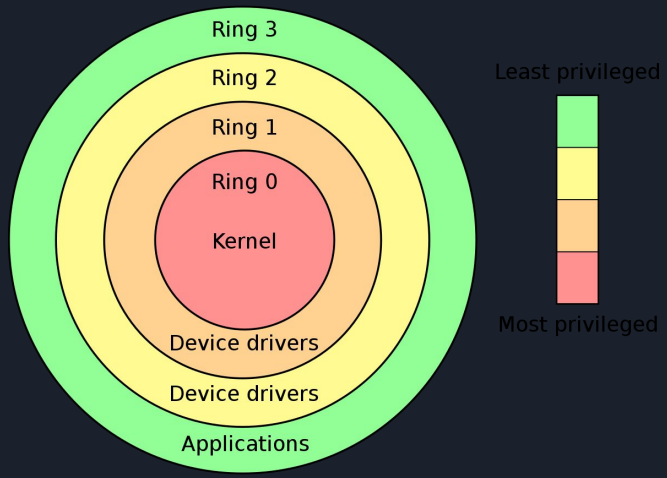
Linux Kernel Pwn

What is a kernel?

- Interacts directly with the hardware
- Provide an environment to run applications



Services Provided by the kernel

- Filesystem
- Syscalls
- Processes/Threads
- Privilege levels
- Network I/O

Kernel Exploitation Use Cases

- Jailbreaking/Rooting devices
- Privilege escalation

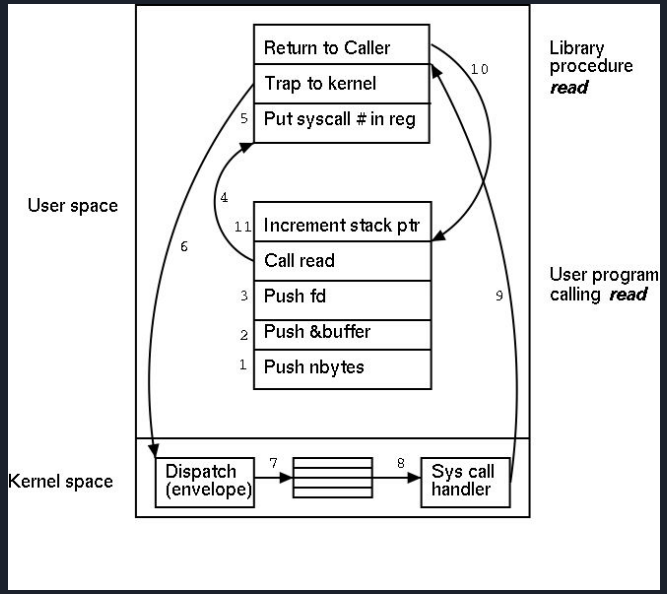
Potential Attack Surfaces?

- The actual kernel
- Device drivers/kernel modules

Communicating w/ the Kernel

- syscalls: user space program asks the kernel to do something that requires higher privileges(read/write/exec etc)
- ioctl: specific syscall for communicating w/ device

How syscalls work



Kernel Exploit Mitigations

- SMEP: executing code in user space when the processor is in ring0 will generate a page fault
- SMAP: basically SMEP for fetching data
- mmap_min_addr: cannot map addresses below this

NULL Pointer Overwrite

- The kernel uses a lot of function pointers
- Potentially bugs where uninitialized function pointers are called
- A malicious program can simply map shellcode into address 0 and trigger the null function pointer bug to run arbitrary code in the kernel

Exploiting a Privesc Vuln

1. Find vuln in the kernel that allows us to run arbitrary code
2. Use vuln to elevate our process' privilege level (set uuid to 0)
3. Switch back to userspace

Elevating Privileges

- The kernel keeps track of a processes privileges in a cred structure

https://code.woboq.org/linux/linux/include/linux/cred.h.html#cred

- The kernel provides a helper function called commit_creds() to change the credentials of the current process

Structure of a CTF kernel pwn problem

- rootfs.cpio: filesystem image
- bzImage: kernel binary
- boot.sh: script for starting the pwnable using qemu

Demo time