# CS 576 – Systems Security

## Introduction to (Software) Systems Security

Georgios (George) Portokalidis

# What Is Software Systems Security About?

# What Is Software Systems Security About?

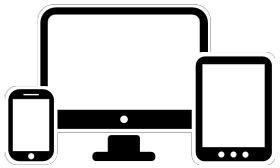
- **Software Bugs**

A software bug is an error, flaw or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways

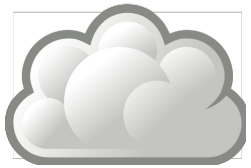
-- Wikipedia

# Software is Everywhere

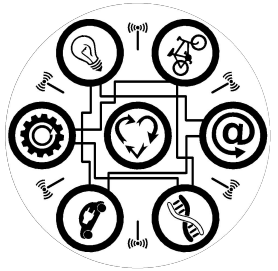
**Personal Computers**



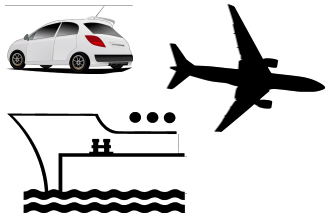
**Cloud**



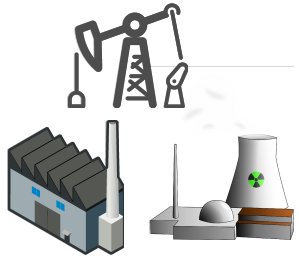
**IoT**



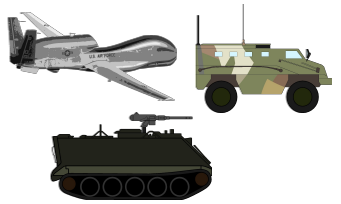
**Vehicles**



**Industrial Systems**



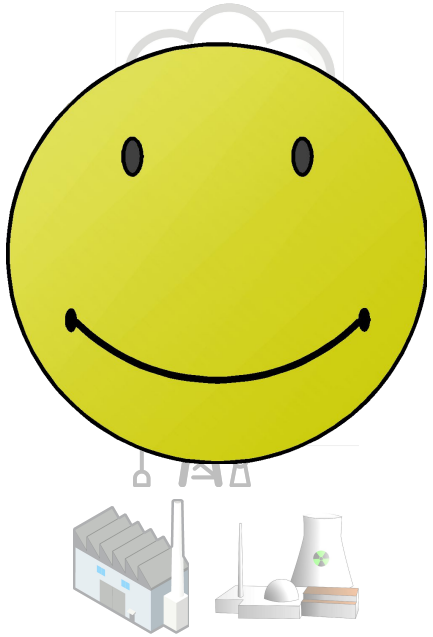
**Defense**



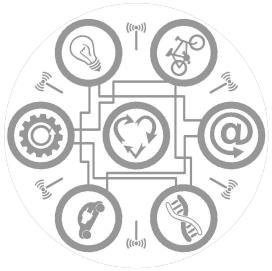
# Software is Everywhere

Personal Computers

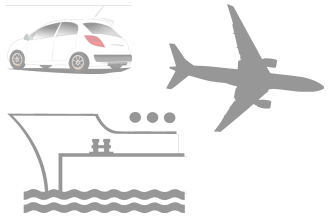
Cloud



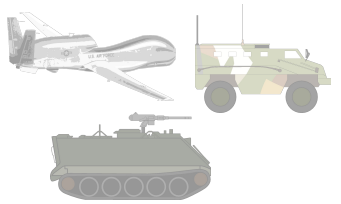
IoT



Vehicles



Defense



# Not all Good


The Atlantic
Popular
Latest
Sections
Magazine
More
Subscribe
The Coming Software Apocalypse
A small group of programmers wants to change how we code—before catastrophe strikes.
Lynn Scurfield
JAMES SOMERS | SEP 26, 2017 | TECHNOLOGY
Share
Tweet
TEXT SIZE
– +

# Software Bugs with Extreme Consequences

- **World War 3**
  - In 1980, NORAD reported that the US was under missile attack. The problem was caused by a faulty circuit, a possibility the reporting software hadn't taken into account.
  - In 1983, a Soviet satellite reported incoming US missiles, but the officer in charge decided to follow his gut feeling that it was a false alarm and decided to do nothing.

https://www.pingdom.com/blog/10-historical-software-bugs-with-extreme-consequences/

# Software Bugs with Extreme Consequences

- **Deadly radiation therapy**
  - The Therac-25 medical radiation therapy device was involved in several cases where massive overdoses of radiation were administered to patients in 1985-87, a side effect of the buggy software powering the device.
  - Another radiation dosage error happened in Panama City in 2000, where therapy planning software from US company Multidata delivered different doses depending on the order in which data was entered.

https://www.pingdom.com/blog/10-historical-software-bugs-with-extreme-consequences/

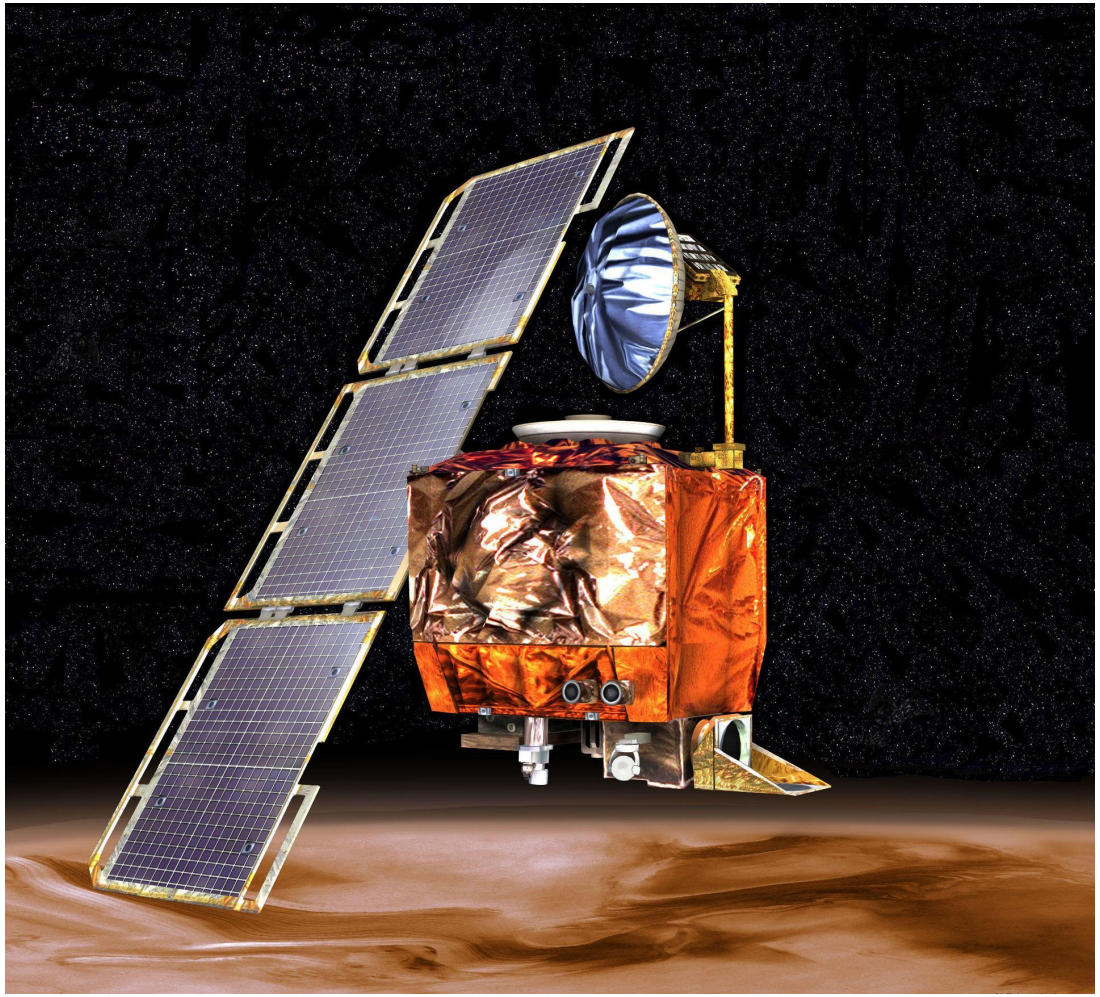
# Software Bugs with Extreme Consequences

- **Rocket launch errors**
  - In 1996, a European Ariane 5 rocket was set to deliver a payload of satellites into Earth orbit, but problems with the software caused the launch rocket to veer off its path a mere 37 seconds after launch. As it started disintegrating, it self-destructed (a security measure).


https://www.pingdom.com/blog/10-historical-software-bugs-with-extreme-consequences/

# Software Bugs with Extreme Consequences

- **Lost in space**
  - One of the subcontractors NASA used when building its Mars climate orbiter had used English units instead of the intended metric system, which caused the orbiter's thrusters to work incorrectly. Due to this bug, the orbiter crashed almost immediately when it arrived at Mars in 1999.



https://www.pingdom.com/blog/10-historical-software-bugs-with-extreme-consequences/

# What Is Software Systems Security About?

- Software Bugs
- **Vulnerabilities**

In computer security, a vulnerability is a weakness which can be exploited by a threat actor, such as an attacker, to cross privilege boundaries (i.e. perform unauthorized actions) within a computer system.

-- Wikipedia

# What Is Software Systems Security About?

- Software Bugs
- **Vulnerabilities**

In computer security, a vulnerability is a weakness which can be exploited by a threat actor, such as an attacker, to cross privilege boundaries (i.e. perform unauthorized actions) within a computer system.

-- Wikipedia

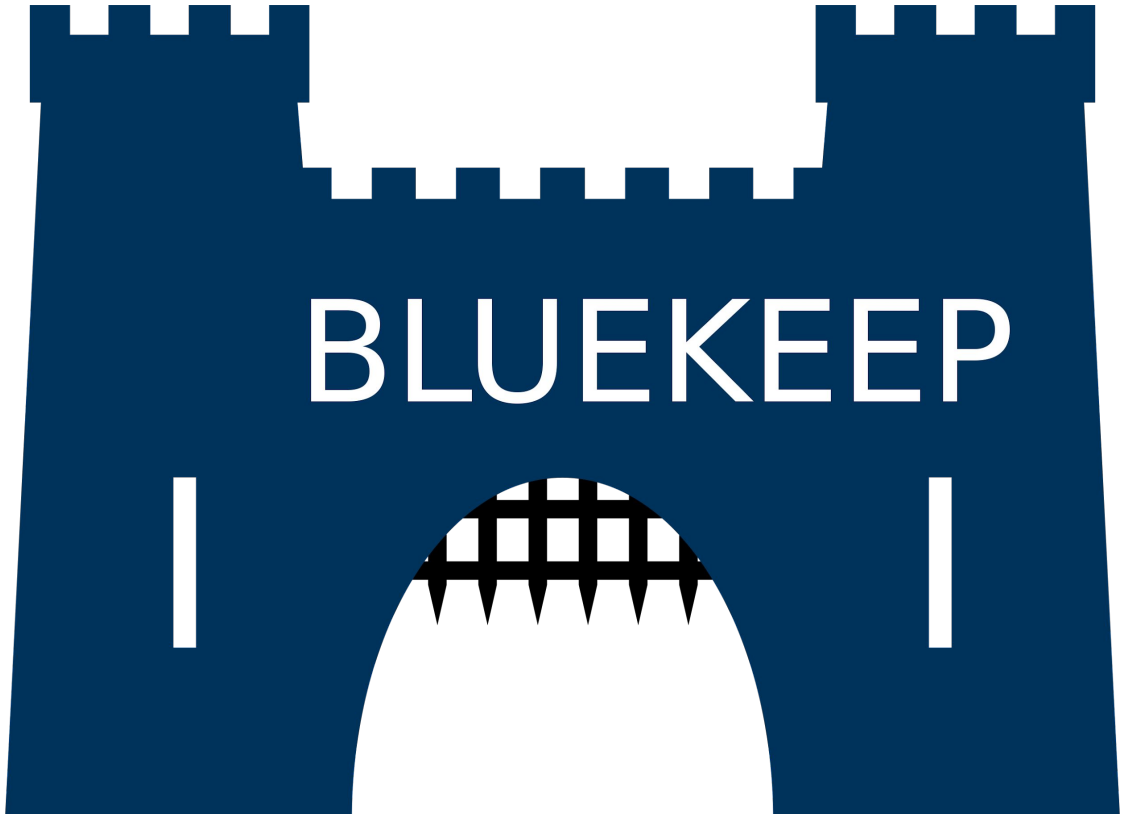
A software bug becomes a vulnerability when it can be exploited to perform unauthorized actions in a system.

# Bad Vulnerabilities

- **CVE-2019-0708 (BlueKeep)**
  - Part of the costliest attacks in history so far, WannaCry and NotPetya both used Eternal Blue-style attacks as part of their payloads. Source: Microsoft
  - Exploits bug in SMB -- Windows' file, printer, etc. sharing
    - A frequent source of vulnerabilities
  - One of the first ransomware

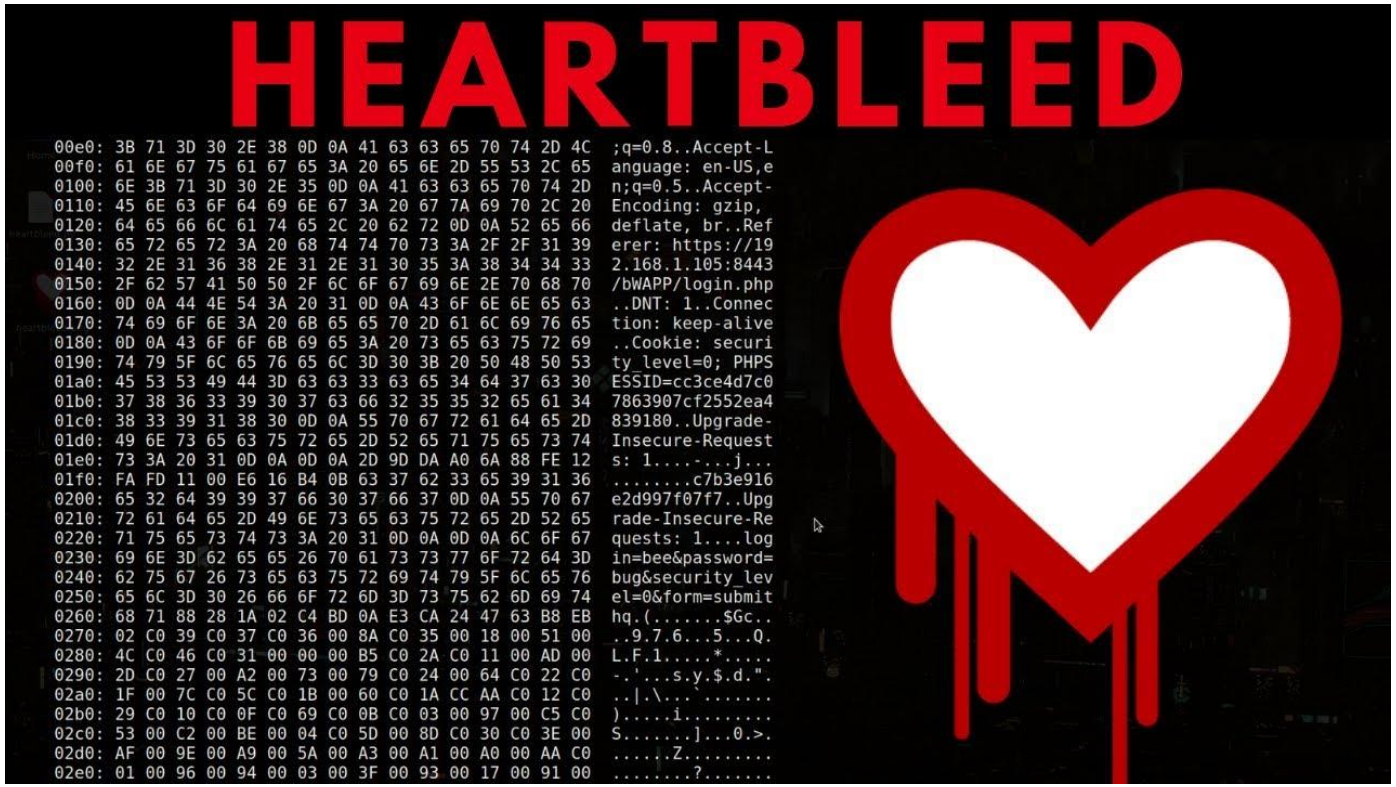
# Bad Vulnerabilities

- **MS17-010 (Eternal Blue)**
  - Despite multiple warnings, it took until November 2019 for the first exploits of Bluekeep to be spotted. It has been predicted that a widespread exploit could be severe.
  - A security vulnerability that was discovered in Microsoft's Remote Desktop Protocol (RDP) implementation



# Bad Vulnerabilities

- **CVE-2014-0160 (Heartbleed)**
  - Heartbleed is a vulnerability in the OpenSSL code that handles the Heartbeat extension for TLS/DTLS.



# What Is Software Systems Security About?

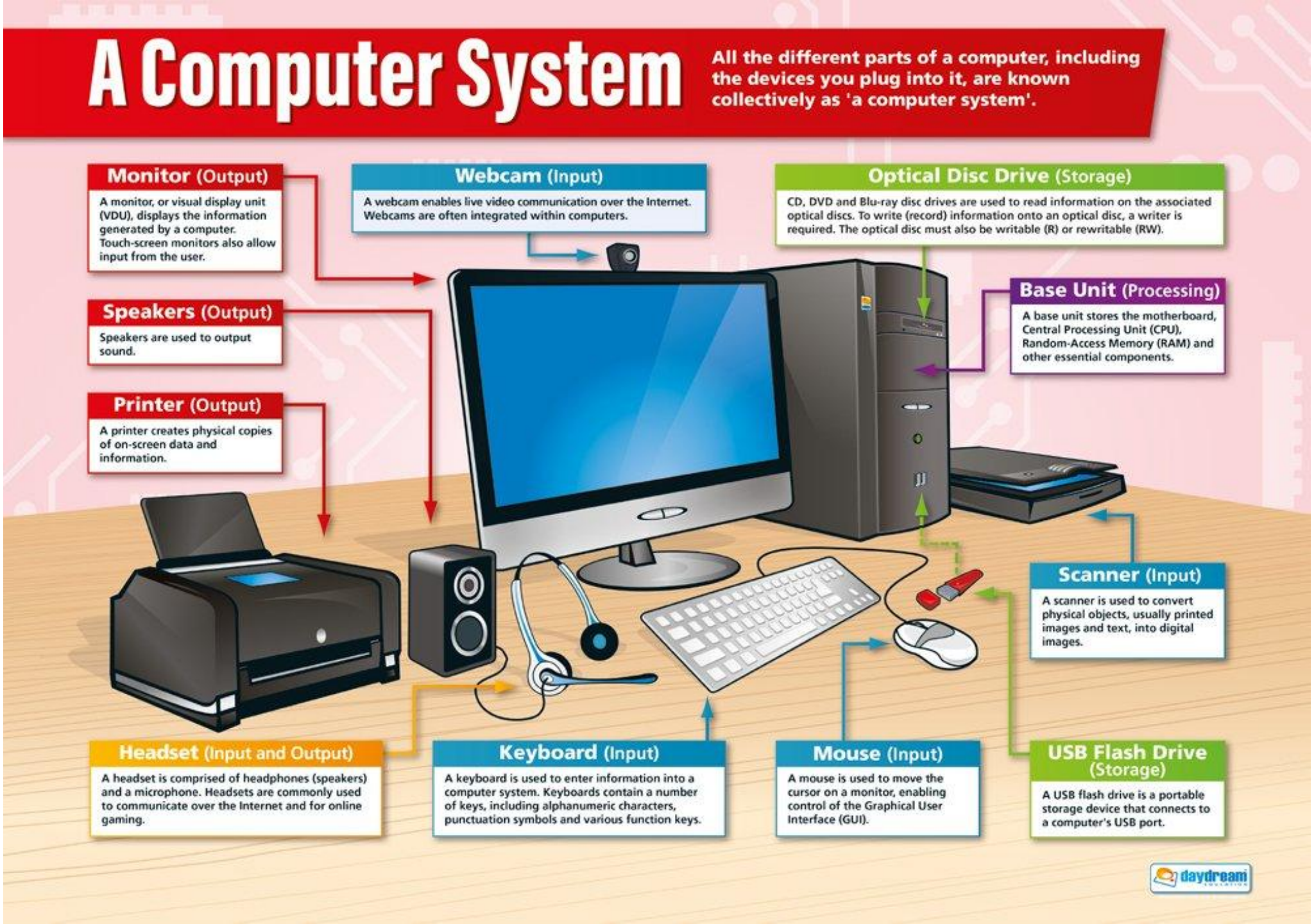
- Software Bugs
- Vulnerabilities
- **Computer Systems**

A computer system is defined as the combination of hardware, software, user and data.

-- Wikipedia

# A Computer System

All the different parts of a computer, including the devices you plug into it, are known collectively as 'a computer system'.



**Monitor (Output)**
A monitor, or visual display unit (VDU), displays the information generated by a computer. Touch-screen monitors also allow input from the user.

**Webcam (Input)**
A webcam enables live video communication over the Internet. Webcams are often integrated within computers.

**Optical Disc Drive (Storage)**
CD, DVD and Blu-ray disc drives are used to read information on the associated optical discs. To write (record) information onto an optical disc, a writer is required. The optical disc must also be writable (R) or rewritable (RW).

**Speakers (Output)**
Speakers are used to output sound.

**Base Unit (Processing)**
A base unit stores the motherboard, Central Processing Unit (CPU), Random-Access Memory (RAM) and other essential components.

**Printer (Output)**
A printer creates physical copies of on-screen data and information.

**Scanner (Input)**
A scanner is used to convert physical objects, usually printed images and text, into digital images.

**Headset (Input and Output)**
A headset is comprised of headphones (speakers) and a microphone. Headsets are commonly used to communicate over the Internet and for online gaming.

**Keyboard (Input)**
A keyboard is used to enter information into a computer system. Keyboards contain a number of keys, including alphanumeric characters, punctuation symbols and various function keys.

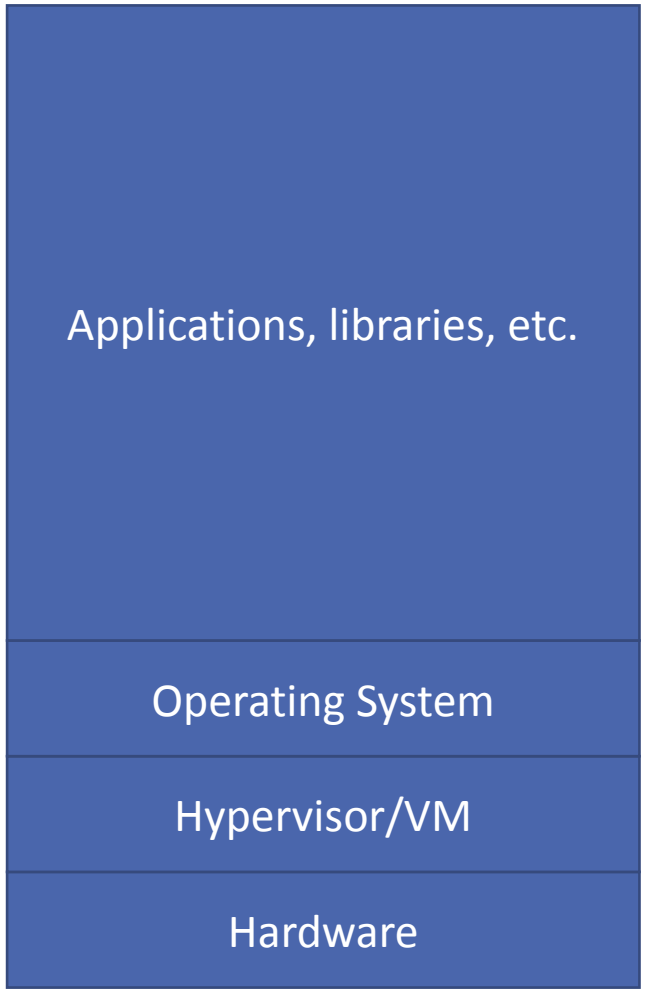
**Mouse (Input)**
A mouse is used to move the cursor on a monitor, enabling control of the Graphical User Interface (GUI).

**USB Flash Drive (Storage)**
A USB flash drive is a portable storage device that connects to a computer's USB port.

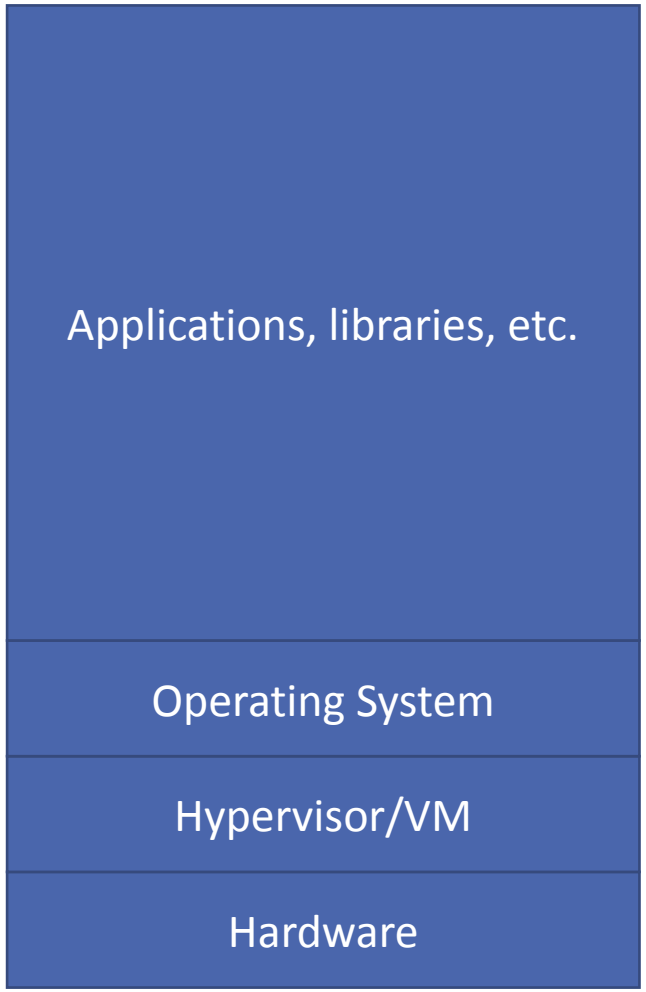
daydream


Swipe up to unlock
12:45
Monday 20 September

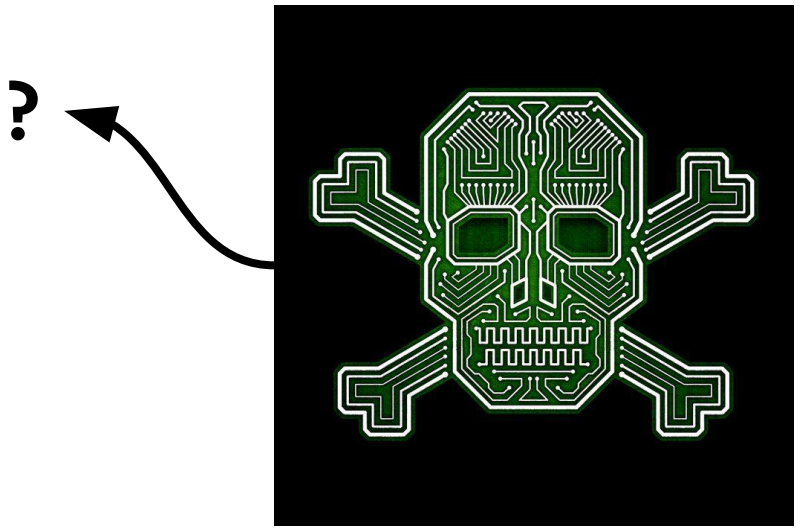
# Computer Systems



# Computer Systems



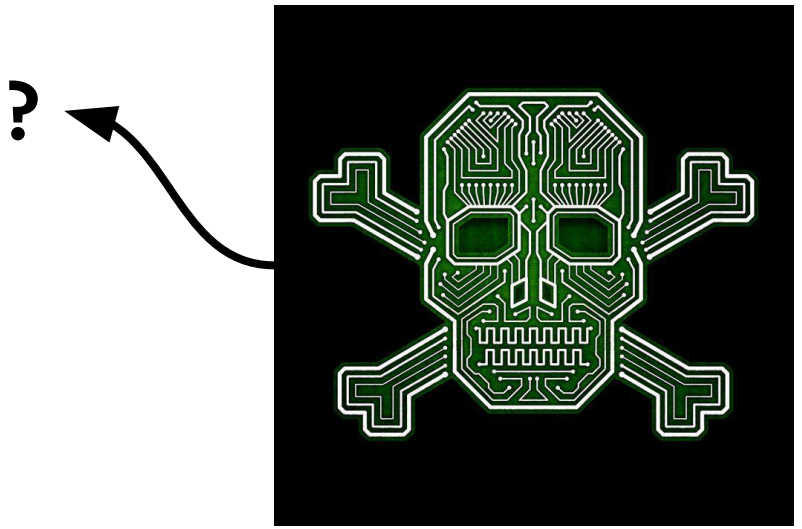

Where do vulnerabilities live?



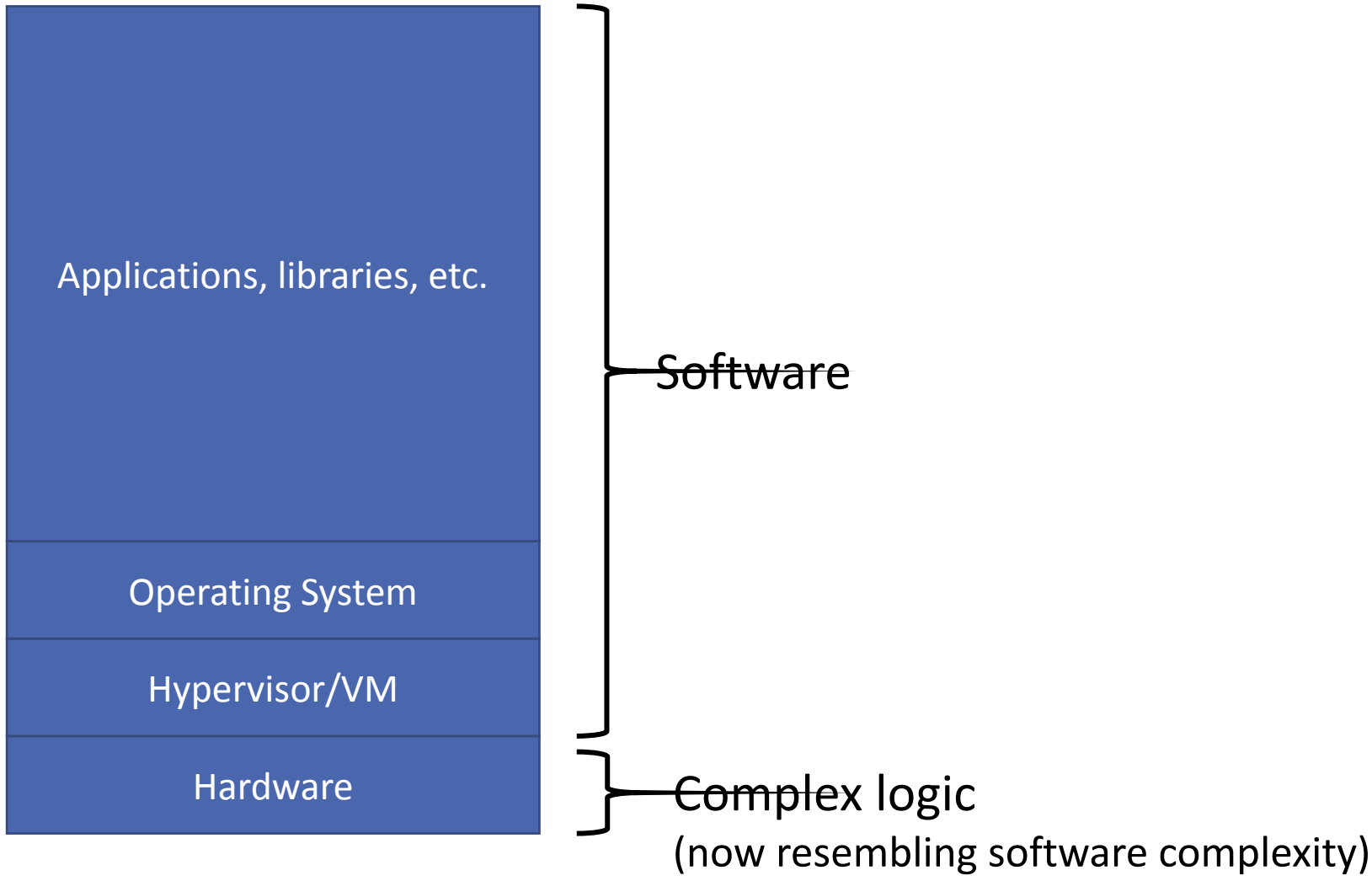
# Computer Systems

Everywhere!

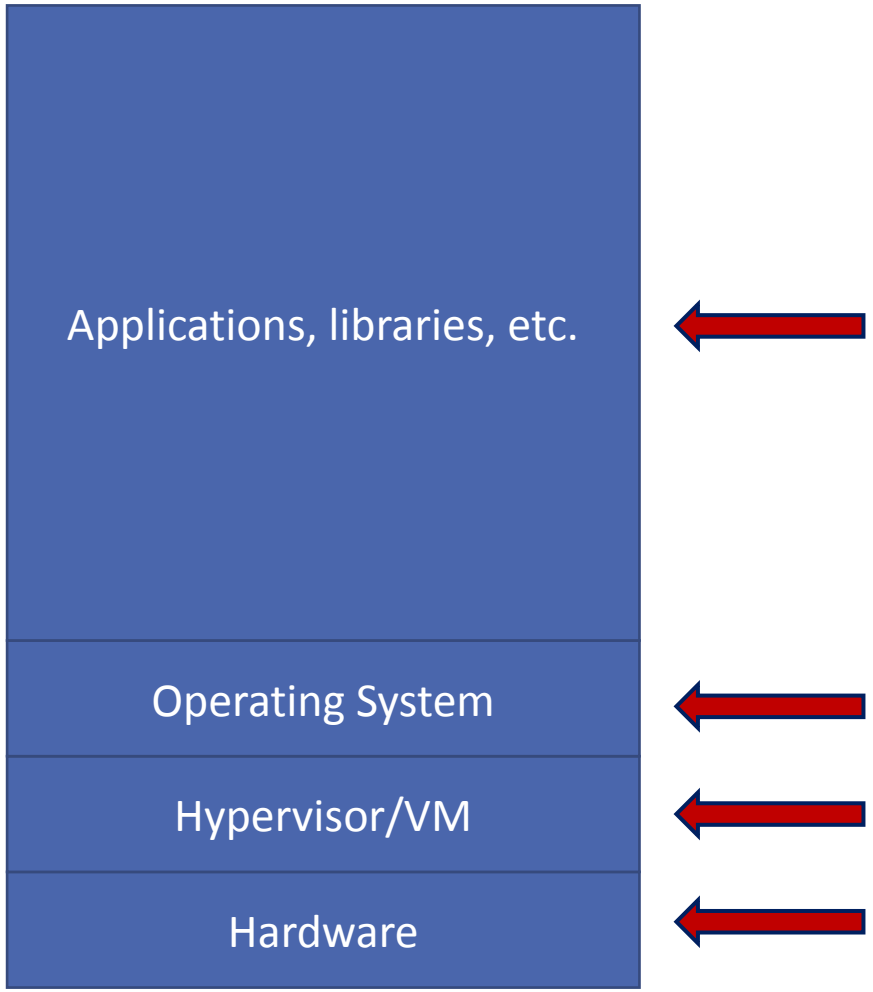
Where do vulnerabilities live?



# Computer Systems

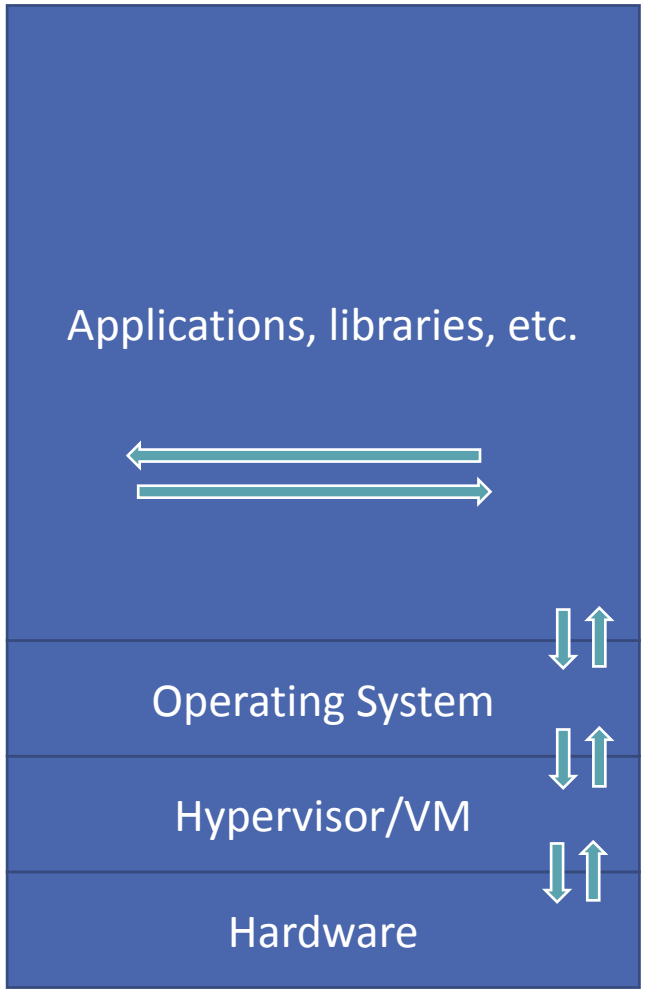


# Computer Systems



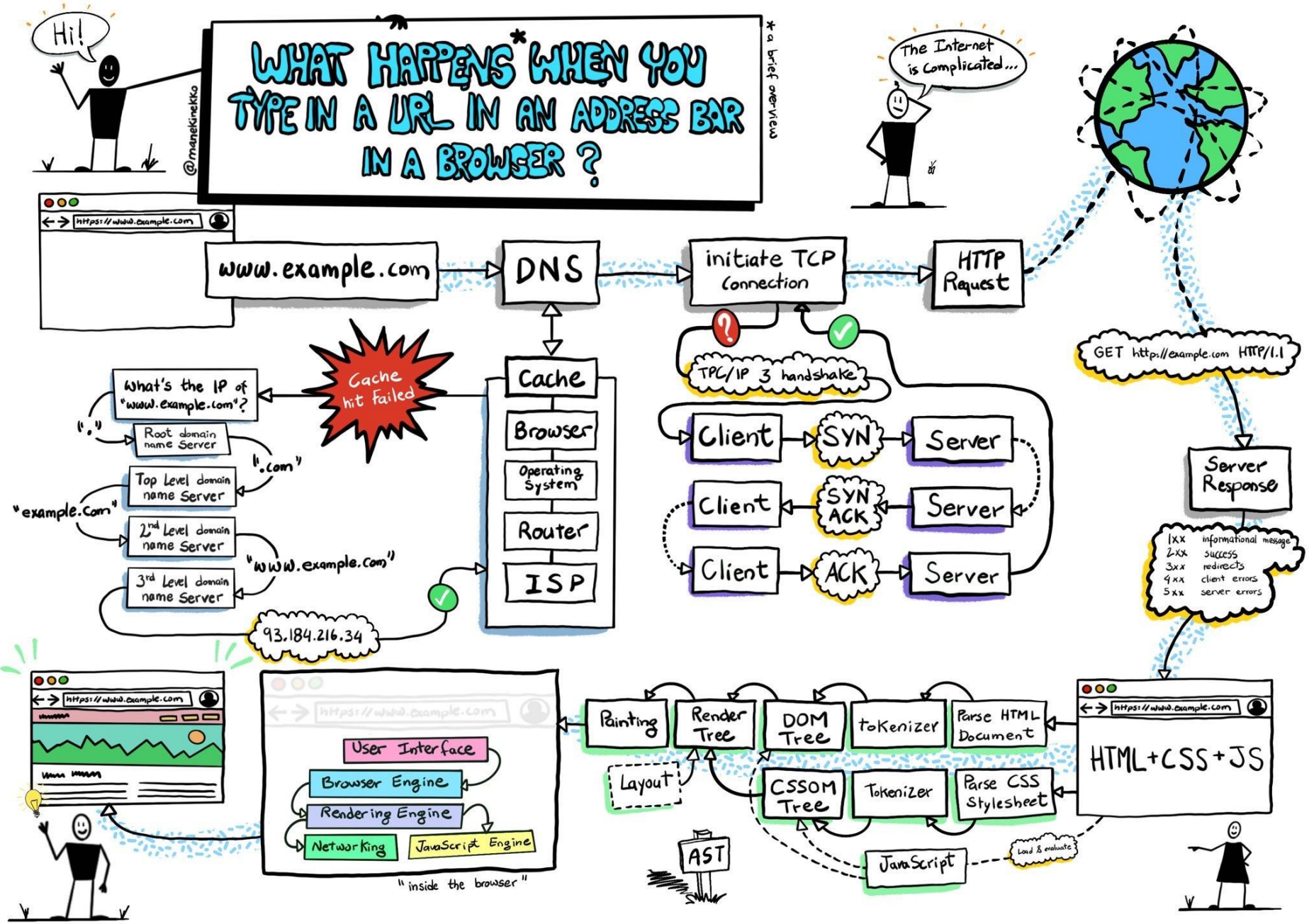

Bugs/vulnerabilities
**can be found in each layer**

# Computer Systems




Bugs/vulnerabilities
**can occur because of layer interactions**
**or**
**Interactions within the layer**

# Networked Systems


Hi!
@manekinekko
WHAT HAPPENS* WHEN YOU TYPE IN A URL IN AN ADDRESS BAR IN A BROWSER ?
* a brief overview
The Internet is complicated...
https://www.example.com
www.example.com
DNS
initiate TCP Connection
HTTP Request
GET http://example.com HTTP/1.1
Server Response
1xx informational message
2xx success
3xx redirects
4xx client errors
5xx server errors
TPC/IP 3 handshake
Client
SYN
Server
Client
SYN ACK
Server
Client
ACK
Server
Cache
Browser
Operating System
Router
ISP
Cache hit Failed
What's the IP of "www.example.com"?
"."
Root domain name Server
".com"
Top Level domain name Server
"example.com"
2nd Level domain name Server
"www.example.com"
3rd Level domain name Server
93.184.216.34
HTML+CSS+JS
Parse HTML Document
tokenizer
DOM Tree
Render Tree
Painting
Layout
Parse CSS Stylesheet
Tokenizer
CSSOM Tree
JavaScript
Load & evaluate
AST
User Interface
Browser Engine
Rendering Engine
Networking
JavaScript Engine
" inside the browser "

# What Is Software Systems Security About?

- Software Bugs
- Vulnerabilities
- Computer Systems
- **System Software**

**System software** is software designed to provide a platform for other software. Examples of system software include operating systems (OS) like macOS, GNU/Linux, Android and Microsoft Windows, computational science software, game engines, search engines, industrial automation, and software as a service applications

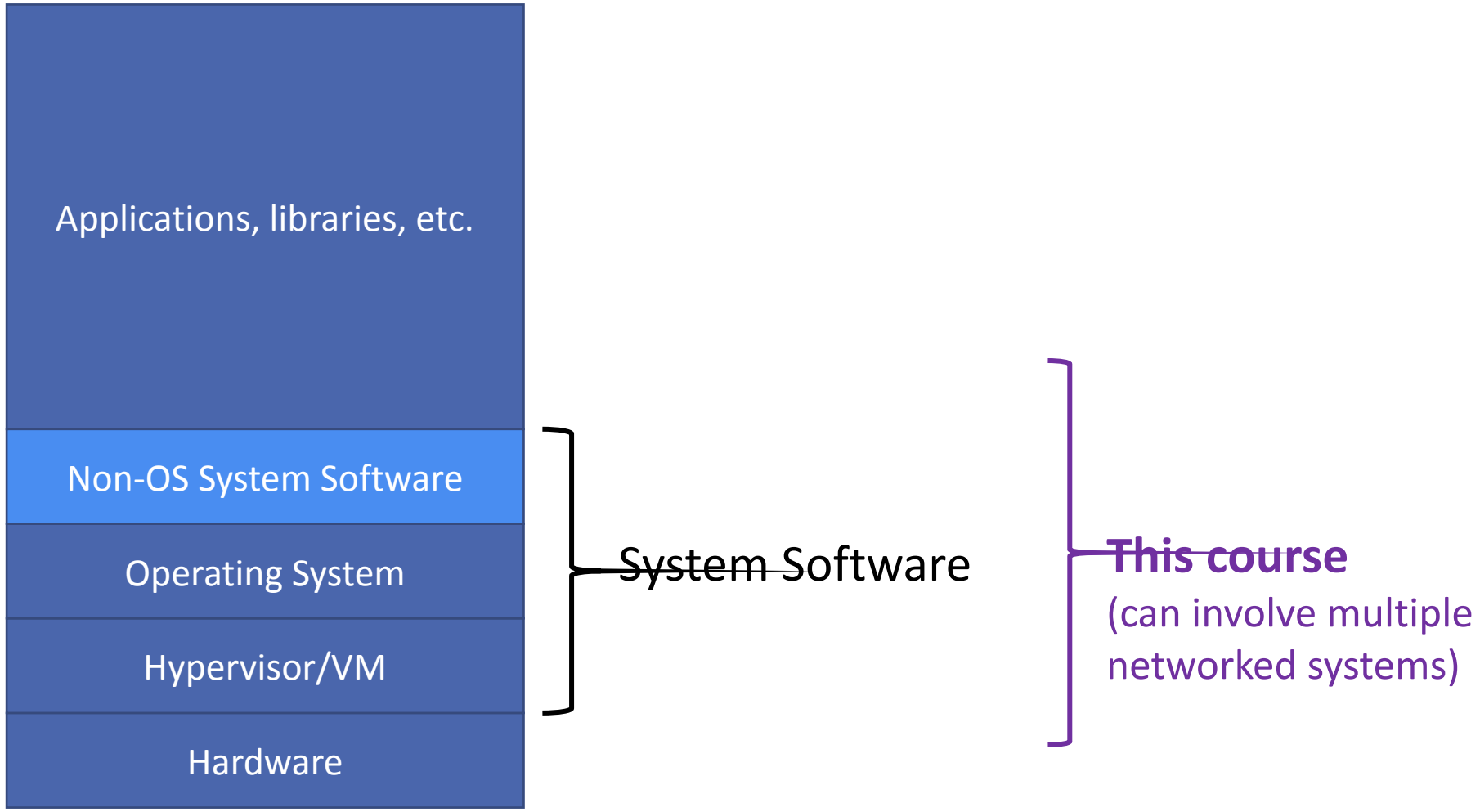
-- Wikipedia

# System Software

| System Software | Application Software |
| --- | --- |
| • System software are mainly designed for managing system resources. | • Application software are designed to accomplish tasks for specific purposes. |
| • Programming of system software is complex. | • Programming of application software is comparatively easy. |
| • A computer cannot run without system software. | • A computer can easily run without application software. |
| • System software do not depend on application software. | • Application software depend on system software and cannot run without system software. |

https://www.tutorialsmate.com/2020/12/difference-between-system-software-and-application-software.html

# System Software



# Software System Security Areas

- Discover bugs
- Discover vulnerabilities
  - Exploitable bugs
- Discover new exploitation techniques
  - Turn more bugs into vulnerabilities
- Harden exploitation
  - Exploiting a vulnerability becomes harder
- Eliminate vulnerability class
  - Eliminate certain vulnerabilities all-together (too ambitious?)

~~**This course**~~
(can involve multiple networked systems)

# When Do Bugs Become Vulnerabilities

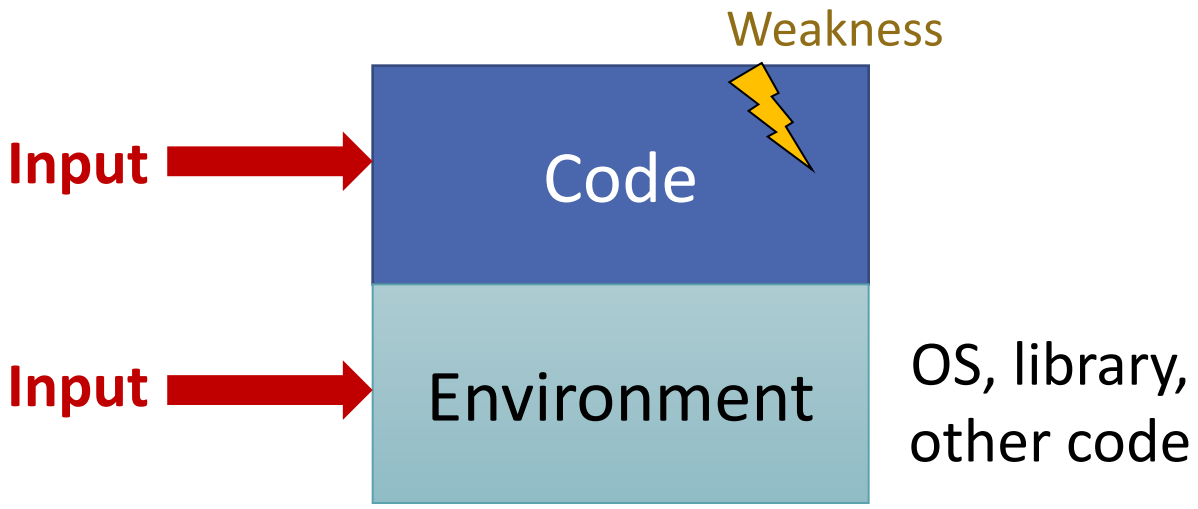
- Definitions
  - "…vulnerability is a weakness which can be exploited by a threat actor, such as an attacker, to cross privilege boundaries"
  - "To exploit a vulnerability, an attacker must have at least one applicable tool or technique that can connect to a system weakness."

# When Do Bugs Become Vulnerabilities

- Definitions
  - "…vulnerability is a weakness which can be exploited by a threat actor, such as an attacker, to cross privilege boundaries"
  - **"To exploit a vulnerability, an attacker must have at least one applicable tool or technique that can connect to a system weakness."**

# When Do Bugs Become Vulnerabilities

- Definitions
  - "…vulnerability is a weakness which can be exploited by a threat actor, such as an attacker, to cross privilege boundaries"
  - **"To exploit a vulnerability, an attacker must have at least one applicable tool or technique that can connect to a system weakness."**



# When Do Bugs Become Vulnerabilities

- Definitions
  - **"…vulnerability is a weakness which can be exploited by a threat actor, such as an attacker, to cross privilege boundaries"**
  - "To exploit a vulnerability, an attacker must have at least one applicable tool or technique that can connect to a system weakness."
- Privilege boundaries ensure what can be accessed …
  - Data
  - Resources

# When Do Bugs Become Vulnerabilities

- Definitions
  - **"…vulnerability is a weakness which can be exploited by a threat actor, such as an attacker, to cross privilege boundaries"**
  - "To exploit a vulnerability, an attacker must have at least one applicable tool or technique that can connect to a system weakness."
- Privilege boundaries ensure what can be accessed …
  - Data
  - Resources
- Attacks violate privilege boundaries:
  - Data  Read (**confidentiality**) or write (**integrity**)
  - Resources  Execute instructions on the CPU, actuate a machine (cyber-physical systems), etc. (**integrity**) or deprive resources from the system (**availability**)

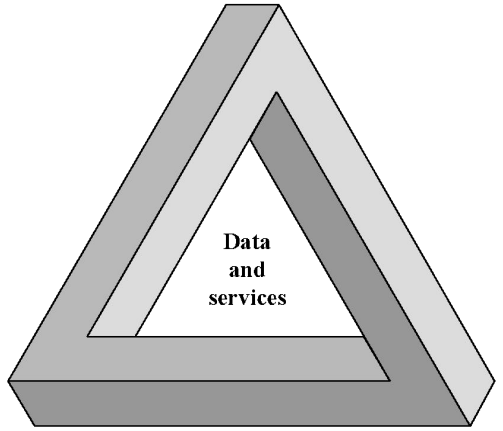



Figure 1.1 The Security Requirements Triad

# When Do Bugs Become Vulnerabilities

- Definitions
  - **"…vulnerability is a weakness which can be exploited by a threat actor, such as an attacker, to cross privilege boundaries"**
  - "To exploit a vulnerability, an attacker must have at least one applicable tool or technique that can connect to a system weakness."
- Privilege boundaries ensure what can be accessed …
  - Data
  - Resources
- Attacks violate privilege boundaries:
  - Data  Read (**confidentiality**) or write (**integrity**)
  - Resources  Execute instructions on the CPU, actuate a machine (cyber-physical systems), etc. (**integrity**) or deprive resources from the system (**availability**)
- **An attack must gain something for the attacker**

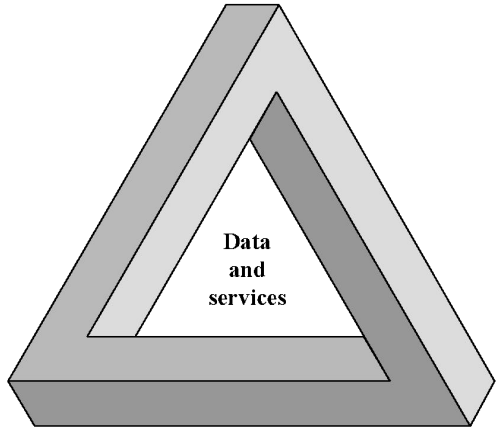



Figure 1.1 The Security Requirements Triad

# TOP 25 Most Dangerous Software Errors
## (CWE/SANS)

- The targeted system and vulnerability class generally determines what is possible

| Porous Defenses | Risky Resource Management | Insecure Interaction Between Components |
|---|---|---|
| Missing Authentication for Critical Function | Buffer Copy without Checking Size of Input ('Classic Buffer Overflow') | Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') |
| Missing Authorization | Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') | Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection') |
| Use of Hard-coded Credentials | Download of Code Without Integrity Check | Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') |
| Missing Encryption of Sensitive Data | Inclusion of Functionality from Untrusted Control Sphere | Unrestricted Upload of File with Dangerous Type |
| Reliance on Untrusted Inputs in a Security Decision | Use of Potentially Dangerous Function | Cross-Site Request Forgery (CSRF) |
| Execution with Unnecessary Privileges | Incorrect Calculation of Buffer Size | URL Redirection to Untrusted Site ('Open Redirect') |
| Incorrect Authorization | Uncontrolled Format String | |
| Incorrect Permission Assignment for Critical Resource | Integer Overflow or Wraparound | |
| Use of a Broken or Risky Cryptographic Algorithm | | |
| Improper Restriction of Excessive Authentication Attempts | | |
| Use of a One-Way Hash without a Salt | | |

Known weaknesses
https://cwe.mitre.org/

Known vulnerabilities
https://cve.mitre.org/

# Generic Attack Types

- Local attacks

- Remote attack

# Types of Attack: Remote

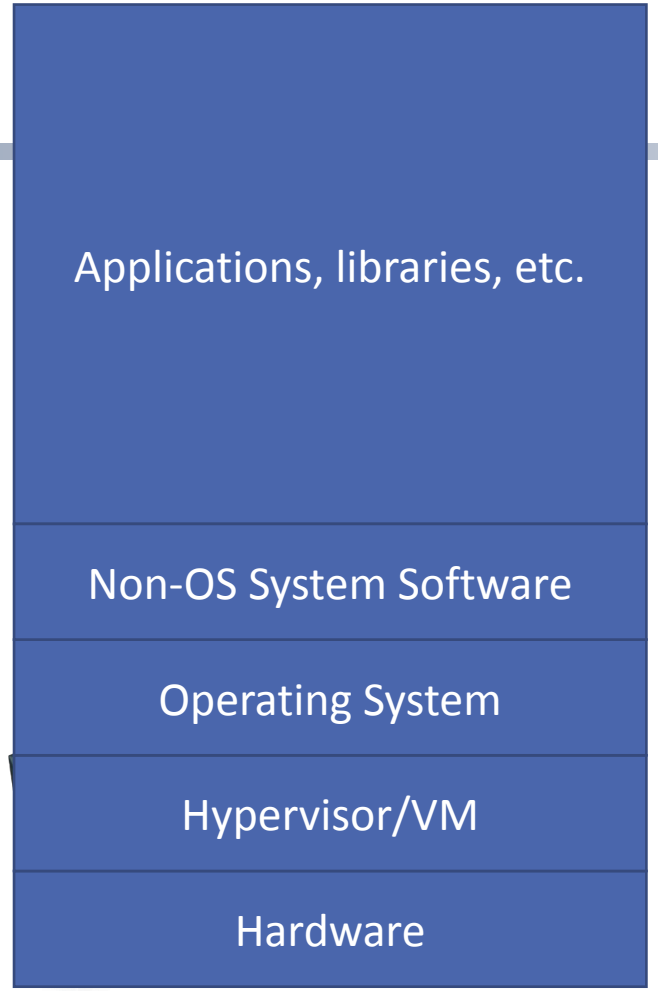
# Types of Attack: Remote

The target is important

Attacker

Malicious Input

?




Vulnerable Target

# Types of Attack: Remote

Attacker

The target is important

Malicious Input

?

Applications, libraries, etc.

Non-OS System Software

Operating System

Hypervisor/VM

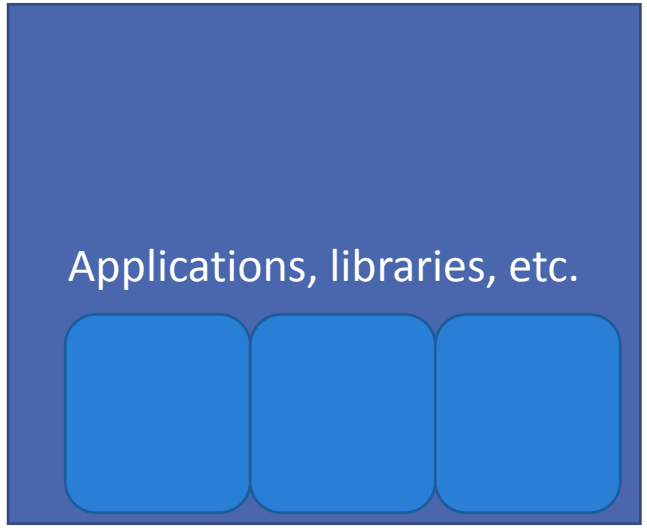
Hardware

Vulnerable Target

More privileged

# Types of Attack: Remote

The application layer
also has multiple components




Malicious Input

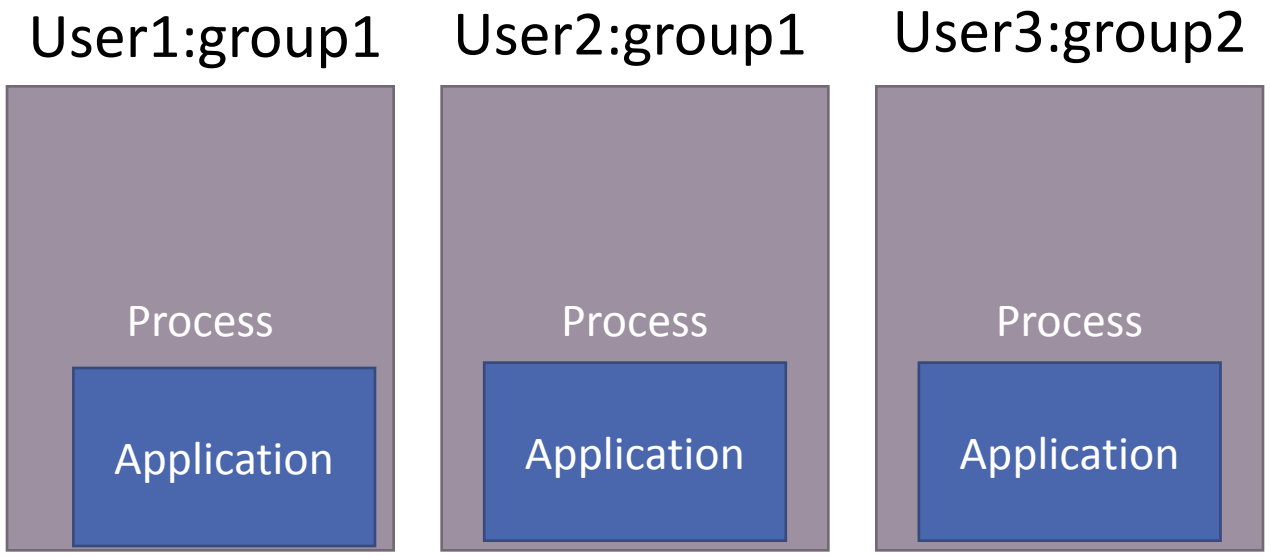
?

Attacker

Vulnerable
Target

# Processes and Permissions

- Example shows UNIX-style permissions

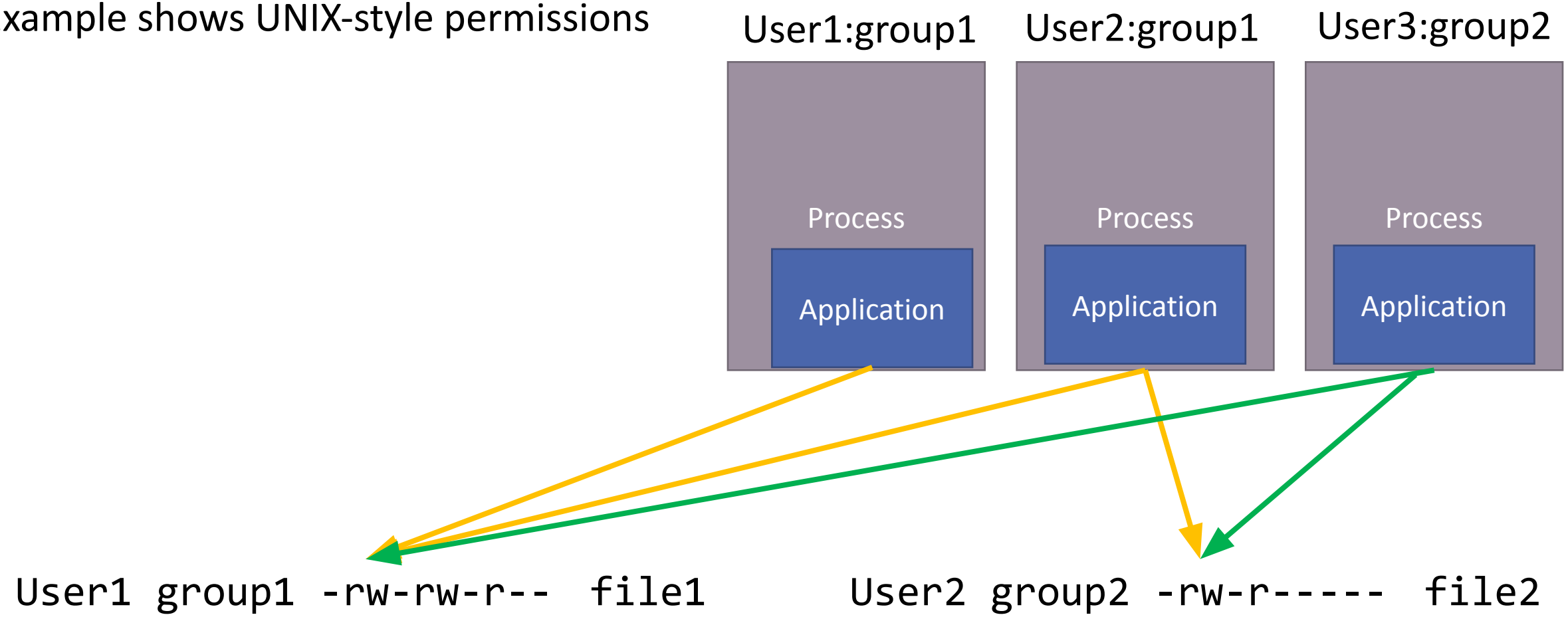



```
User1 group1 -rw-rw-r--  file1
```

```
User2 group2 -rw-r-----  file2
```

# Processes and Permissions

- Example shows UNIX-style permissions



# Processes and Permissions

- The attacker is still limited by system permissions

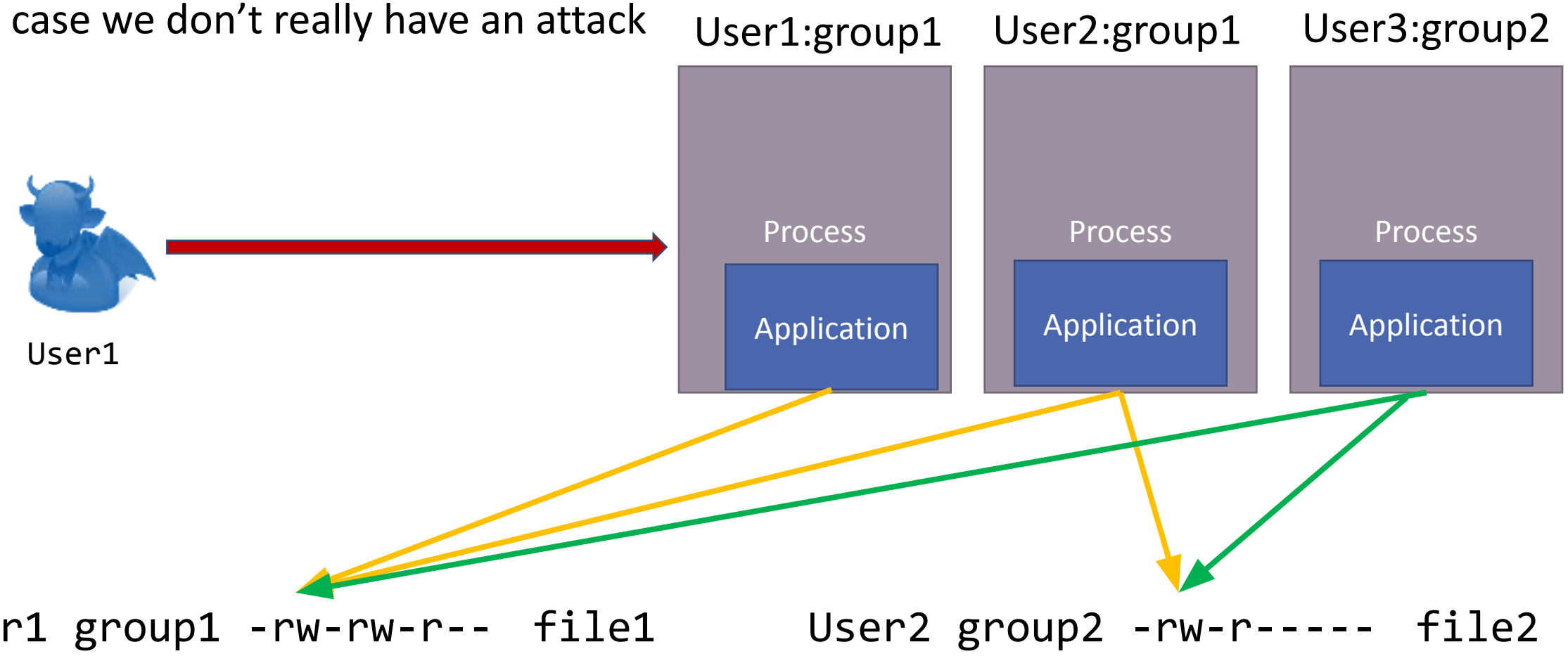
User1:group1 User2:group1 User3:group2

Process Application | Process Application | Process Application

```
User1 group1 -rw-rw-r--  file1        User2 group2 -rw-r-----  file2
```

# Processes and Permissions

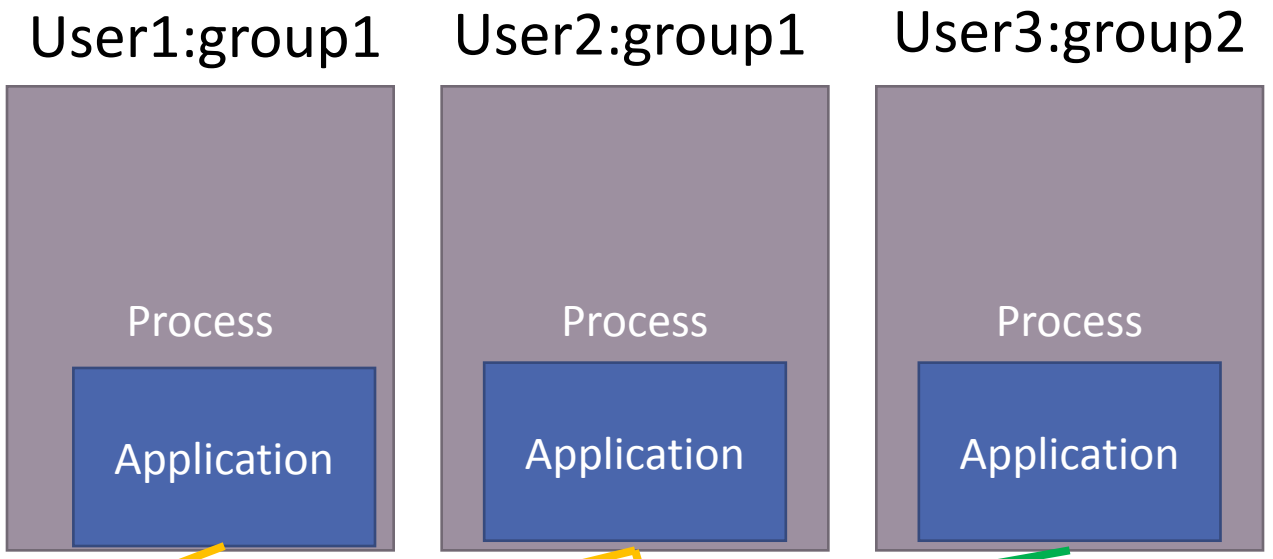
- In this case we don't really have an attack



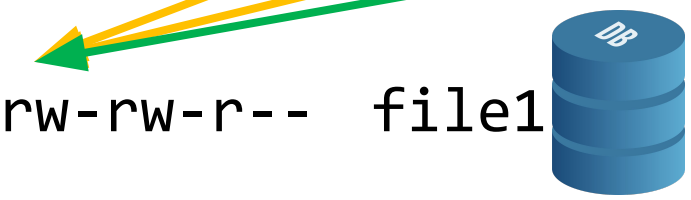
# Processes and Permissions

- Frequently applications implement its own access control system

UserN




- **Compromising such applications can expose all user data**



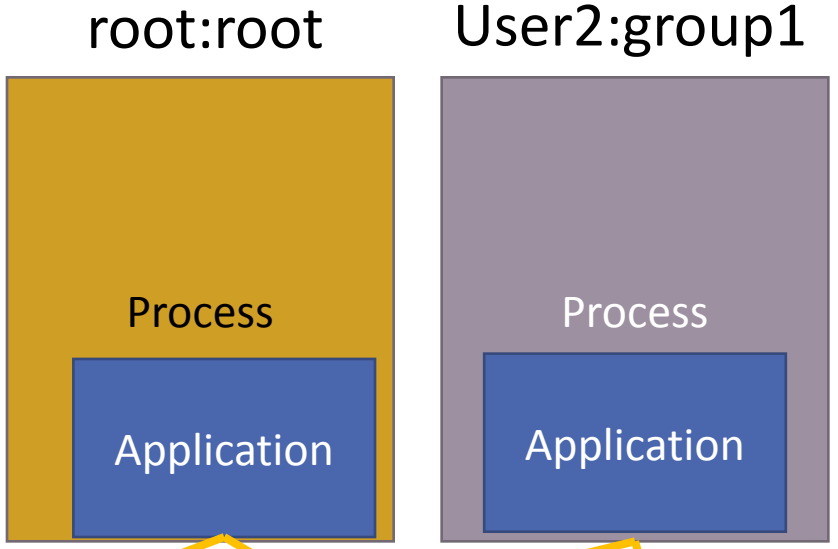

UserA
UserB
...

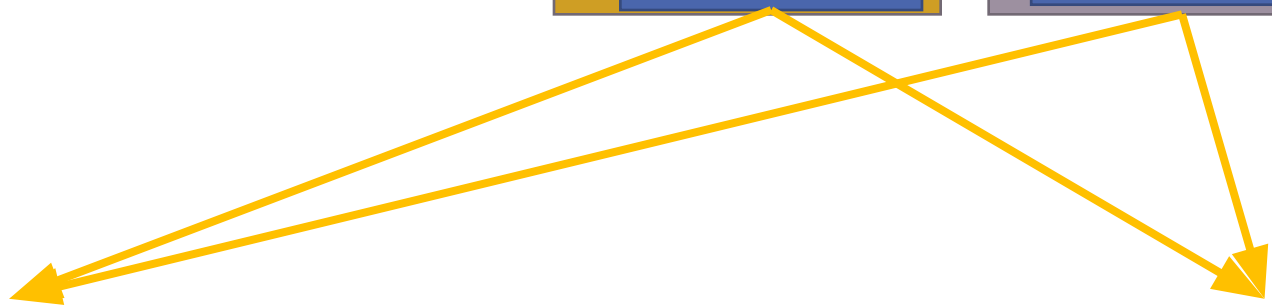
User1 group1 -rw-rw-r-- file1

User2 group2 -rw-r----- file2

# Processes and Permissions

- Applications running with administrator privileges are the worst/best

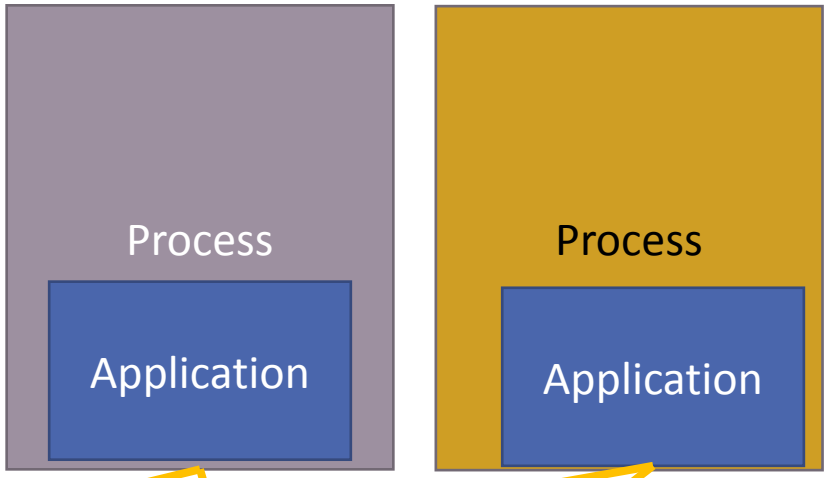


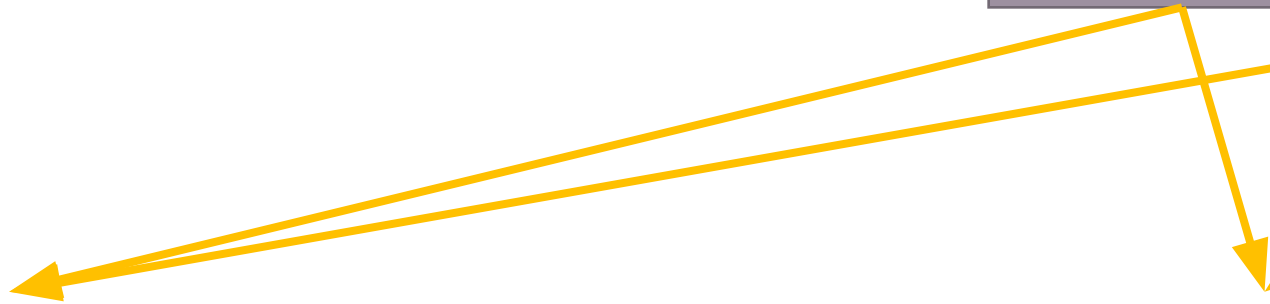



```
User1 group1 -rw-rw-r--  file1        User2 group2 -rw-r-----  file2
```

# Processes and Permissions

- Applications designed carefully avoid running as administrator or only run part of their code as one

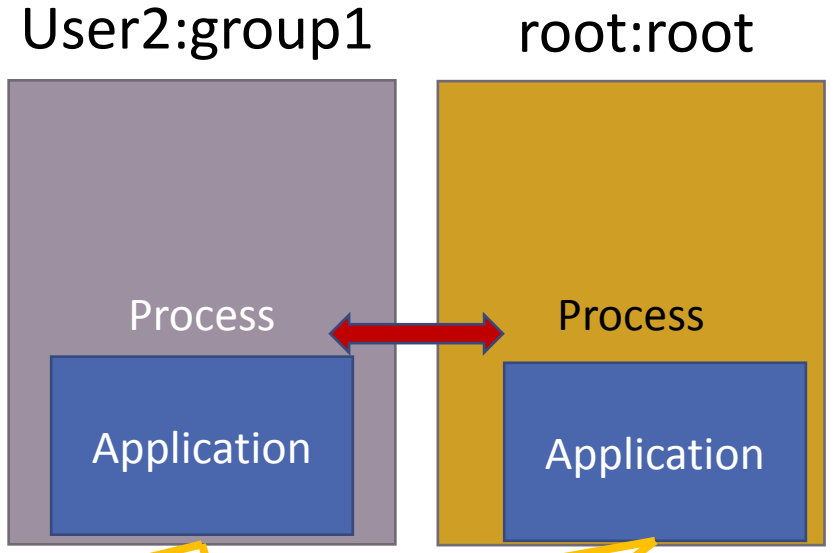


```
User1 group1 -rw-rw-r--  file1        User2 group2 -rw-r-----  file2
```

# Processes and Permissions

- Attackers aim to “jump” from least to most privileged domain/process
  - Example: by exploiting another vulnerability




- This is called **privilege escalation**

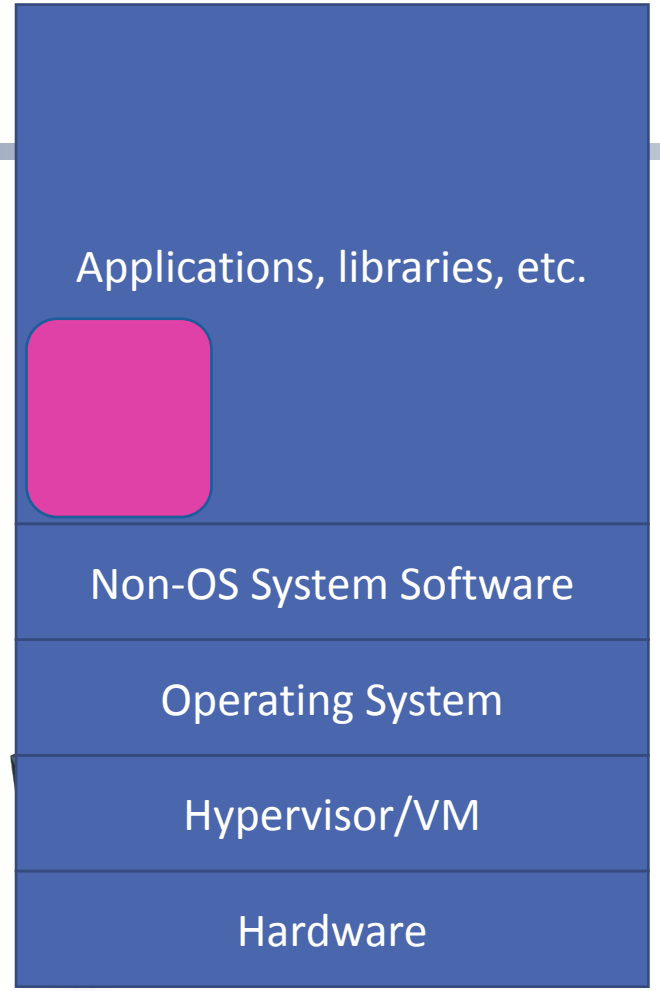
```
User1 group1 -rw-rw-r--  file1        User2 group2 -rw-r-----  file2
```

# Typical Privilege Escalation

Attacker

1. Compromise software running in user process

**Malicious Input**




Vulnerable Target

More privileged

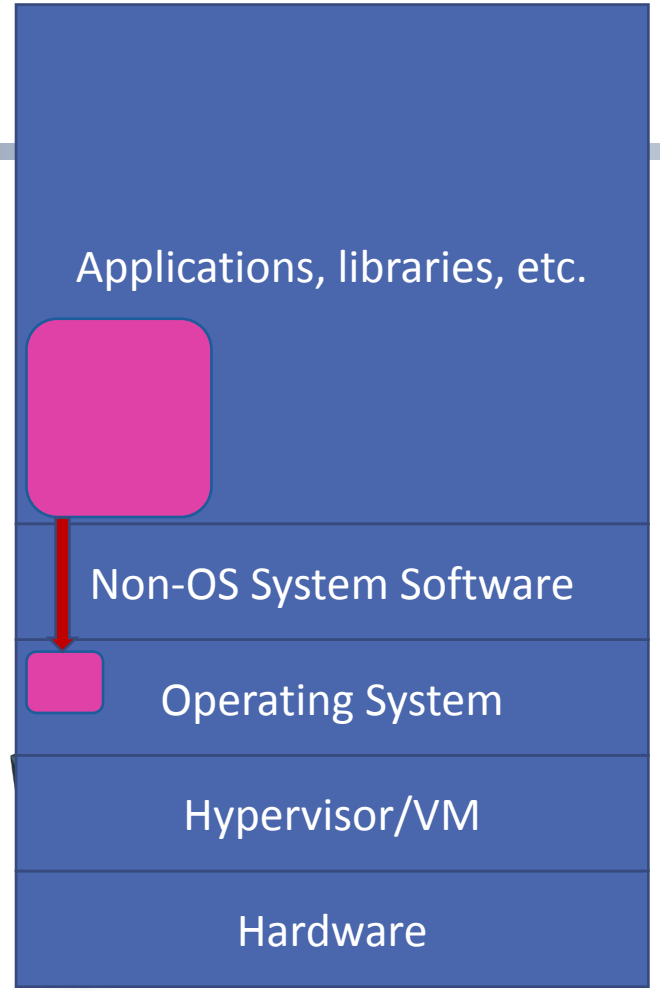
# Typical Privilege Escalation

Attacker

1. Compromise software running in user process

Malicious Input

2. Compromise operating system

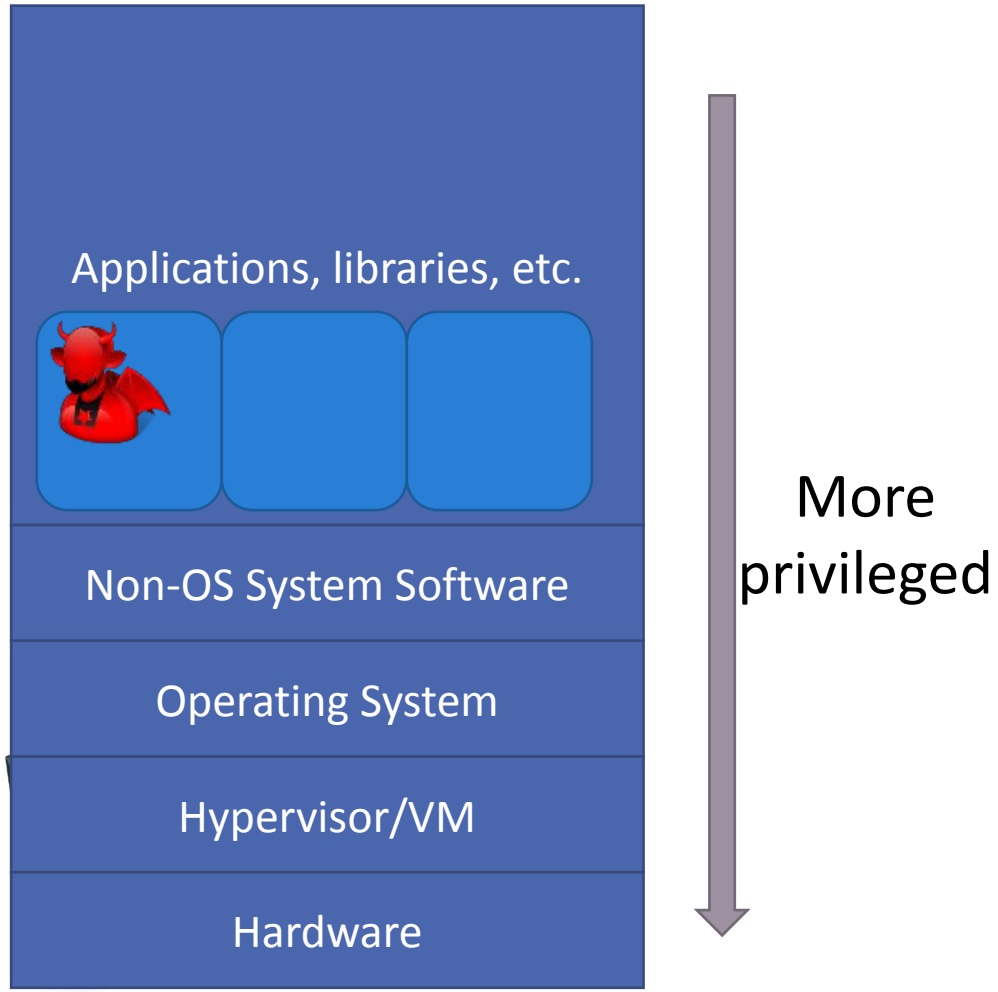



Vulnerable Target

More privileged

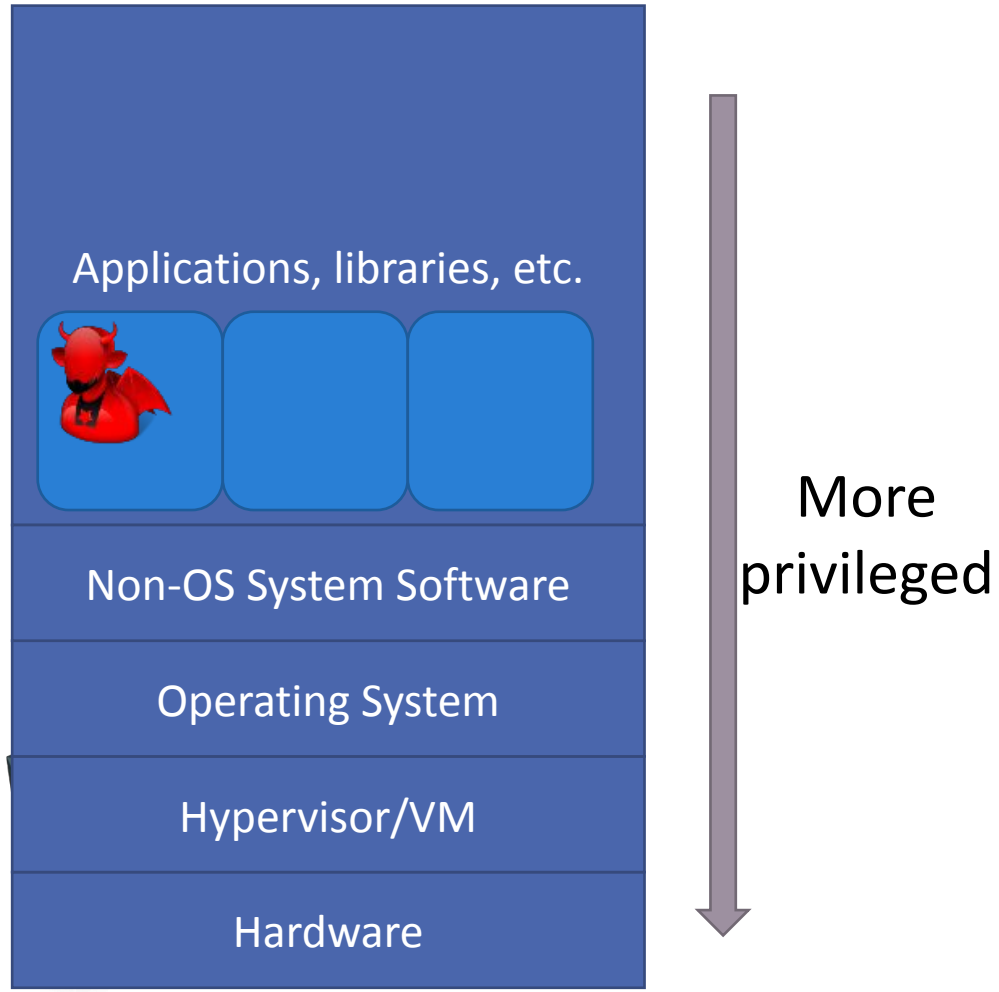
# Types of Attack: Local

- The attacker already has access to the system but is limited by the access control in place
  - Example: non-admin user in a UNIX-based system



# Types of Attack: Local

- The attacker already has access to the system but is limited by the access control in place
  - Example: non-admin user in a UNIX-based system
- Goal: elevate privileges
  - Exploit a vulnerability in software running in process with privileges
  - Exploit a vulnerability in the layers below
  - Special case: SUID binaries



# (SUID) Set User ID on Execution

- Example: Enable a user to change their own password

user:user

$ passwd

File contains user meta data (name, shell, etc.)

-rw-r--r-- 1 root root …
/etc/passwd

-rw-r----- 1 root shadow …
/etc/shadow

File contains hashed password

-rwxr-xr-x 1 root root …
/usr/bin/passwd

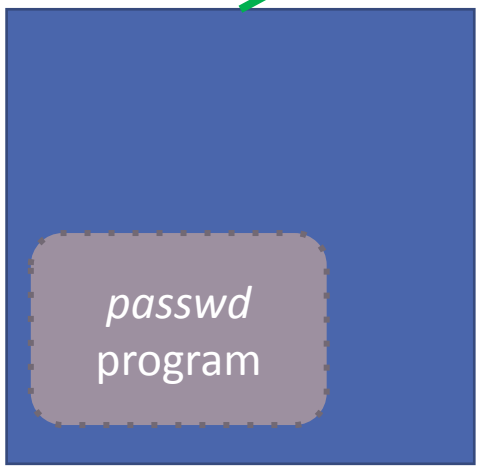
passwd program for changing password

# (SUID) Set User ID on Execution

- Example: Enable a user to change their own password

user:user

$ passwd




Process running as user:user

File contains user meta data (name, shell, etc.)

-rw-r--r-- 1 root root …
/etc/passwd

-rw-r----- 1 root shadow …
/etc/shadow

File contains hashed password

-rwxr-xr-x 1 root root …
/usr/bin/passwd

passwd program for changing password

# (SUID) Set User ID on Execution

- Example: Enable a user to change their own password
- **SUID programs run as the owner**

user:user

$ passwd


Process running as **root:**user

File contains user meta data (name, shell, etc.)

-rw-r--r-- 1 root root …
/etc/passwd

-rw-r----- 1 root shadow …
/etc/shadow

File contains hashed password

-rwx**s**-xr-x 1 root root …
/usr/bin/passwd

passwd program for changing password

# (SUID) Set User ID on Execution

- Example: Enable a user to change their own password
- **SUID programs run as the owner**
- Useful for performing actions that would otherwise require super-user privileges
- The program is **trusted** to perform only the actions advertised
- Also, SGID programs run as the group

- You can find SUID programs in your system using `find`

*$ sudo find /usr/bin -perm -u=s*

# (SUID) Set User ID on Execution

- Example: Enable a user to change their own password
- **SUID programs run as the owner**
- Useful for performing actions that would otherwise require super-user privileges
- The program is **trusted** to perform only the actions advertised
- Also, SGID programs run as the group

- You can find SUID programs in your system using `find`

*$ sudo find /usr/bin -perm -u=s*

- **How can these programs be misused by attackers?**