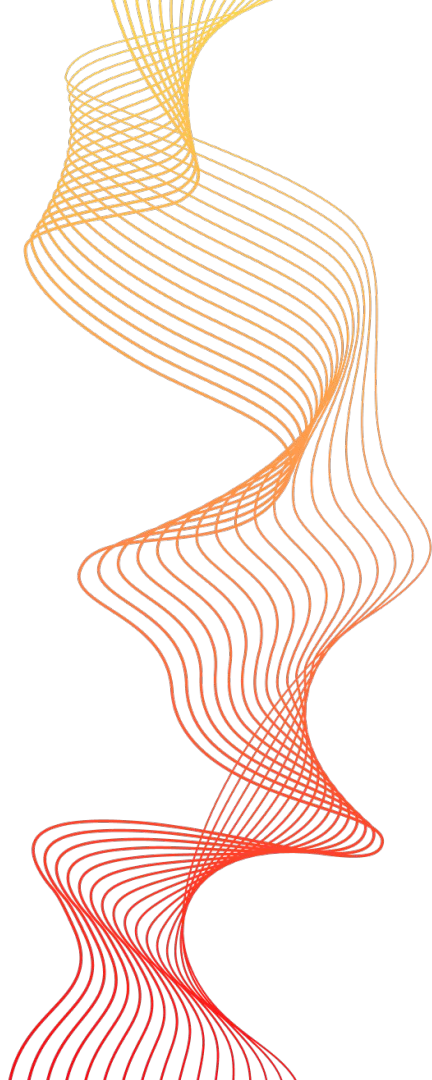




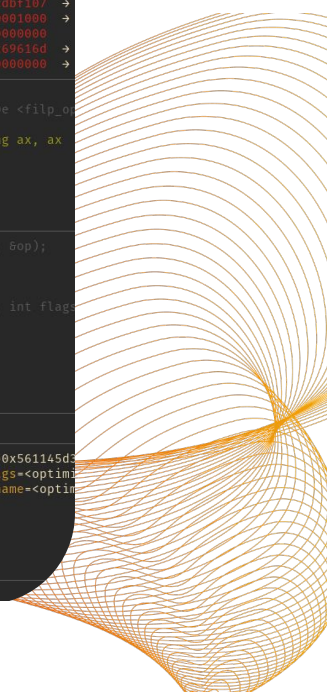
# Kernel Debugging for Linux

An introduction to kernel debugging techniques for identifying vulnerabilities in the Linux kernel or developing exploits.





- Introduction to kernel debugging techniques for identifying vulnerabilities or developing exploits
- Practical exercises using prepared kernel modules with vulnerabilities





# What is the Kernel?

- The kernel is a system program responsible for the main functions of the operating system
- From bootup to shutdown, the kernel controls hardware and loads applications
- The kernel has absolute authority in the system

```
Reached target Timers.
systemd[1]: Reached target Timers.
systemd[1]: Starting Journal Socket.
...ing on Journal Socket.
systemd[1]: Listening on Journal Socket.
systemd[1]: Starting dracut cmdline hook.
...ng dracut cmdline hook...
systemd[1]: Starting Journal Service...
...ng Journal Service...
...ed Journal Service.
systemd[1]: Started Journal Service.
...ing Create list of required static device nodes
...ing Setup Virtual Console...
...ning on udev Kernel Socket.[ 6.559659] s
...freed 0 bytes

...ning on udev Control Socket.
...ed target Sockets.
...ed target Swap.
...ed target Local File Systems.
...ed Create list of required static device nodes
...ing Create static device nodes in /dev...
...ed Create static device nodes in /dev.
...ed Setup Virtual Console.
```

[illegible]

Figure 1 displays a phylogenetic tree and sequence logos for the 1000 Genomes Project. The tree on the left illustrates the genetic relationships between various populations, categorized into African, European, East Asian, and South Asian groups. The logos on the right show the conservation of nucleotides for each population, with the top logo representing the combined population. The logos are color-coded by nucleotide: A (green), C (blue), G (red), and T (yellow).

- Access to system resources is controlled by the kernel based on permissions
- A vulnerability in the kernel can allow a hacker to act beyond their privileges

## [CVE-2022-1786] A Journey To The Dawn

📅 Posted on 2022-10-15 | 📅 Edited on 2022-10-19

I discovered a day vulnerability in the Linux kernel and exploited it on Google's kCTF platform. I contacted the Linux kernel security team and helped them fix the vulnerability.

The patching process was timely and smooth, which was not expected by me but quite nice.

I submitted my exploit to kCTF in May. After a pretty long waiting, I got the response that they had granted me the first full bounty (\$91,337) in kCTF's history (before the COVID pandemic).

I am grateful and really appreciate the recognition of my work!

COVID recovery (never regretted attending the DEF CON after-party btw, thank you for the headlines (on that note, stay tuned for some more Linux kernel insanity!)). Finally, I'm back then. Hopefully, this blog can inspire more people to join the Linux kernel community.

[CVE-2022-1786] is the first bug that I found, analyzed, exploited, and reported publicly. Thus, this blog will not be in the style of "what is the correct way to do it" but more of a celebration and excitement in the crazy 7 days that I spent developing the exploit.

Thank you for proofreading this blog. Really appreciate it!!

## Importance of Patching

- Device drivers that need to be executed with the privileges of the kernel are frequently developed and patched
- If a serious security problem is found in the kernel, it must be patched immediately

# Types of Kernel Debugging

Figure 1. Kernel Debugger

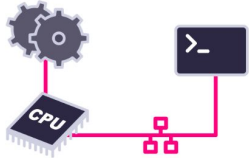


Figure 2. Hardware Debugger

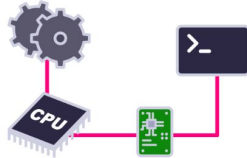
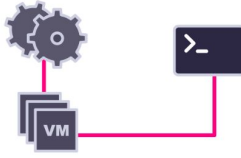


Figure 3. Virtual Machine Debugger



- Kernel Debugger - Issue debugging commands directly to the kernel.
- Hardware Debugger - Directly query and change the state of the circuit through JTAG, etc. used for hardware inspection. To use this method, separate debugging equipment is required.
- Virtual Machine Debugger - A way to utilize a virtual machine instead of running the kernel directly on top of hardware. No special debugging equipment is required.



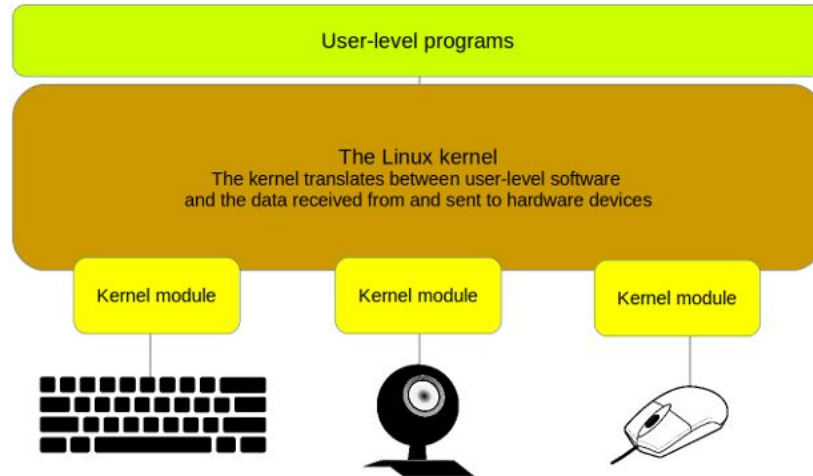


# Loadable Kernel Modules (LKM)

LKM allows necessary functions to be attached or detached from the kernel to improve scalability

Kernel modules, when attached to the kernel, use the same

Vulnerabilities in kernel modules can lead to kernel



# User Space and Kernel Space

- The kernel divides memory into user space and kernel space
- Running processes can only access user space in user mode
- In kernel mode, the processor has access to both user space and kernel space







**Thank you for your time and attention 😊**