Stefan Walter
(with help from Manuel Stotz)

SySS
THE PENTEST EXPERTS.

Intro to Linux Kernel Debugging

…Source-code level debugging with gdb

Goals

THE PENTEST EXPERTS.

- ➔ Interactively guide through kernel code
- ➔ Inspect kernel state/memory
- ➔ Modify kernel state/memory

Content

THE PENTEST EXPERTS.

- Part I: Intro to kernel debugging
- Part II: Understanding the Linux kernel exploit Dirty Pipe (CVE-2022-0847)

About me

THE PENTEST EXPERTS.

- ➔ Interested in Linux / kernel
- ➔ Small disclaimer: no expert (no contributor, not the author of Dirty Pipe)

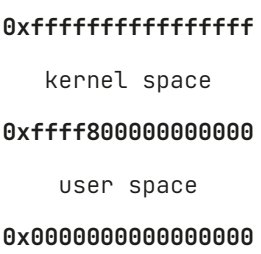
THE PENTEST EXPERTS.

Part I: Intro to kernel debugging

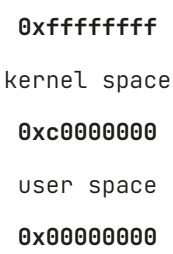
Operating system terminology

- Kernel / kernel space: *central OS-component managing hardware*
- Userland / user space: *everything else running on the system*

x86-64 virtual address space layout[1]

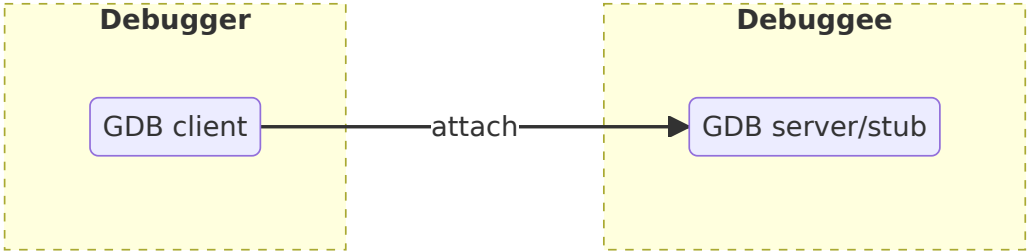


x86 virtual address space layout[2]



[1] https://www.kernel.org/doc/html/latest/arch/x86/x86_64/mm.html
[2] https://www.kernel.org/doc/html/latest/mm/highmem.html

Debug setup



QEMU gdbstub

- Option `-s` (= `-gdb tcp::1234`)
- Documentation:
  - `https://qemu-project.gitlab.io/qemu/system/gdb.html`
  - `https://www.kernel.org/doc/html/latest/dev-tools/gdb-kernel-debugging.html`

Linux kernel gdbstub

THE PENTEST EXPERTS.

- kdb/kgdb documentation: `https://www.kernel.org/doc/html/latest/dev-tools/kgdb.html`
- *From now on: focus on Linux kernel gdbstub*

Compiling the kernel

THE PENTEST EXPERTS.

- Source code: `https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git`
- Kernel build system: `https://docs.kernel.org/kbuild/index.html`

Install build dependencies

```
$ git clone --depth 1 --branch  \
    v5.10 git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
$ cd linux/

$ podman run -ti -v .:/data docker.io/debian:buster /bin/bash
$ cd /data

$ sudo apt -y install build-essential linux-config-5.10
$ sudo apt -y build-dep linux

$ # add source packages repositories (if not already there)
$ sed -i 'p; s/^deb/deb-src/' /etc/apt/sources.list
$ sudo apt update
```

Also see `https://www.kernel.org/doc/html/latest/process/changes.html`

Kernel compilation options (1)

THE PENTEST EXPERTS.

```
$ xzcat /usr/src/linux-config-5.10/config.amd64_none_amd64.xz > .config

$ scripts/config --enable CONFIG_DEBUG_INFO
$ scripts/config --enable CONFIG_FRAME_POINTER
$ scripts/config --enable CONFIG_GDB_SCRIPTS        # not necessary but useful
$ scripts/config --enable CONFIG_KDB_KEYBOARD
$ scripts/config --enable CONFIG_KGDB
$ scripts/config --enable CONFIG_KGDB_KDB
$ scripts/config --enable CONFIG_KGDB_SERIAL_CONSOLE
$ scripts/config --disable CONFIG_STRICT_KERNEL_RWX

$ make olddefconfig             # apply default options for undefined values
```

Build and install kernel

THE PENTEST EXPERTS.

```
$ make -j $(nproc) bindeb-pkg
$ make scripts_gdb

$ # Transfer built deb to debuggee and install there:
$ sudo dpkg -i linux-image-5.10.0_5.10.0-1_amd64.deb
```

Kernel boot options[1]

- `kgdboc`: io driver, kdbg over (serial) console
- `nokaslr`: disable KASLR

```
$ diff -u /etc/default/grub.orig /etc/default/grub
--- /etc/default/grub.orig      2023-06-13 12:54:30.284000000 +0200
+++ /etc/default/grub   2023-06-13 12:55:35.560000000 +0200
@@ -6,7 +6,7 @@
 GRUB_DEFAULT=0
 GRUB_TIMEOUT=5
 GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
-GRUB_CMDLINE_LINUX_DEFAULT="quiet"
+GRUB_CMDLINE_LINUX_DEFAULT="quiet kgdboc=ttyS0,115200 nokaslr"
 GRUB_CMDLINE_LINUX=""
```

[0] https://docs.kernel.org/admin-guide/kernel-parameters.html

Attach debugger

THE PENTEST EXPERTS.

On host: start gdb

```
$ gdb -q \
    -iex "add-auto-load-safe-path $PWD
    /vmlinux-gdb.py" \
    -ex "source $PWD/vmlinux-gdb.py" \
    -ex 'set serial baud 115200' \
    -ex 'target remote /dev/pts/1' \
    vmlinux
```

In VM: trigger Magic SysRq (SysRq-G)

```
echo g | sudo tee /proc/sysrq-trigger
```

Part II: Kernel exploit Dirty Pipe CVE-2022-0847

General information

- Kernel vulnerability in versions 5.8 to 5.16.11
- Discovered by Max Kellermann
- Allows arbitrary file write on readable files

Openwall mailing list post

THE PENTEST EXPERTS.

https://www.openwall.com/lists/oss-security/2022/03/07/1

Documentation

THE PENTEST EXPERTS.

- Official writeup (including proof of concept (PoC)): `https://dirtypipe.cm4all.com/`
- Another good source: `https://redhuntlabs.com/blog/the-dirty-pipe-vulnerability.html`
- For more information, specifically in the context of kernel debugging: My writeup as outsider: `https://github.com/stfnw/Debugging_Dirty_Pipe_CVE-2022-0847/blob/main/Debugging_Dirty_Pipe_CVE-2022-0847.pdf`

Fix in upstream source

THE PENTEST EXPERTS.

- Patch submitted to linux kernel mailing list (LKML):
  `https://lore.kernel.org/lkml/20220221100313.1504449-1-max.kellermann@ionos.com/`
- Commit in upstream source:
  `https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=9d2231c5d74e13b2a0546fee6737ee4446017903`
- Fix is literally only two lines

Proof of concept

THE PENTEST EXPERTS.

Proof of concept exploit: `https://dirtypipe.cm4all.com/#exploiting`

PoC: first run (1)

THE PENTEST EXPERTS.

Helper function for preparing non-writable file:

```
prepare_file() {
  sudo rm -rfv /tmp/tmp ; mkdir /tmp/tmp
  echo AAAAAA > /tmp/tmp/testfile
  sudo chmod 0444 /tmp/tmp/testfile
  sudo chown -R root:root /tmp/tmp
  sudo -K
  ls -al /tmp/tmp
  cat /tmp/tmp/testfile
}
```

PoC: first run (2)

First run of PoC:

```
$ prepare_file
removed '/tmp/tmp/testfile'
removed directory '/tmp/tmp'
total 12
drwxr-xr-x  2 root root 4096 Apr  1 07:17 .
drwxrwxrwt 11 root root 4096 Apr  1 07:17 ..
-r--r--r--  1 root root    7 Apr  1 07:17 testfile
AAAAAA
```

PoC: first run (3)

First run of PoC:

```
$ ./write_anything /tmp/tmp/testfile 1 BBBB
It worked!

$ cat /tmp/tmp/testfile
ABBBBA

$ ls -al /tmp/tmp/testfile
-r--r--r-- 1 root root 7 Apr  1 07:17 /tmp/tmp/testfile
```

Root problem

THE PENTEST EXPERTS.

General:

- ➔ Incorrect performance optimizations
- ➔ Data usage without reset / without correct initialization

Root problem

THE PENTEST EXPERTS.

General:

- ➔ Incorrect performance optimizations
- ➔ Data usage without reset / without correct initialization

More specifically:

- ➔ Some pointer first references data that is memory and appendable
- ➔ Pointer target is switched with memory backed by a file *but control information is not reset*, memory stays appendable
- ➔ Next write dirties the cache and propagates to disk

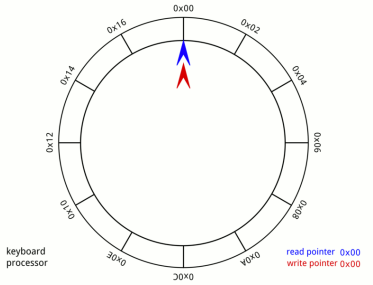
Root problem

General:

- ➔ Incorrect performance optimizations
- ➔ Data usage without reset / without correct initialization

More specifically:

- ➔ Some pointer first references data that is memory and appendable
- ➔ Pointer target is switched with memory backed by a file *but control information is not reset*, memory stays appendable
- ➔ Next write dirties the cache and propagates to disk



MuhannadAjjan, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

`https://upload.wikimedia.org/wikipedia/commons/f/fd/Circular_Buffer_Animation.gif`

Breakpoints for demonstration

THE PENTEST EXPERTS.

```
# printf write_an | hexdump -C: 77 72 69 74 65 5f 61 6e
hbreak iov_iter.c:409 if *(u64*)$lx_current(0).comm == 0x6e615f6574697277

print buf->flags        # => 16
# include/linux/pipe_fs_i.h: #define PIPE_BUF_FLAG_CAN_MERGE    0x10
#                            /* can merge buffers */
```

Summary

Kernel debugging via:

- gdbstub in QEMU/KVM
- gdbstub in the Linux kernel (kdb/kgdb)

CVE-2022-0847 Dirty Cow:

- What: arbitrary file write
- Root cause: incomplete reset of control information
- Arises as combination of:
  - memory management/caching
  - pipes
  - performance optimizations

THE PENTEST EXPERTS

WWW.SYSS.DE