Device Drivers Overview

Advanced Embedded Linux Development

with **Dan Walkes**

University of Colorado **Boulder**

Learning objectives:

Identify Common Linux Device Classes

Driver Security Concerns

Kernel Versioning and Licensing

Device Classes

- Devices belong to one of a set of classes
- Classes share common access code and methods
- Classes:
  - Character
  - Block
  - Network

Character Device Class

- Accessed as a stream of bytes
  - Look like a file to user space applications
  - open, close, read, write
- /dev/console
- /dev/ttyS0
- May or may not be possible to seek or map memory like you can do with real files.

Block Device Class

- Device (disk) which can host a filesystem
- Transfers are always on block boundaries at the device level
  - Usually 512 bytes or larger
- Linux device drivers allow you to access at less than 1 block sizes.
  - Kernel manages split blocks for you
- /dev/sda1

Network Interface Class

- May be a hardware device.
- May also be a software device like loopback interface.
- Handles packets.
- Uses a name like eth0 instead of a device endpoint.

Network Interface Class

- Does not have an entry in the filesystem like char or block drivers.
  - Why not?
  - Typically the use case and packet IO doesn’t map well to read() and write()

Linux Device Drivers 3rd Edition Chapter 1
https://unix.stackexchange.com/a/23796/124912

Kernel Filesystems

- Maps low level blocks on disk to directories and files.
- Independent of the data transfer mechanism used to transfer blocks to and from the disk (SATA, USB, SD card)
  - Uses devices and device drivers to perform transfers.

Security Concerns

- Security checks are ultimately enforced by kernel code
    - kernel becomes critical point of exploit
- Only authorized user can load modules

Security Concerns

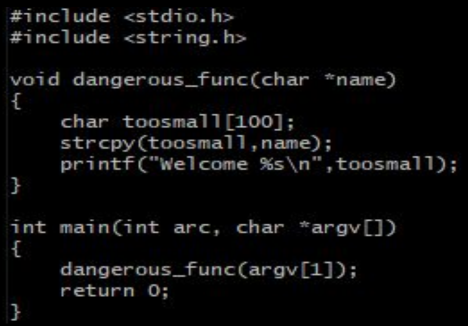
- Drivers typically do not encode security policy
    - System admin controls based on permissions of device associated with driver.
    - Exceptions: global resources changes (interrupt line, firmware update)

Security Best Practices

- Don’t trust user data without verifying
    - Don’t allow anything the user sends to write any areas of memory it should not.
- Zero memory obtained in the kernel before passing outside
    - Avoids information leakage.
- Length check user space data before copying
    - Avoids buffer overrun

Buffer overrun

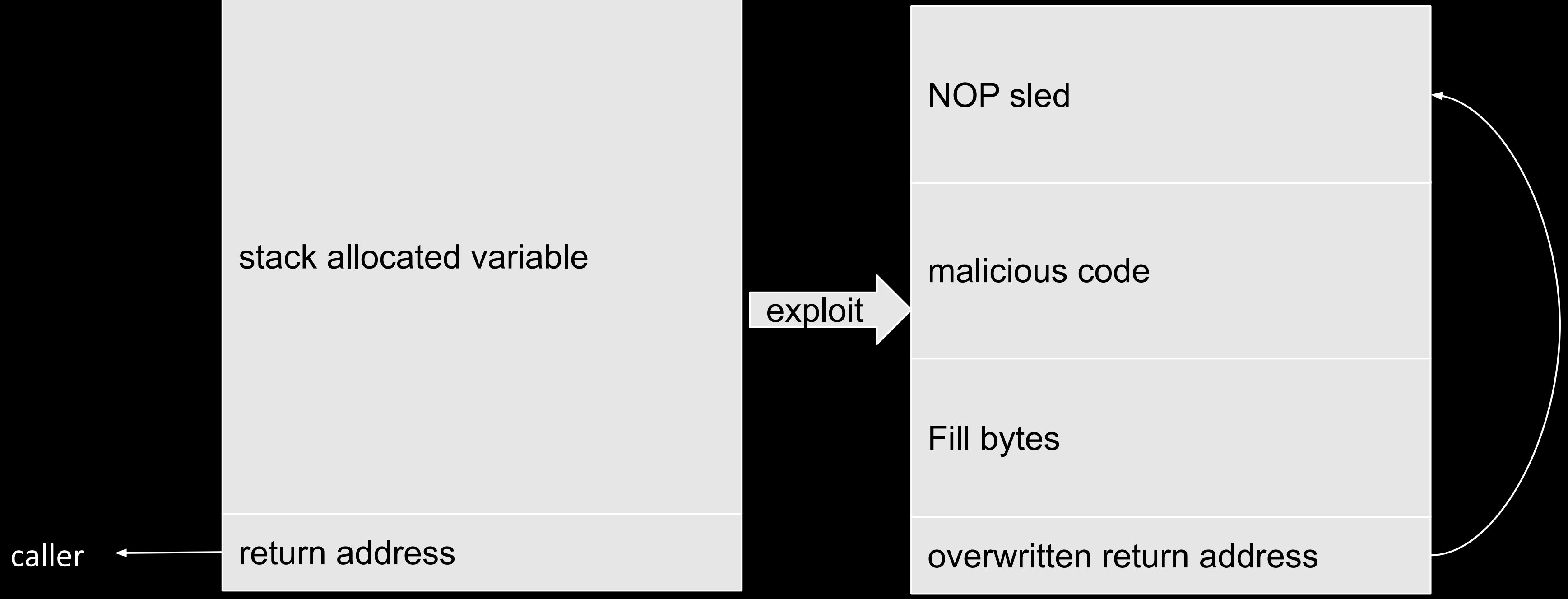
- What is a buffer overrun error?



```
#include <stdio.h>
#include <string.h>

void dangerous_func(char *name)
{
      char toosmall[100];
      strcpy(toosmall,name);
      printf("Welcome %s\n",toosmall);
}

int main(int arc, char *argv[])
{
      dangerous_func(argv[1]);
      return 0;
}
```



stack allocated variable

return address

caller

exploit

NOP sled

malicious code

Fill bytes

overwritten return address

https://www.coengoedegebure.com/buffer-overflow-attacks-explained/

Kernel Versioning

- The kernel even/odd numbering scheme discussed in the book is out of date
    - Now uses a simpler major.minor
    - Not using feature based releases, based on time and intending to simplify version numbers.
    - Why?
        - Linus was “running out of fingers and toes”

https://lkml.org/lkml/2015/4/12/178

Licensing

- GNU General Public License GPL version 2
  - Anyone can redistribute and sell as long as the recipient has ***access to the source*** and is ***able to exercise the same rights***.
  - Any software product ***derived from*** a product covered by the GPL, ***if redistributed***, must be released under the GPL.

Licensing

- Allow growth of knowledge while allowing commercial use.
- Can a kernel module be shared only in binary form?
  - “Deliberately ambiguous” - “kernel developers have no qualms against breaking binary modules between kernel releases”
  - Seek legal advice