Analysis of the VENOM Linux rootkit

Introduction

EGI[1] operates an infrastructure joining national research and development sites all over Europe. The operational security in the infrastructure is supervised by the EGI CSIRT. This document reports on malicious software and other artifacts that were analyzed by the EGI CSIRT recently.

The malware studied acted as a Linux rootkit aimed at maintaining unauthorized access on compromised Linux systems. The rootkit was titled VENOM, referring to a term often used in the internal protocol implemented in the malware.

The main part of the rootkit consisted of a kernel module and userland executable. The analysis showed functions very similar to backdoor tools found during investigation of a compromise of an IRC server at freenode in 2014, which were described in report published after the incident [1].

In short, when the attacker gains access to a machine as root, they deploy the malicious kernel module and userland binary and configure them to be loaded upon the boot of the machine. The kernel module implements a port-knocking mechanism, which allows the attacker to start an instance of the backdoor process that can be controlled remotely. The protocol between the backdoor and attacker is based on exchange of text messages that are encrypted using a key hard-wired in the code. Before being granted access to the backdoor the attacker must authenticate using a static password. If it is successful, the attacker can utilize several features provided, such as execution of an interactive shell or manipulating files on the compromised node.

The rootkit samples that were examined by EGI CSIRT showed new features added to the malware, suggesting that the rootkit was extended and new generation of the malware deployed. More details about the functions of this malware are provided in the rest of this report.

Rootkit description

The rootkit suite consists of a Linux loadable kernel module and a userland helper executable. It was deployed with root privileges. Several samples of the rootkit were collected and analyzed. It was only possible to analyze the binary samples since no source codes were available.

[1] https://www.egi.eu/

Kernel module

Having analyzed the module, we find it performs in the same way as was described in the report on the freenode incident [1]. The report did not disclose the precise way how the backdoor was activated in order not to threaten investigations ongoing at that time. For the sake of completeness, the precise mechanism is described here.

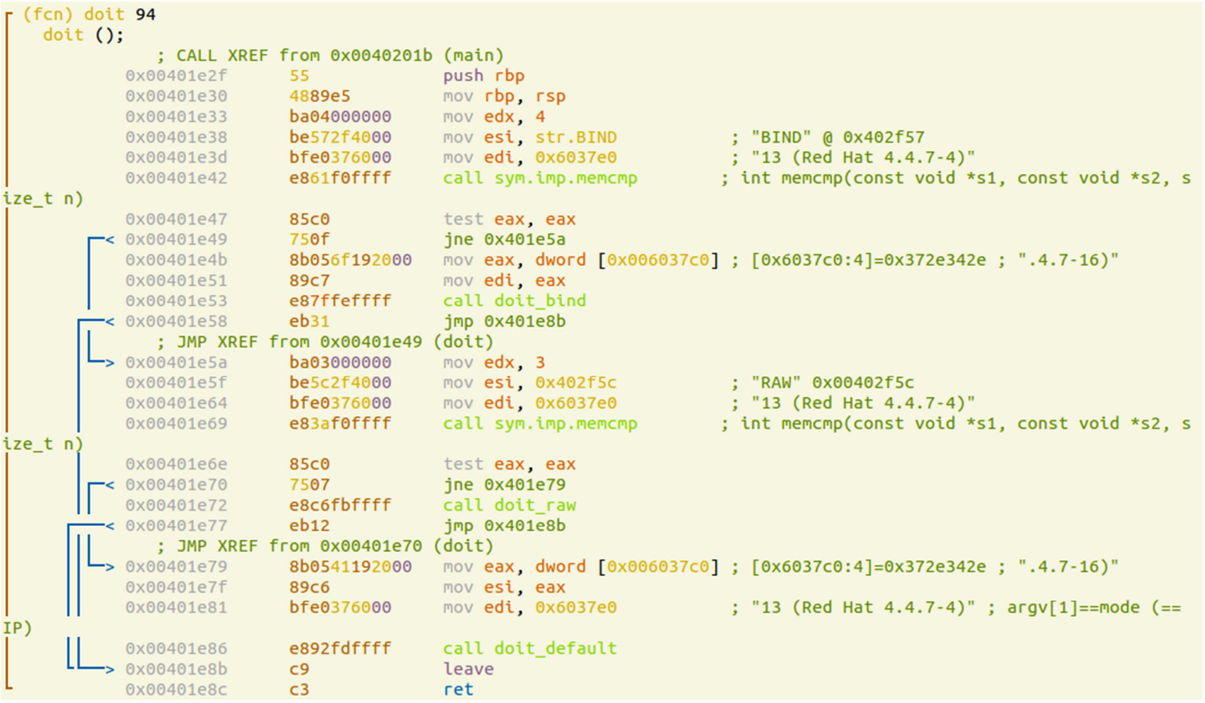
The module checks all incoming TCP packets and waits for a ‘magic’ set of packets that triggers the backdoor function. In order to trigger the backdoor the module must see three specific TCP packets where Source Port + Sequence Number sums to the value of 1221. Once three such packets are received, the module executes the userland part of the rootkit with the source IP and the value of the TCP window field passed as the first and second parameter, respectively. This userland, backdoor process then uses these parameters as destination IP and port and tries to connect back to the attacker. The path of the userland executable is built into the kernel module.

The module was named cpu_cachelift.ko and was compiled as standard ELF relocatable binary file.

Userland helper

Analyses performed suggests that the userland binary was significantly extended compared to what is described in the report for freenode incident. The helper was extended to support a different way of handling the backdoor together with additional features for the attacker.

The executable now implements three different modes of operation that determine its behavior.



```
┌ (fcn) doit 94
│   doit ();
│            ; CALL XREF from 0x0040201b (main)
│           0x00401e2f      55             push rbp
│           0x00401e30      4889e5         mov rbp, rsp
│           0x00401e33      ba04000000     mov edx, 4
│           0x00401e38      be572f4000     mov esi, str.BIND          ; "BIND" @ 0x402f57
│           0x00401e3d      bfe0376000     mov edi, 0x6037e0          ; "13 (Red Hat 4.4.7-4)"
│           0x00401e42      e861f0ffff     call sym.imp.memcmp       ; int memcmp(const void *s1, const void *s2, s
ize_t n)
│           0x00401e47      85c0           test eax, eax
│       ┌─< 0x00401e49      750f           jne 0x401e5a
│       │   0x00401e4b      8b056f192000   mov eax, dword [0x006037c0] ; [0x6037c0:4]=0x372e342e ; ".4.7-16)"
│       │   0x00401e51      89c7           mov edi, eax
│       │   0x00401e53      e87ffeffff     call doit_bind
│      ┌──< 0x00401e58      eb31           jmp 0x401e8b
│      │└     ; JMP XREF from 0x00401e49 (doit)
│      │└─> 0x00401e5a      ba03000000     mov edx, 3
│      │    0x00401e5f      be5c2f4000     mov esi, 0x402f5c          ; "RAW" 0x00402f5c
│      │    0x00401e64      bfe0376000     mov edi, 0x6037e0          ; "13 (Red Hat 4.4.7-4)"
│      │    0x00401e69      e83af0ffff     call sym.imp.memcmp       ; int memcmp(const void *s1, const void *s2, s
ize_t n)
│      │    0x00401e6e      85c0           test eax, eax
│      │┌─< 0x00401e70      7507           jne 0x401e79
│      ││   0x00401e72      e8c6fbffff     call doit_raw
│     ┌───< 0x00401e77      eb12           jmp 0x401e8b
│     ││└     ; JMP XREF from 0x00401e70 (doit)
│     ││└─> 0x00401e79      8b0541192000   mov eax, dword [0x006037c0] ; [0x6037c0:4]=0x372e342e ; ".4.7-16)"
│     ││    0x00401e7f      89c6           mov esi, eax
│     ││    0x00401e81      bfe0376000     mov edi, 0x6037e0          ; "13 (Red Hat 4.4.7-4)" ; argv[1]==mode (==
IP)
│     ││    0x00401e86      e892fdffff     call doit_default
│     └└──> 0x00401e8b      c9             leave
└           0x00401e8c      c3             ret
```

Choosing among three modes of operation

- RAW mode
    - Specified by the RAW option on the command-line
    - In this mode the executable acts as a daemon listening for initiating commands from the network
    - A raw socket is created and the process starts listening to all packets incoming to the machine.
    - Once it detects a magic packet, it executes itself again and passes arguments specified in the packet (IP and port to connect back to the attacker). The parameters are expected to be delimited using ‘|’.
    - The magic packet is any IP packet that contains SSH-2.5-OpenSSH_6.1.9 string starting at offset 52
    - Note, that this is just a trigger of the backdoor, the rest of the communication is handled by the Connect mode.
- BIND mode
    - Specified by the BIND option on the command-line
    - In this mode the process binds to the local port specified and waits for incoming connections
    - All subsequent communication follows the standard VENOM protocol implemented in the Connect mode.
- Connect mode
    - The default if neither RAW nor BIND is specified on the command-lineArguments are interpreted as IP and port to connect to
    - This mode is what was described in the original freenode report, i.e. it connect back to the attacker using the VENOM protocol:
        - All exchanges are encrypted using RC4 (key == 'justCANTbeSTOPPED').
        - First the backdoor sends '%%VENOM%AUTHENTICATE%%'
        - It expects a password, which should correspond to crypt('ABZFZU.wWdRDU')
        - If the password matches, the backdoor sends '%%VENOM%OK%OK%%' and waits for a command

The code seems to be prepared to operate as a long-running daemon now, not relying on the trigger by the kernel module. It can be deemed as a fail-over solution if the module cannot be used or is not functioning

The set of functions seems to have been extended by new functions that the attacker can use. In addition to existing functions to execute a shell and read/write arbitrary file, three other commands were implemented:

- Command execution (blind)
- Remote proxy (connect to remote host)
- Local proxy (bind to local port and wait for incoming connection)

The protocol is easily extensible so the changes are backwards compatible.

The backdoor requires a password. The code contained a hash of the DES password, which was identical in all samples we analyzed.

All the samples analyzed were standard ELF binaries, dynamically linked. In all the cases we analyzed the binary was named /var/lib/mkinitramfs.

Persistence

The rootkit is deployed by the attacker as a tar.gz archive file stored on the local filesystem as file named /usr/share/man/man5/printers_cupsd.conf.5.gz. The package contains the kernel module and userland binary.

In order to make sure that the malware is active, the attacker modifies the startup script to bootstrap the malware processes. /etc/rc.d/init.d/functions is modified to run a Bash script that unpacks the malware from a local archive and launches the processes.

The script was always stored as /etc/X11/applnk/.window. Upon invocation, it unpacks the archive to /tmp, tries to load the kernel module and possibly starts the helper executable in the RAW mode. Once all processes are started it removes the directory from /tmp.

The change of the functions file as well as the installation script is slightly obfuscated to prevent detection of readable strings. For instance the adaptations of /etc/rc.d/init.d/functions read:

```
if [ $(echo $0|grep -c network) -eq 0 ]; then

L41=`/bin/echo                                                      -e
"\0057\0145\0164\0143\0057\0130\0061\0061\0057\0141\0160\0160\0154\0156\0153
\0057\0056\0167\0151\0156\0144\0157\0167"`

    $L41 >/dev/null 2>&1

fi
```

Operations of the attacker

The attacker uses an automated way (a script or binary executable) to deploy the malware files on the filesystem. During the deployment they manipulate system time and/or change the filesystem so that the dropped files are assigned the wrong modification times, which assigns misleading dates to the files.

All changes are completed, in a matter of minutes. A sequence of multiple machines with similar changes applied was reported which might indicate a massive update of the malware was performed by the attacker.

Whenever possible (i.e. a C environment and kernel development files are present), the attacker compiles the code on the compromised machine. The compilation is done from RAM disk so no traces or artifacts are left on disks.

The attacker is believed to use stolen SSH credentials to obtain access to machines. They try to remove all local records of these activities very carefully However, no traces of a trojan in SSH binaries have been detected.

Detection

Host-based Indicators of Compromise

- Existence of files related to rootkit:
  - /etc/X11/applnk/.window
  - /var/lib/mkinitramfs
  - /usr/share/man/man5/printers_cupsd.conf.5.gz
- Modified /etc/rc.d/init.d/functions
- Loaded kernel module cpu_cachelift
- Running process(es) (e.g. crond) using a raw socket (consult the output of netstat/ss)

Network-level Indicators of Compromise

- The kernel port-knocking requires 3 specific TCP packets featuring:
  "TCP Source Port + Sequence Number == 1221"
- The userland knocking requires an IP packet containing "SSH-2.5-OpenSSH_6.1.9" at offset 52. Note that, the SSH-2.5-OpenSSH_6.1.9 pattern is unique despite the resemblance with the format of the OpenSSH version used widely.
- Traffic on 9090/tcp involving binary or ELF files was mentioned. However, the attacker can easily use other ports.

Relevant strings

- %%VENOM%CTRL%MODE%%
- %%VENOM%OK%OK%%
- %%VENOM%WIN%WN%%
- %s%c%d:%d
- %%VENOM%AUTHENTICATE%%
- . entering interactive shell
- %s%c%c%c%s
- . processing ltun request
- . processing rtun request
- . processing get request
- . processing put request
- - accept failed
- - listen failed
- - bind failed
- venom by mouzone

- justCANTbeSTOPPED

Recommendations

It is suggested to check systems thoroughly since the filenames and name of the kernel module can be changed easily by the attacker. It is highly recommended to deploy and use a remote logging service (e.g. for system logs).

References

[1] David Cannings. "Analysis of the Linux backdoor used in freenode IRC network compromise". NCC Group. October 2014. https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2014/october/analysis-of-the-linux-backdoor-used-in-freenode-irc-network-compromise/.

Credits

The analysis of the VENOM rootkit was performed by EGI CSIRT together with other security teams. The forensics analysis was done by members of the EGI CSIRT, namely experts from Academia Sinica, CERN, and CESNET. In the course of the handling of the incident close collaboration existed between EGI CSIRT and the security team of CERN/WLCG who provided more insight into the scope and nature of the incident.

Comments or questions can be sent to csirt@egi.eu.

The report was published on 12 January 2017 (version 1.0), https://wiki.egi.eu/wiki/Venom_Rootkit.