It's a Unix system! I know this!

HEART Computer Security and Privacy for the Modern World

💙 `EN.600.111(33)`

September 16, 2021

Tushar Jois

It's a Unix system! I know this!

HEART Computer Security and Privacy for the Modern World

🧡 `EN.600.111(26)`

September 22, 2021

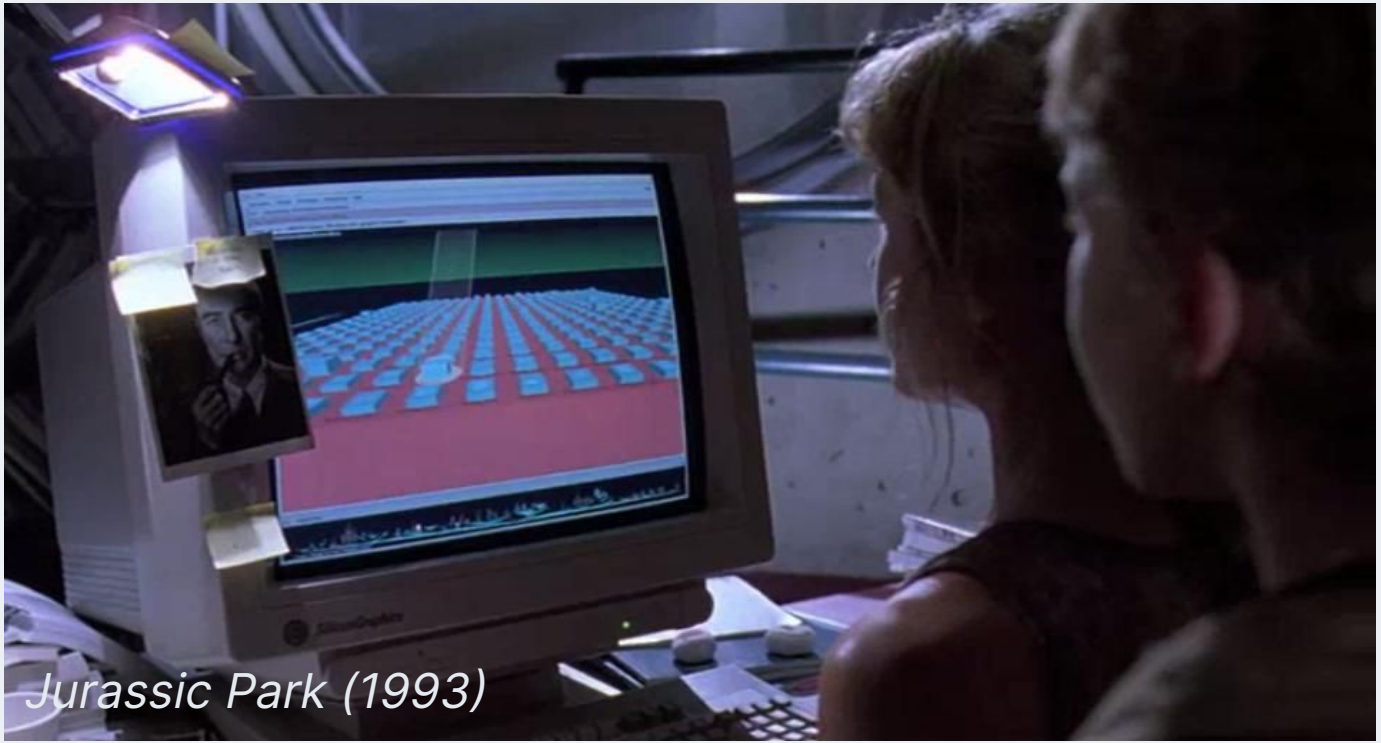
Tushar Jois

Recap

- Computers are pervasive, connected, and faster than ever before
- Trusting code executed on a computer requires trusting its developer
- Minimizing trust is at the core of a real-world threat model

Lesson objectives

- Identify user, process, and kernel isolation in Unix
- Understand, intuitively, how a buffer overflow attack works
- Recognize the importance of the principle of least privilege and user input sanitization in systems security

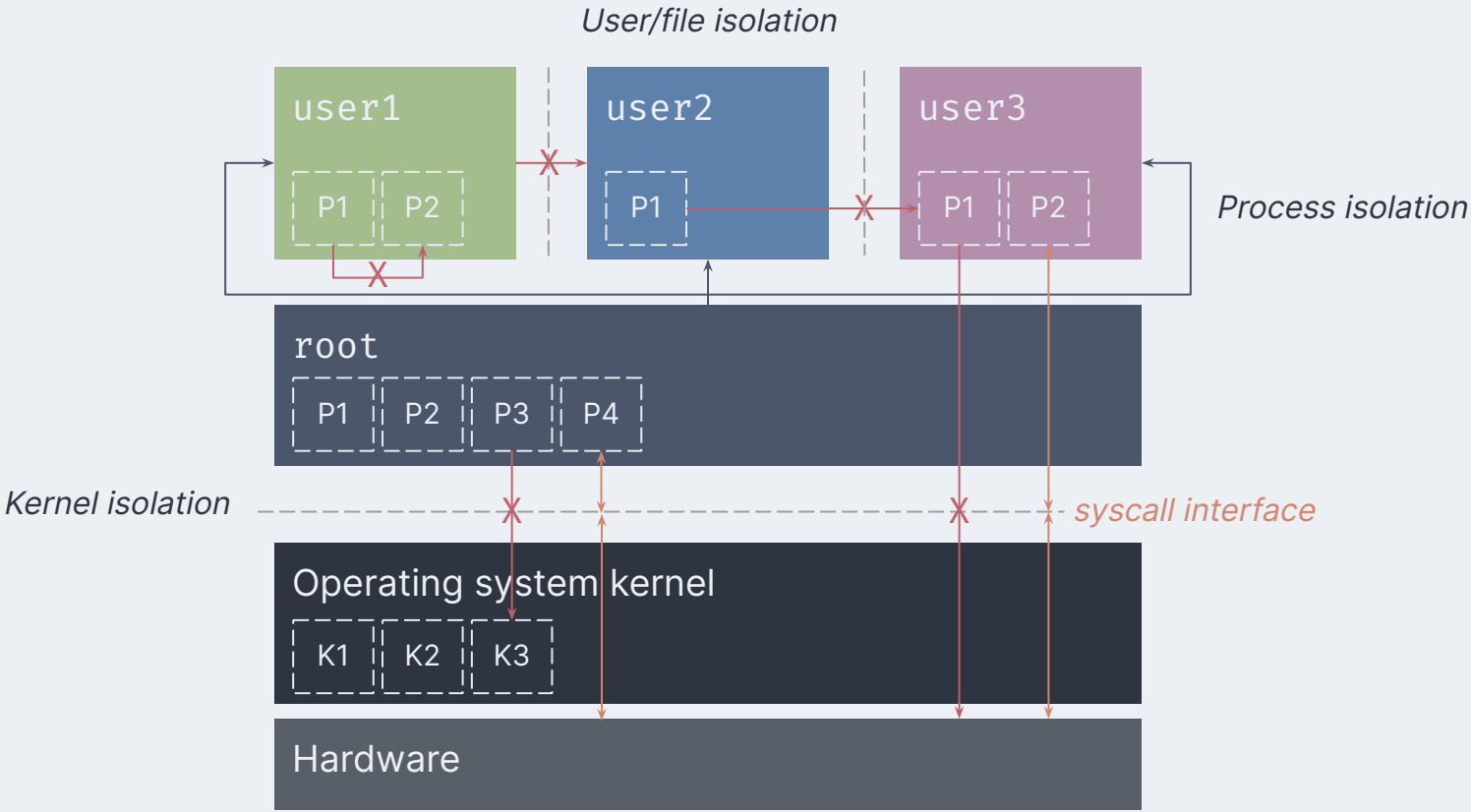


Jurassic Park (1993)

Unix

- Developed in the 1970s by Ken Thompson and Dennis Ritchie (and others) at Bell Labs
- Originally ran on PDP-11 minicomputers as a single-user operating system
  - Ported to several architectures
  - Multi-tasking and multi-user support
- “Unix philosophy”
  - “Make each program do one thing well”
- Modern Unix derivatives: Linux (Android), Darwin (macOS), Solaris
  - POSIX Standard

Unix isolation



minimizing trust

The principle of least privilege

“Every program and every privileged user of the system should operate using the least amount of privilege necessary to complete the job.”

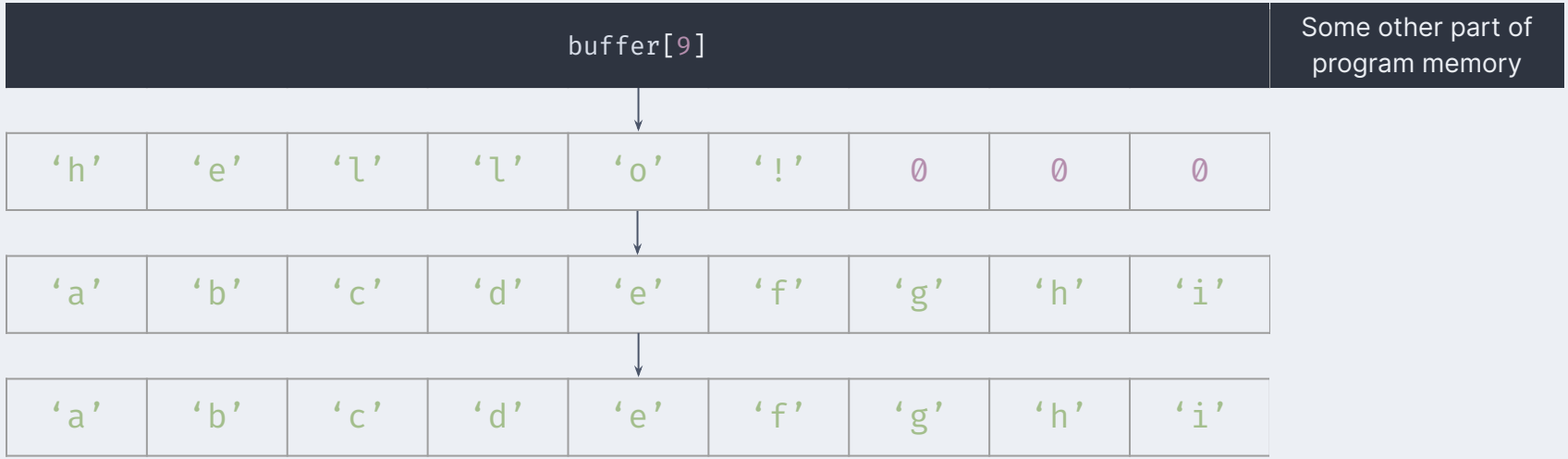
Jerome Saltzer, Communications of the ACM (1974)

Systems security is all about **code** providing defense against malicious activity.

(the code has to actually work)

C

```
if (access(user, resource) ≠ ACCESS_OK) {
    panic("user cannot access resource!");
} else {
    use(user, resource);
}
```




```
char buffer[9] = {0};

memcpy(buffer, "hello!", 6);

memcpy(buffer, "abcdefghij", 9);

memcpy(buffer, "abcdefghij", 10);

memcpy(buffer, "abcdefghijklmnopqrstuvwxyz", 26);
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:
        Index out of bounds for length 9
        at Buffer.main(Buffer.java:4)
```

```
Program terminated with signal SIGSEGV, Segmentation fault
```

Process isolation

Buffer overflow attack

Can overwrite data past the buffer to change the control flow!

Stores information about the order of program execution (**control flow**)

part of
mory

‘h’	‘e’	‘l’	‘l’	‘o’	‘!’	0	0	0	191	22

“the stack”

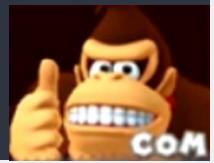
program data | program info

‘a’	‘b’	‘c’	‘d’	‘e’	‘f’	‘g’	‘h’	‘i’	‘j’	22

106

```
char buffer[9] = {0};
memcpy(buffer, "abcdefghij", 10);
```

```
if (access(user, resource) ≠ ACCESS_OK) {
   panic("user cannot access resource!");
} else {
   use(user, resource);
}
```



Exploitation steps:

1. Identify your target buffer
2. Figure out where we want to go
3. Overwrite the buffer to change the control flow of the program
4. The program then moves to where we want it to!

Let’s try it!

Shellcode

For when you don’t know where you want to go

- Instead of moving to a specific place in the program, spawn a new *shell*
  - A place to send commands
- The new shell will spawn with the *same permissions* as the application running
  - If it’s `root` → you’re done

minimizing trust

Memory protection

Use stack canaries, a non-executable stack, and randomization to prevent buffer overflow attacks (*all enabled by default*).

minimizing trust

User input sanitization

“Input validation is a programming technique that ensures only properly formatted data may enter a software system component.”

OWASP Top Ten Protective Controls (2018)

Looking ahead

- Don’t worry if you didn’t fully get the buffer overflow details!
  - Review the slides when posted on the website
- Do the reading: *Nine Algorithms that Changed the Future*, Chs. 4 & 9
  - If you can, click through the optional “The Illustrated TLS Connection”
  - Bring a Unix laptop (or the course VM) to class next time!
    - Email me if you have trouble: jois@cs.jhu.edu
- Research information → more next time
- Exit ticket before class is done

Lesson objectives

- Identify user, process, and kernel isolation in Unix
- Understand, intuitively, how a buffer overflow works
- Recognize the importance of the principle of least privilege and user input sanitization in systems security