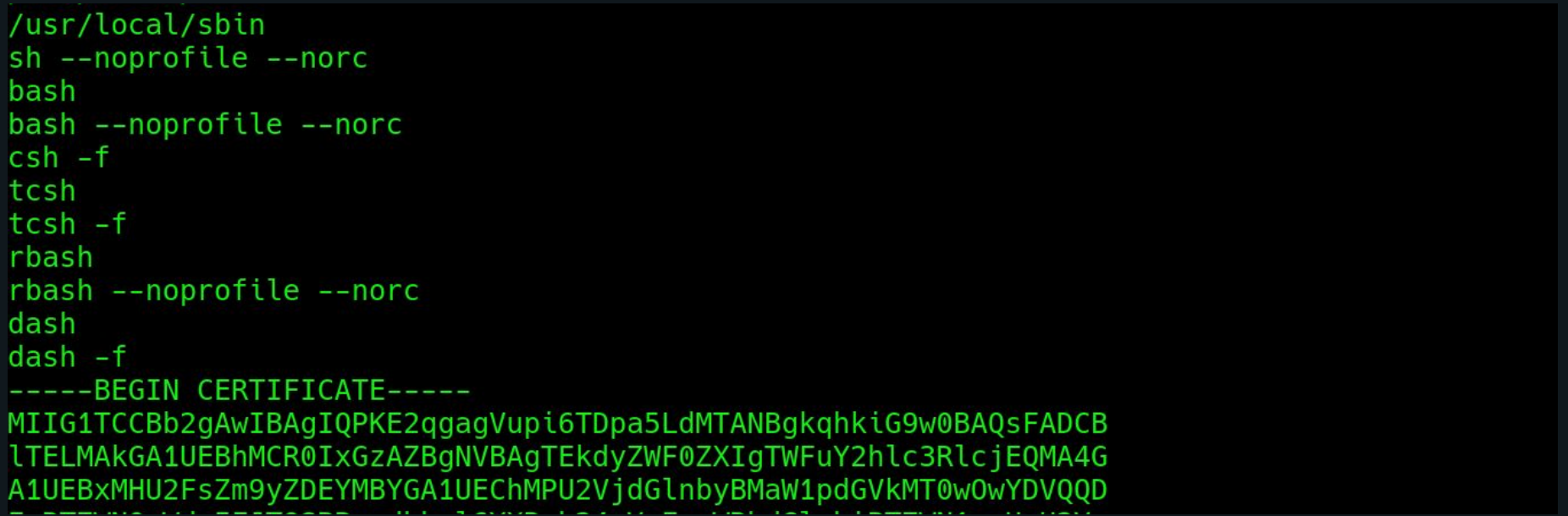



```
cp /proc/PID/exe /tmp/recovered_bin

file /tmp/recovered_bin
sha512sum /tmp/recovered_bin
…
```

Make a backup and move it onto a secured/isolated host.

Backdoor Binary Analysis



`strings /tmp/recovered_bin`

Inspecting backdoor shows shells, private keys, suspicious function names, etc.

Load Detection (Experimental)

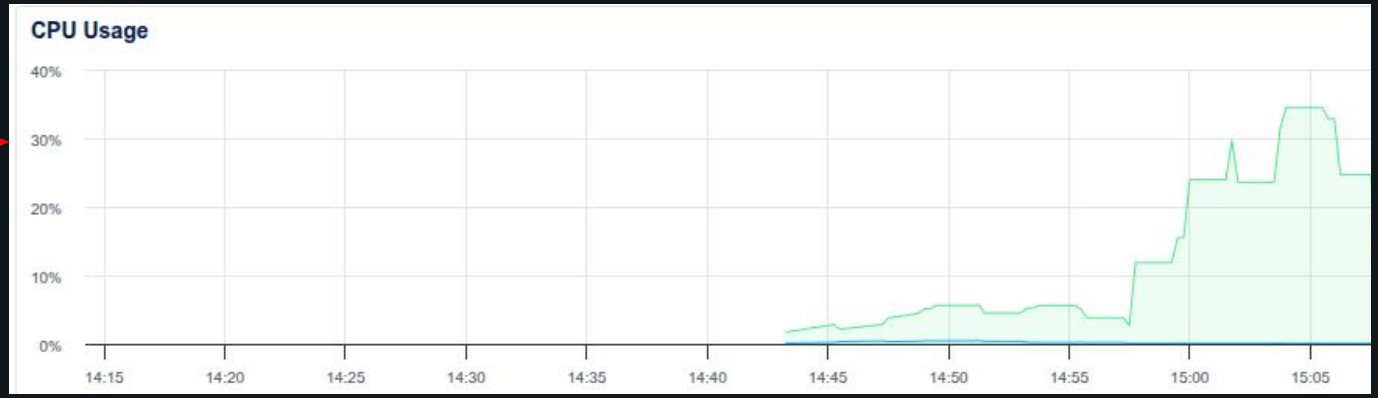
Load Detection

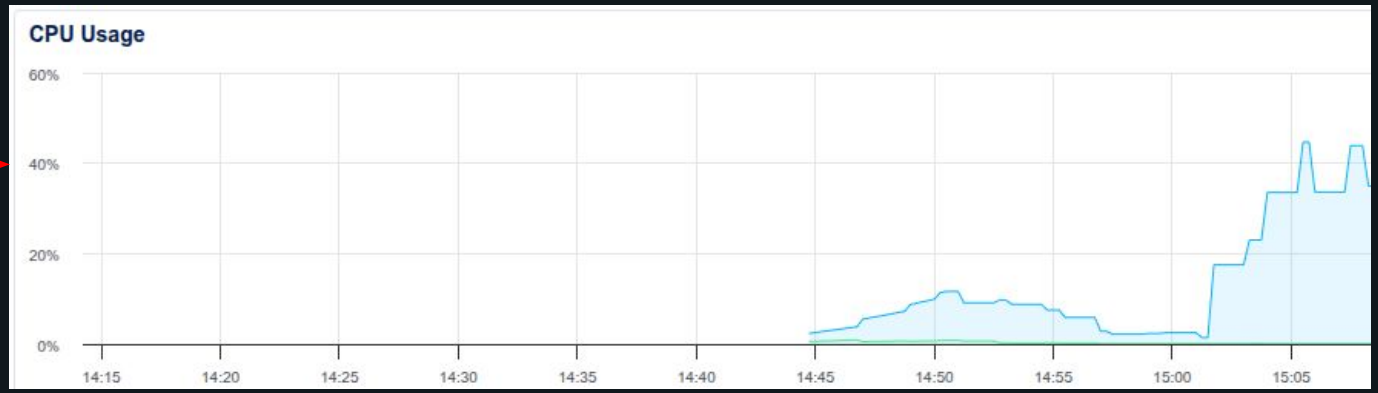
```
root@sandflysecurity-attacker:/# ./findspeed.sh
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
real    0m56.105s
user    0m23.172s
sys     0m27.940s
root@sandflysecurity-attacker:/#
```

```
root@sandflysecurity-victim:/# ./findspeed.sh
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43
45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84
86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
real    1m36.800s
user    0m34.256s
sys     0m57.093s
root@sandflysecurity-victim:/#
```

```
time {
for i in $(seq 1 100)
do
    echo -n "$i "
    find / -name "sandfly" &> /dev/null
done
}
```

Load Detection





Conclusions

Stealth rootkits work, but are fragile

More visible when operator presses attack

Often not found because nobody is looking

Links

https://phrack.org/issues/72/7_md#article

https://sandflysecurity.com/blog/leaked-north-korean-linux-stealth-rootkit-analysis

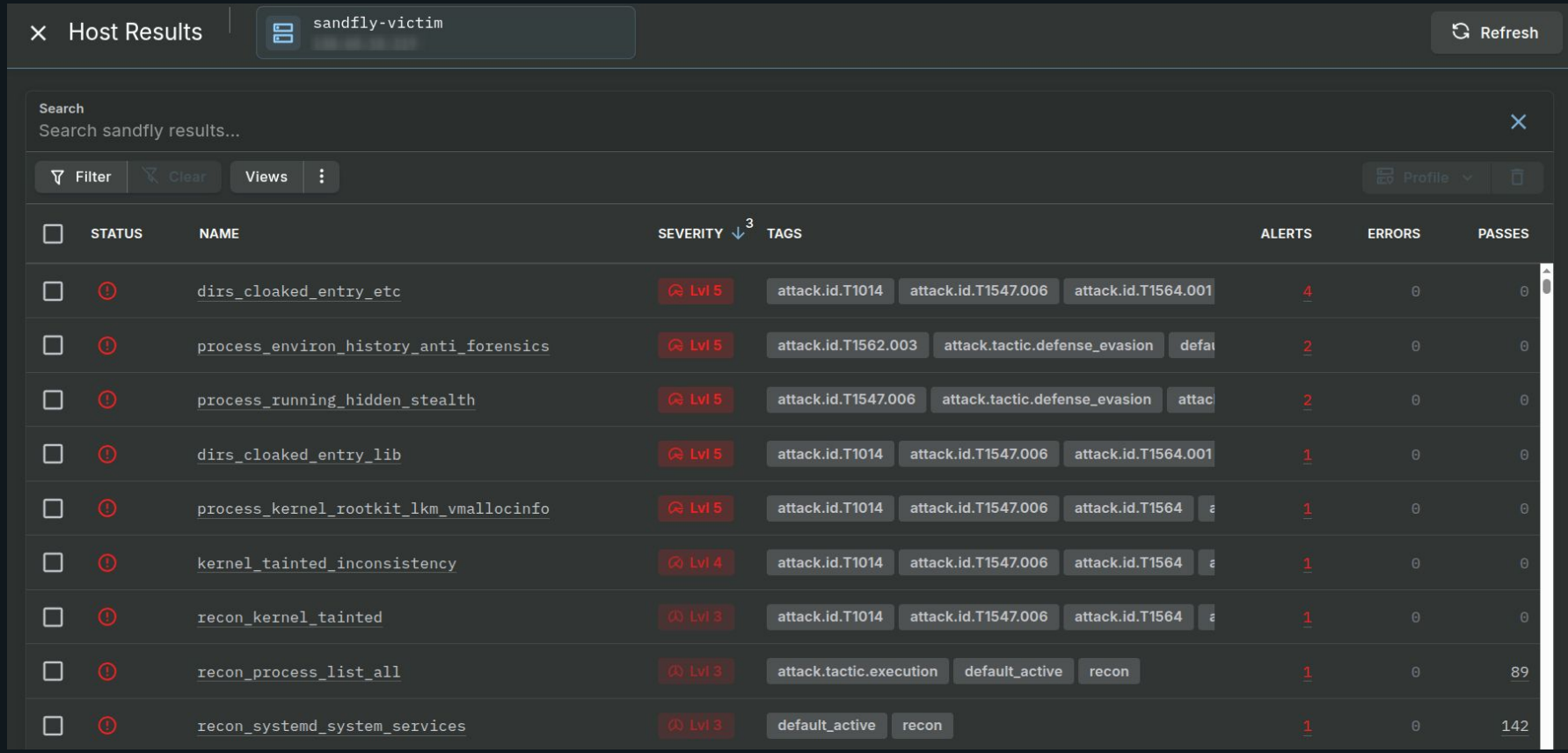
https://github.com/sandflysecurity

https://docs.kernel.org/admin-guide/tainted-kernels.html

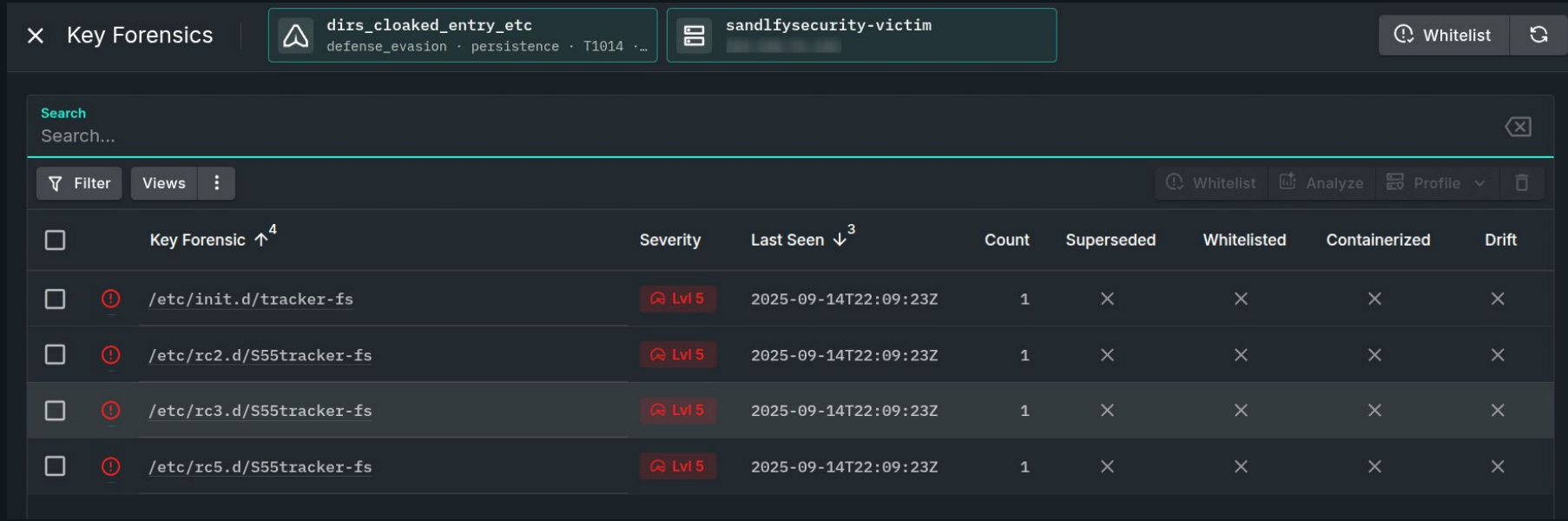
Extra Slides

Using Sandfly to Cheat

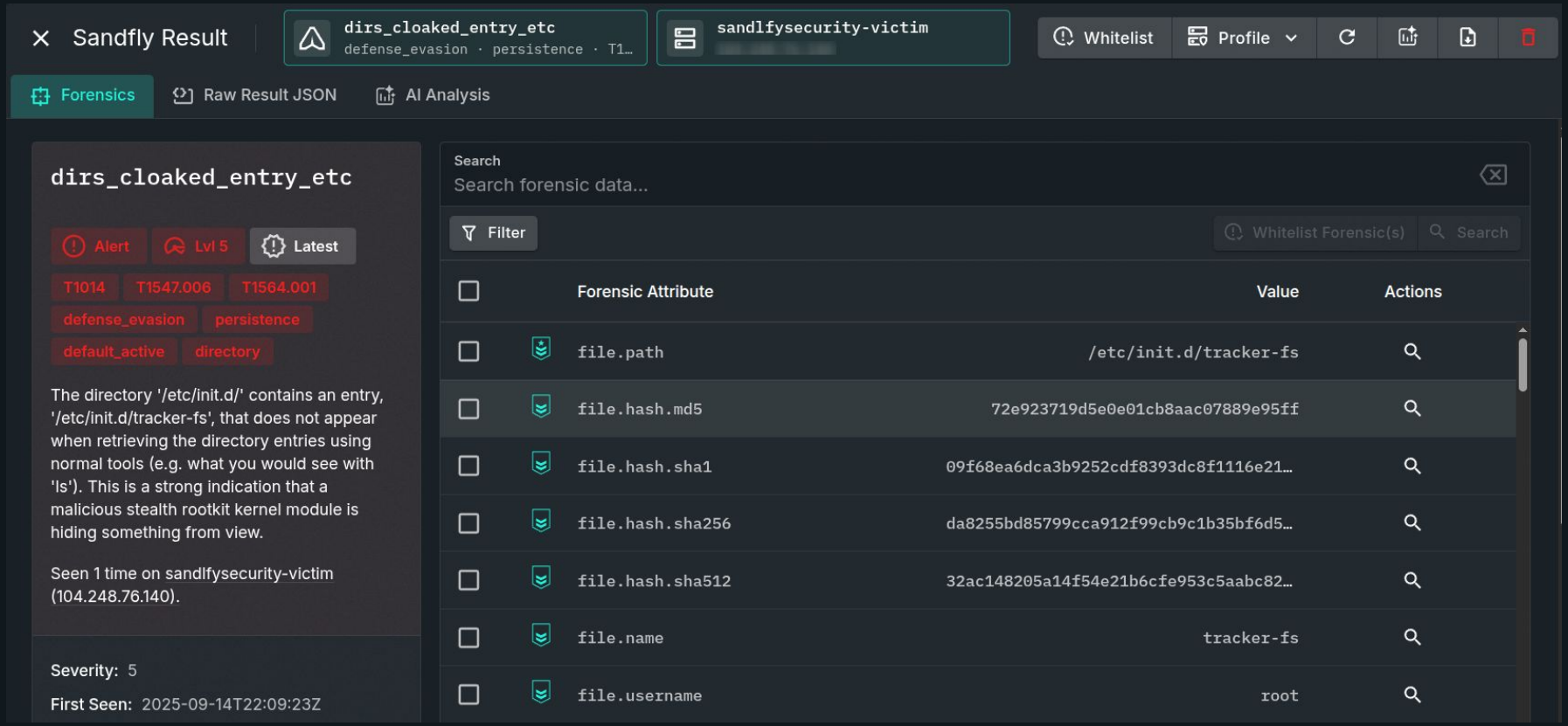
Sandfly Stealth Rootkit Alerts



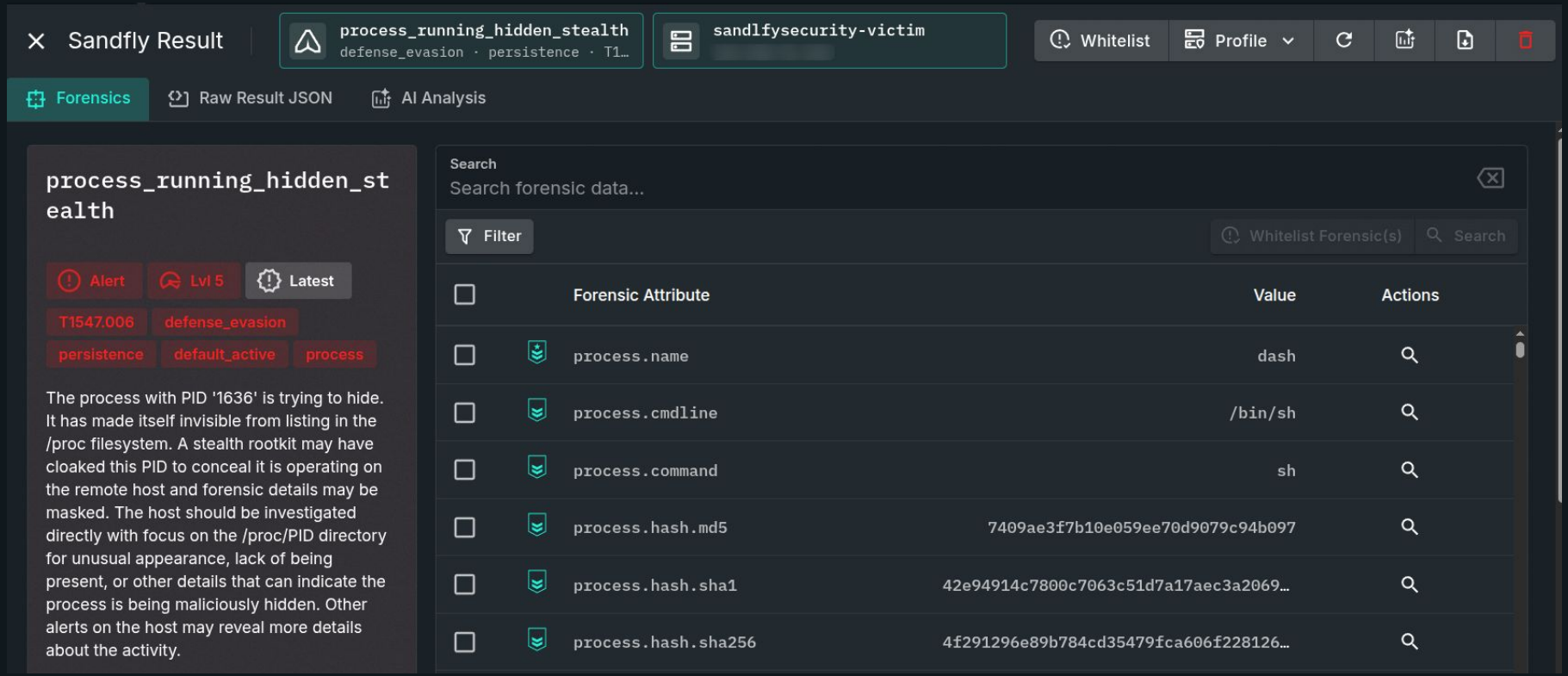
Sandfly Decloaking Hidden Files



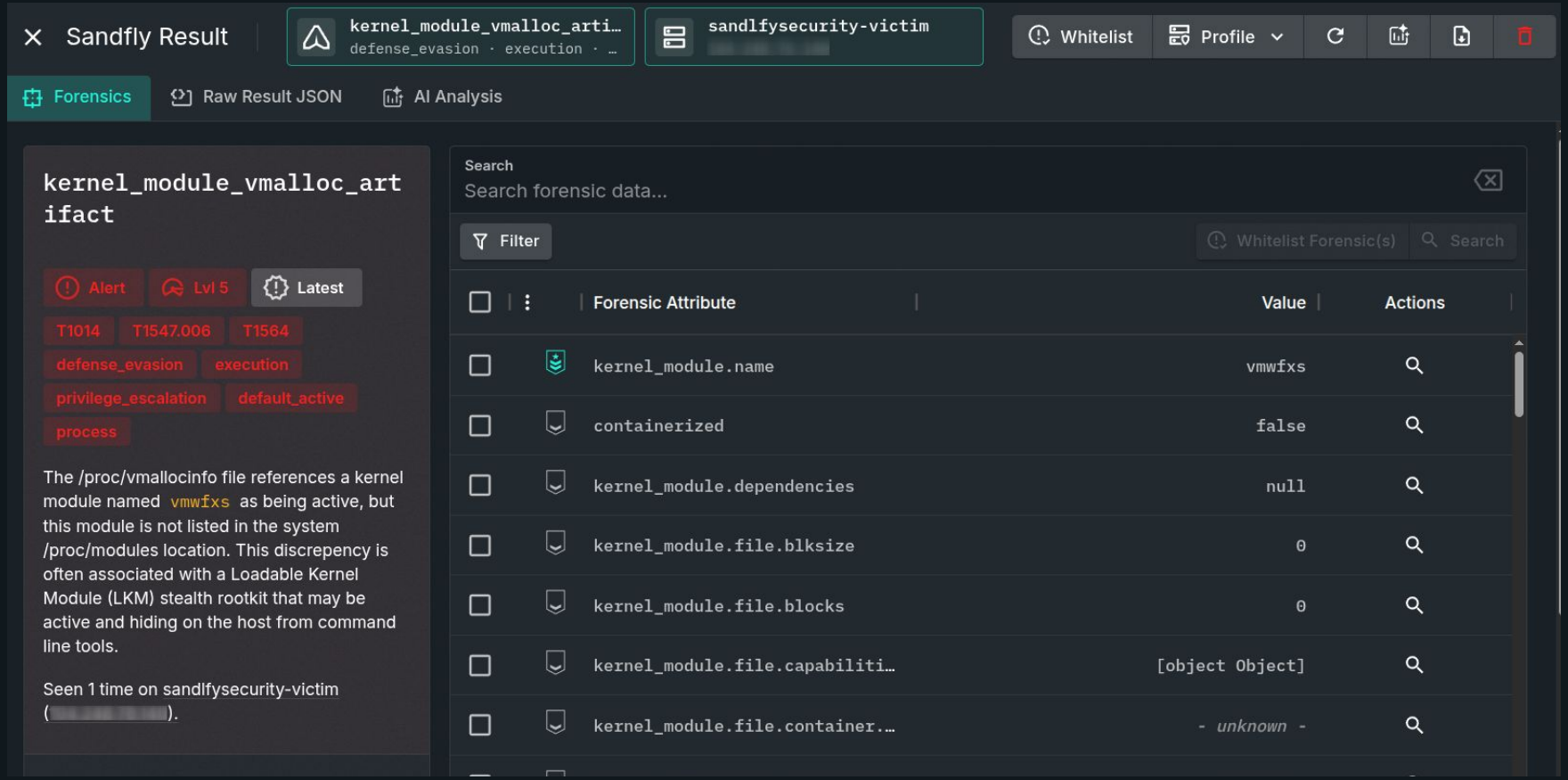
Sandfly Decloaking Init Scripts



Sandfly Decloaking Backdoor Process



Sandfly Decloaking Hidden Kernel Module



Sandfly Identifies Kernel Taint Inconsistency