

# Computer and Network Security

## Lecture 14: OS Security & Isolation

COMP-5370/6370  
Fall 2024

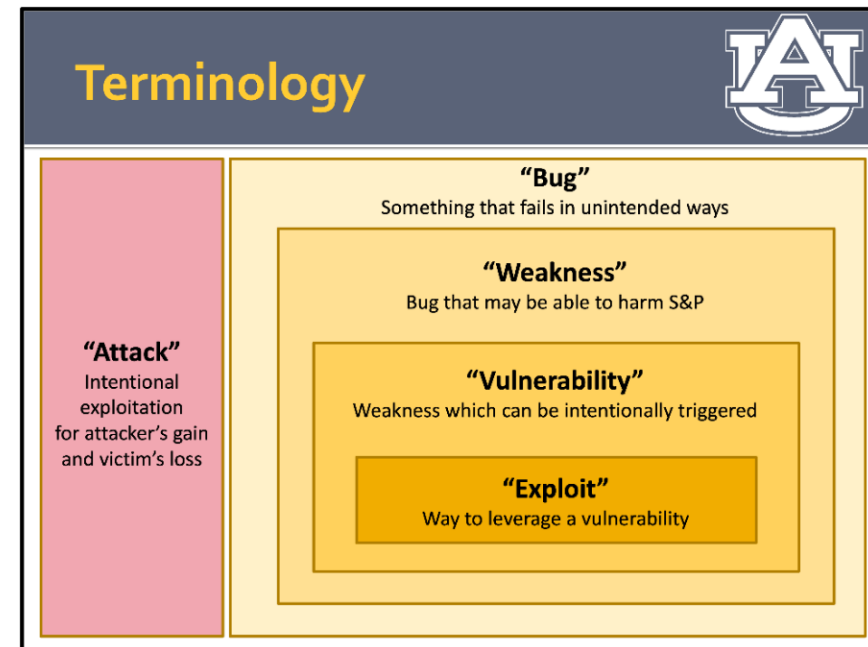




# Software Facts of Life



- Software has bugs
- Some bugs are weaknesses
- Some weaknesses are vulnerabilities
- Some vulnerabilities can be exploited
- Someone has an interest in exploiting others for gain



# Do you trust other people's code?



- Linux kernel: 27.8M lines of C
  - Pointer math, raw byte buffers, etc.
  - C99 compliant code lacks features like “smart pointers”, threads, implicit bounds-checking,...
  - Written, reviewed, and tested by those strange and mythical people known as “kernel devs”

# Do you trust other people's code?



- Do you use an HP computer?
  - Android OS or Google's Android Apps?
  - Google services (GMail, Docs, Calendar, etc)?



Finster Professor



Shady Professor

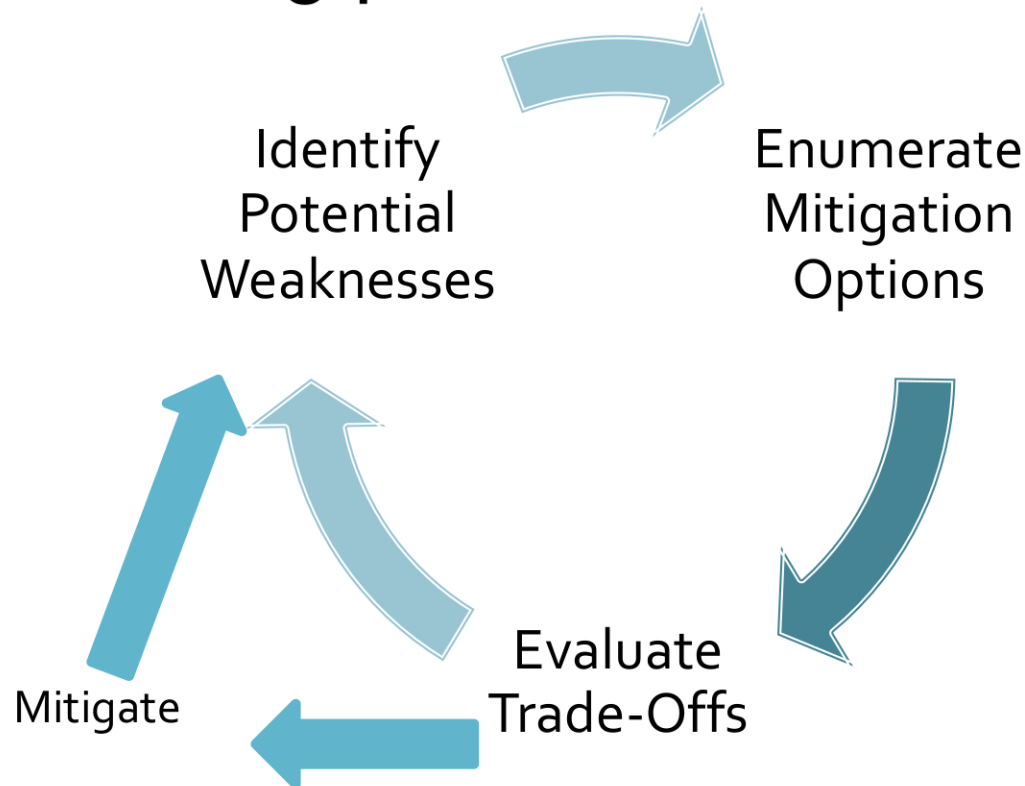


Profesor Turbio

# Threat Modeling



A systematic approach to analyzing and understanding potential weaknesses.



# During Planning



- Create a system that is defensible



# Principle of Least Privilege



The **Principle of Least Privilege** is that access to data & resources is limited those who need for routine, authorized purposes.

- “Specific users can only run specific apps”
- “Normal users aren’t root-users”
- “routine” means **ROUTINE**
  - Having everyday access for something that happens once every 3 years is not recommended



# Principle of Complete Mediation



The **Principle of Complete Mediation** is having a trusted entity validate any privilege use to ensure its validity.

- OS validates user X can run app Y
- OS validates that app Y is allowed to use permission Z

# Applying Principles



- **Security Model** – An abstraction to *subjects*, *permissions*, and *objects* to allow reasoning about S&P properties.
- **Security Policy** – The mapping of *subjects*, *permissions*, and *objects* to implement the security model.
- **Security Mechanism** – The technical measure that enforces the security policy.

# Access Control Model



## Formal Models

- Discretionary Access Control (DAC)
- Mandatory Access Control (MAC)
- Role-based Access Control (RBAC)
- Bell-LaPadula
- <many, many more>

**The real-world is a mixture of all.**

## Filesystem Example

### **Subjects**

Users

Groups

### **Permissions**

Read

Write

Execute

### **Objects**

“Files” (data, binary, device)  
Hierarchies (multiple files)

# Access Control Policy

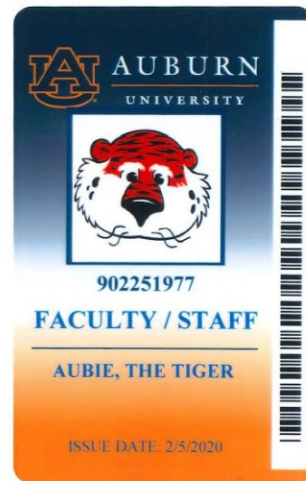
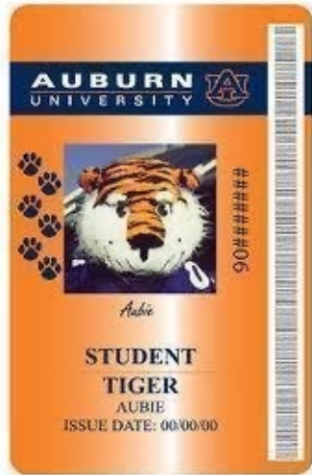


Create a logical representation of who has access to what file.

- Enumerate users
- Enumerate files
- Give specific users specific access to specific files
- Default: no-access

		Objects			
		File 1	File 2	File 3	File 4
Subjects	Alice	read	read/write	no access	no access
	Bob	read	read/write	no access	no access
	Carol	read	write	read/write	read/write
	Wendy	read/write	read/write	read	read

# During Operation



- Create a system that is defensible
- Maintain the mechanisms for defending it

# Subject Tracking Mechanism



Implement the policy in the real-world.

- Users assigned **user id**
- Users can **masquerade** as other users
- Users assigned to groups for simplicity (logic/mgmt)
- Groups are **meta-users** in some but not all aspects

```
user@desktop:~$ whoami; id -u
user
1000
```

```
user@desktop:~$ sudo whoami
[sudo] password for user:
root
```

```
user@desktop:~$ groups
user adm faculty
```

```
user@desktop:~$ $ sudo -u adm bash -c 'hi'
Sorry, user adm is not allowed to execute '/usr/bin/bash'
user@desktop:~$ $ sudo -u root bash -c 'hi'
hi
```

# User Permission Tracking



- 3 permission bits per object (RWX)

<code>rwX</code>	<code>rwX</code>	<code>rwX</code>
Owner	Group	Others

```
user@desktop:~$ ls -l
d rwX rwX --- 1 user user      4096 Apr  2 15:56 go
- rw- rw- r-- 1 user user          0 Apr 11 04:15 test.py
d rwX rwX --- 1 user faculty 4096 Dec 28 21:09 courses
```

Permissions	Owner	Group	Object
-------------	-------	-------	--------

- Each object has an “owner” and a group
- Only owner can change the permissions or group

```
user@desktop:~$ ls -l
- rw- r-- --- 1 user faculty 302 Apr 11 04:15 main.py
user@desktop:~$ chmod 751 main.py
user@desktop:~$ chgrp adm main.py
user@desktop:~$ ls -l
- rwX r-x r-x 1 user adm      302 Apr 11 04:15 main.py
```

# OS Mediation of File Access



- Who is trying to act?

```
user@desktop:~$ whoami; id -u
user
1000
```

- What are they trying to act on?

```
user@desktop:~$ ls -l
d rwx rwx --- 1 user user 4096 Apr  2 15:56 go
- rw- rw- r-- 1 user user  0 Apr 11 04:15 test.py
d rwx rwx --- 1 user faculty 4096 Dec 28 21:09 courses
```

- What are they trying to do?

- Allow action or not?

<code>rwX</code>	<code>rwX</code>	<code>rwX</code>
Owner	Group	Others



# Processes as Subjects



Processes permissions are **nearly identical** but slightly different security mechanism.

- Process inherits user permissions (default)
  - Effective User ID (EUID)
  - Effective Group ID (GUID)
- EUID/GUID can be set manually:
  - `sudo, setuid, sg, ...`
  - Requires root user

# Do you trust other people's code?



- Do you use an HP computer?
  - Android OS or Google's Android Apps?
  - Google services (GMail, Docs, Calendar, etc)?



Finster Professor



Shady Professor

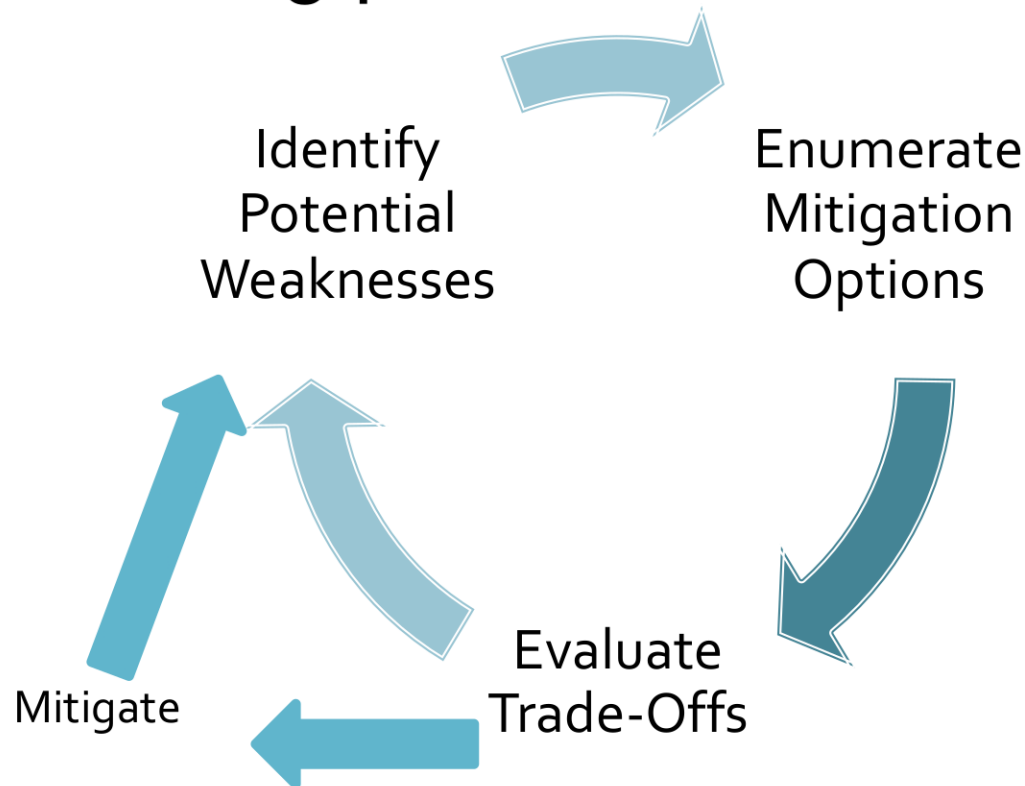


Profesor Turbio

# Threat Modeling



A systematic approach to analyzing and understanding potential weaknesses.



# Architectural Defenses



An **architectural defense** is one which is both generic in terms of implementation and focused on *isolating* a potential misbehaving application or process.

- Not tied to a specific application or attack
- ***Can defend against unknown attacks in the future due to generality***

# During Un-Wanted Events



## REPORT YOUR LOST TIGER CARD

### Students

If you cannot locate your student Tiger Card, you should immediately take one of the following actions to deactivate and protect your card. Once you have located your Tiger Card, you can easily reactivate your card the same way the card was deactivated:

1. Activate/deactivate your Tiger Card online.
  - You will need to login using your Auburn University student credentials (abc1234).
  - Once logged in, select "I Lost/Found My Card" under the "Quick Links" menu.
2. Activate/deactivate your cards through the mobile Tiger Card App.
3. Report your card as lost/found in person by coming by the Tiger Card office (Monday-Friday 7:30 a.m.- 4:30 p.m.)
4. Report your card as lost/found by phone at (334) 844-4507 (Monday-Friday 7:30 a.m.- 4:30 p.m.)

### Faculty/Staff

For information on reporting a lost **Faculty/Staff** ID, please visit [ID Card Services \(Onboarding Center\)](#) or contact them at 334-844-1763. ID Card Services (Onboarding Center) is located at [1530 East Glenn Avenue](#).

Last updated: August 13, 2021



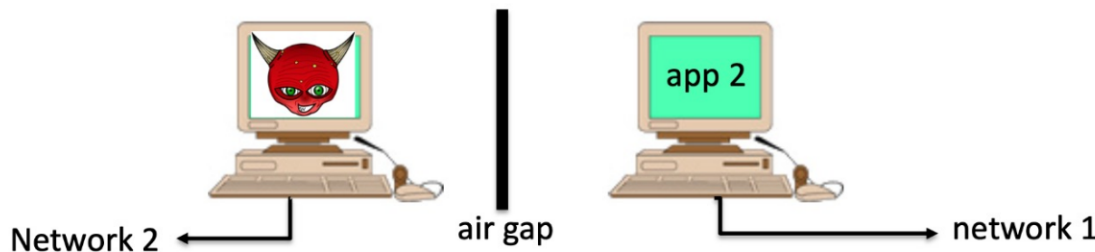
BACK TO TOP

- Create a system that is defensible
- Maintain the mechanisms for defending it
- Train and educate non-experts
- Make safety easy

# Defense: Air Gap Hardware



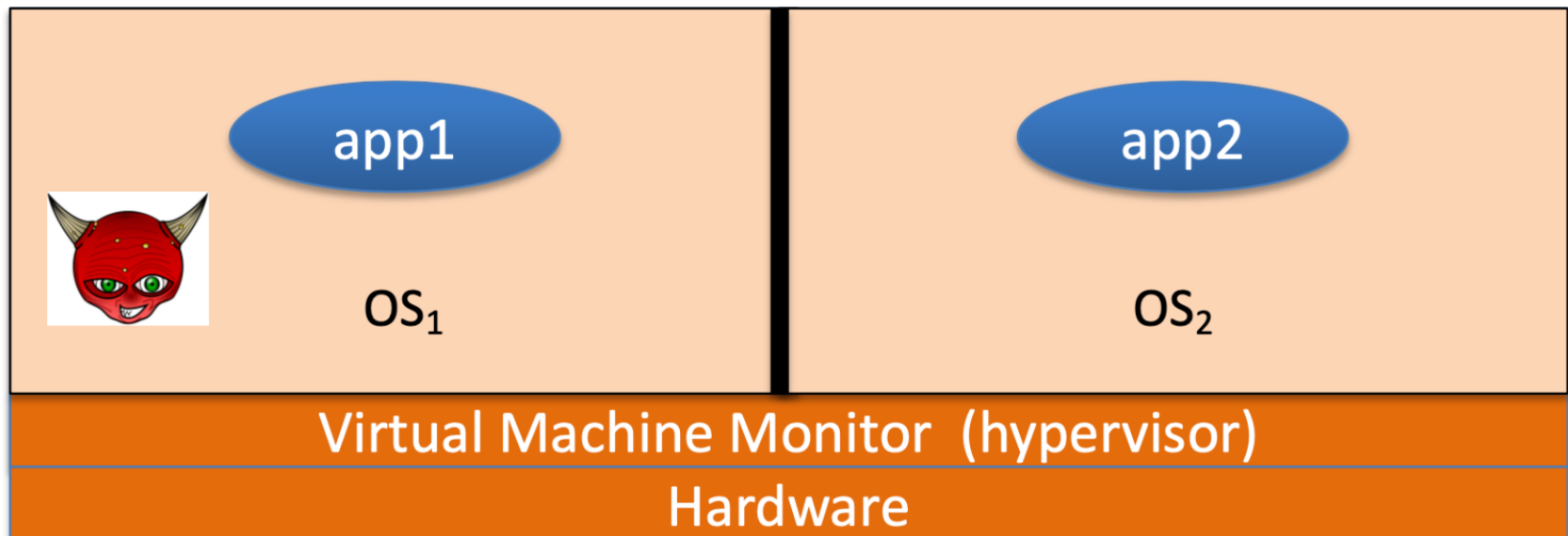
- Physically separated systems/networks
- No logical interaction across boundary
- Physical data transfer across boundary



# Defense: Virtual Machines



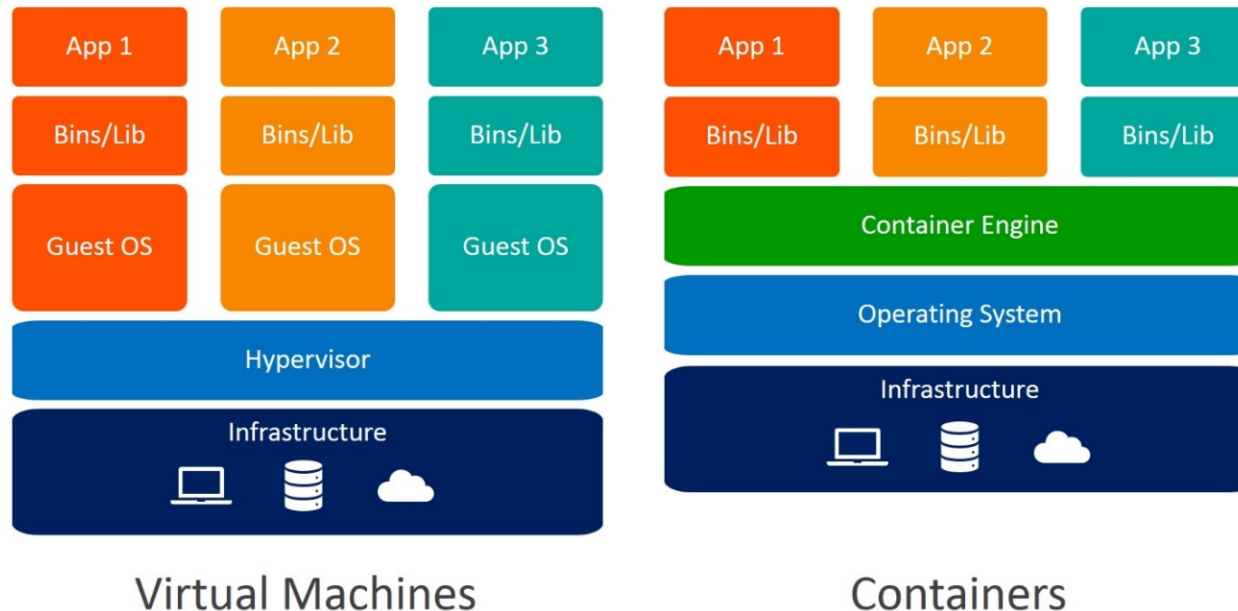
- OS-level isolation
  - Similar to dual-boot but simultaneous
- A “hypervisor” coordinates hardware access
- OSes are isolated (Linux VM on Windows)



# Defense: Containers



- Perspective-level isolation
- Shared kernel w/ isolated perspective
  - Each container thinks it's the only thing running on the entire computer

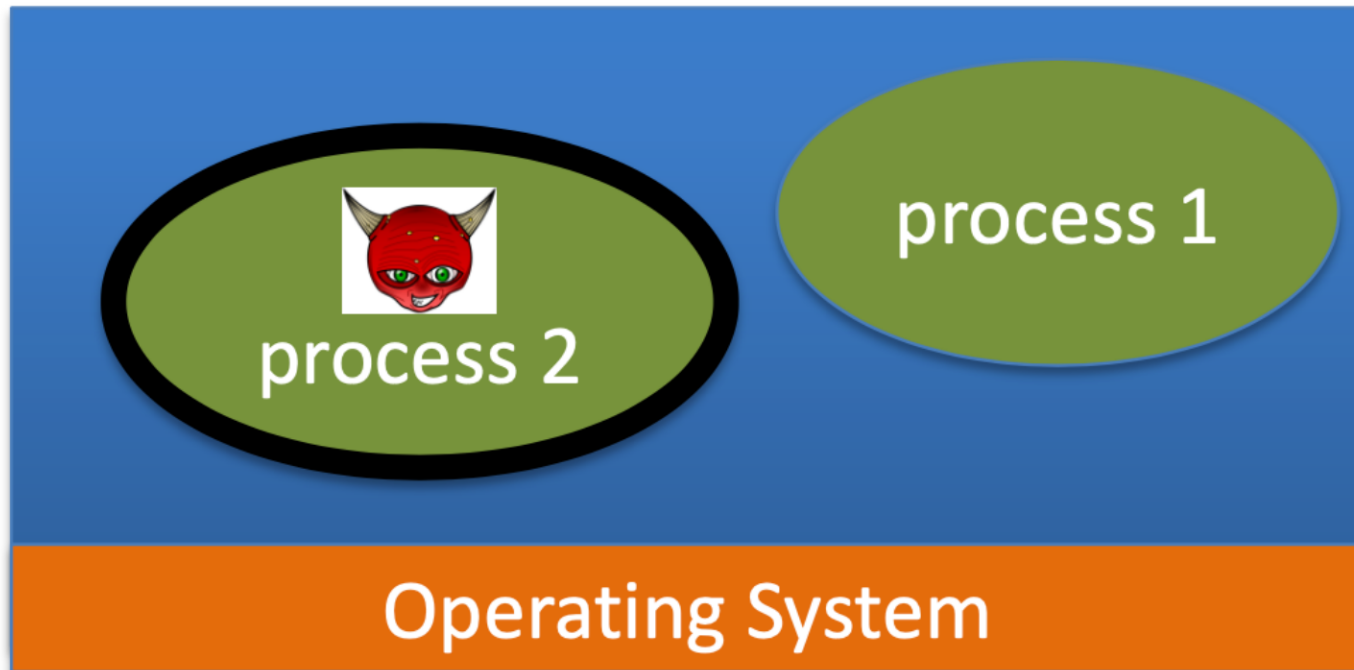




# Defense: Process Isolation



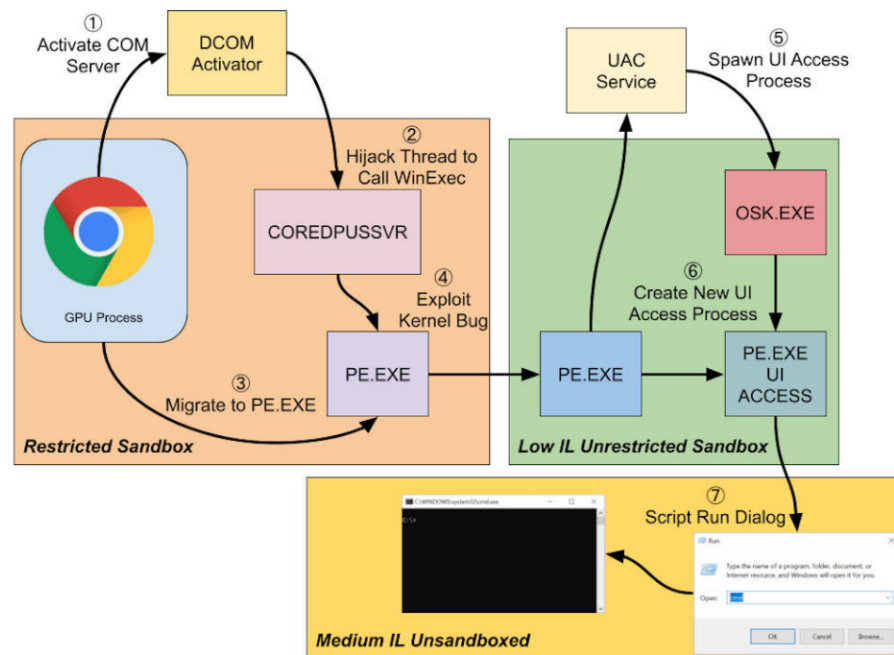
- Memory-level isolation
- Managed via OS scheduler
- Extremely efficient due to virtual memory



# Defense: Thread Sandbox



- Logic-level isolation (code logic)
- Threads interact via memory & IPCs
- Tainted threads can be killed and restarted



# Defense: Thread Sandbox

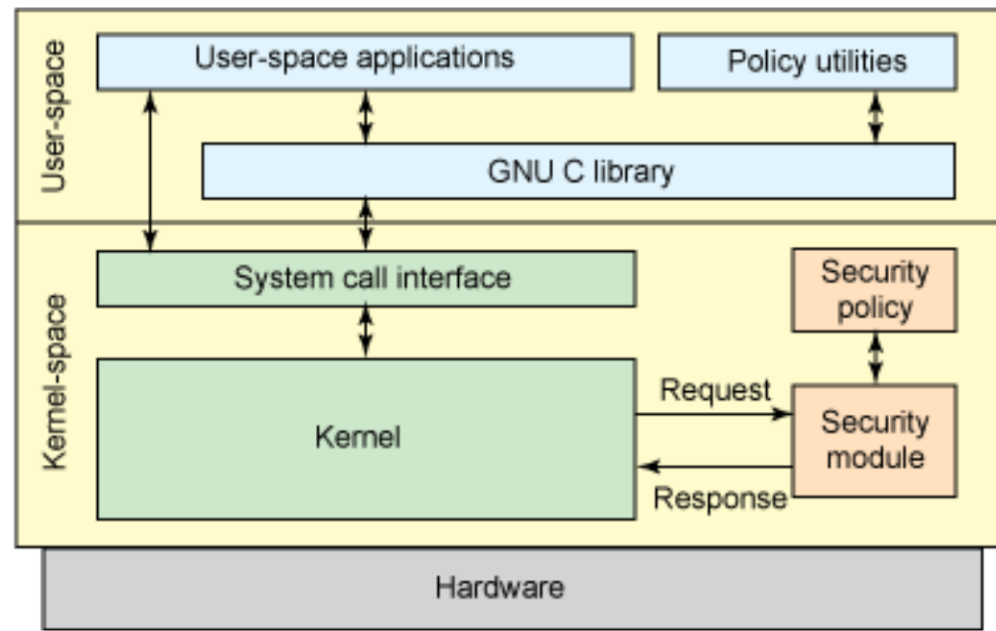


- “Do one thing and do it safely”
- A “policy engine” can blindly enforce data interactions and data exchanges
- Very useful for web browsers
  - Request content, run JS, render image, etc are *\*very\** different things with predictable inputs and outputs
- Better than nothing but is exceptionally difficult to internally isolate behavior

# Defense: SELinux/App Armor



- Permission-level isolation
- **HEAVILY** patched set of kernel modules
- “Know what an application is suppose to do and don’t let it do anything else.”
  - `ls` doesn’t need network access
  - Print driver doesn’t need keystrokes



# Defense: Use-Specific HW



- Required to install a shady VTC program?
  - Get a \$100 tower from 2012 and \$20 monitor
- Need to test a really sketchy app?
  - Get a \$30 Android phone with Wifi
- Need to install a known-spyware extension?
  - Chromebooks are \$100 on-sale



What do you think?



avast



McAfee™



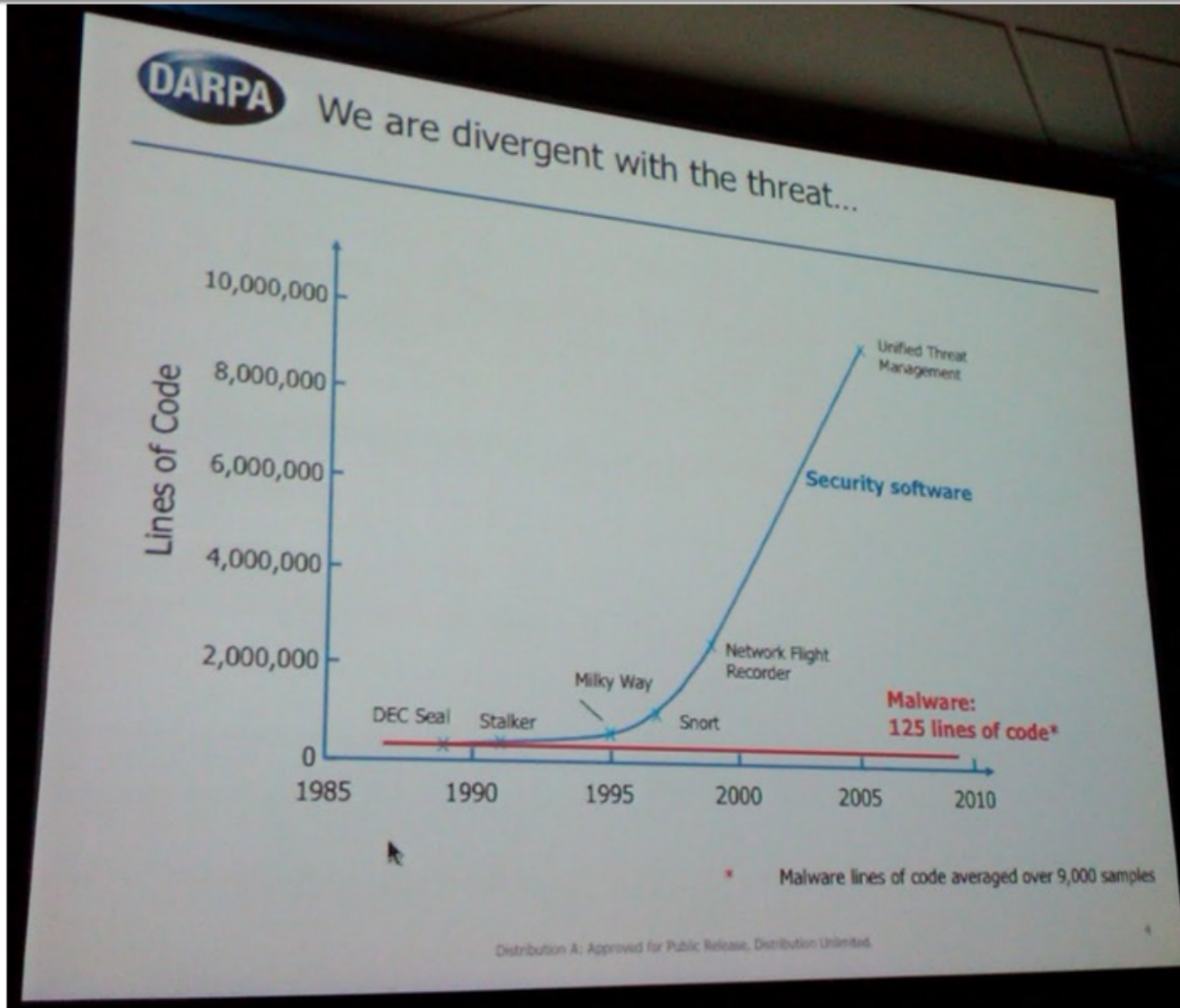
Norton™

# Defense: Anti-Virus



- Anti-Virus and “host-based defense systems” are ... complicated...
- **Good** – They can quarantine and alert you when there’s known malware.
- **Maybe** – They can only tell you about known malware and the naïve versions
- **Bad** – They love to quarantine dev-tools and known-benign security tools/apps
- **Worse** – They have to touch **everything**

# Defense: Anti-Virus

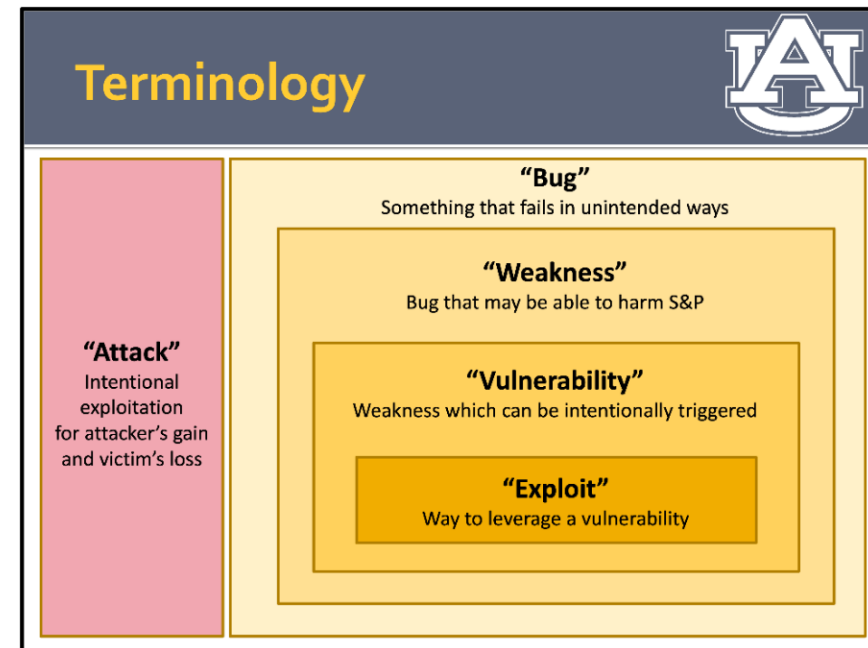




# Software Facts of Life



- Software has bugs
- Some bugs are weaknesses
- Some weaknesses are vulnerabilities
- Some vulnerabilities can be exploited
- Someone has an interest in exploiting others for gain



# Defense: Anti-Virus



I have something fun for you, I pulled the javascript interpreter out of Avast and ported it to Linux 😊

This runs unsandboxed as SYSTEM, any vulns are wormable pre-auth RCE on 400M endpoints 🤖👎👎

[github.com/taviso/avscript](https://github.com/taviso/avscript)

```
Terminal — Javascript
tavis@localhost:~/projects/avscript$ ./avscript
main(): Ready, type javascript (history available, use arrow keys)
main(): Use 'console.log()' to show output, use 'D to exit
> String.fromCharCode(0x68, 0x65, 0x6c, 0x6c, 0x6f)
hello
> "Sandboxed?".match(/^(.)(.)(.)(.)/.slice(1,3).join("")
no
> (new Date()).toString().toUpperCase()
TUE FEB 3 12:18:26 UTC+0100 2009
> Math.log({valueOf: function() { return Math.sin(1) * 2; }})
0.5205434342908536
> for (var i = 0; i < 4; i++) { console.log(i, "why does this exist"); }
0 why does this exist
1 why does this exist
2 why does this exist
3 why does this exist

> document.write("<h1>html</h1><script>console.log('parser');</script>");
parser

> try { ljkhl } catch (e) { console.log('exception'); }
exception

> ljkhl
Exception: undefined
>
```

2:59 PM · Mar 9, 2020 · Twitter Web App

1.1K Retweets 99 Quote Tweets 3.1K Likes



## BIZ & IT — This Windows Defender bug was so gaping its PoC exploit had to be encrypted

Is there a fuzzer in the house?

DAN GOODIN - 6/26/2017, 7:10 PM

### Avast Antivirus JavaScript Interpreter

The main Avast antivirus process is called AvastSvc.exe, which runs as SYSTEM.

Process Name	Private Bytes	Working Set	Process Name	Company Name	Architecture	Session Name	Process ID
svchost.exe	2,512 K	11,380 K	384 Host Process for Windows S...	Microsoft Corporation	System	NT AUTHORITY\LOCAL SERVICE	...
audiiodg.exe	6,184 K	11,972 K	1,424 Windows Audio Device Grep...	Microsoft Corporation	System	NT AUTHORITY\LOCAL SERVICE	...
svchost.exe	1,156 K	5,436 K	592 Host Process for Windows S...	Microsoft Corporation	System	NT AUTHORITY\LOCAL SERVICE	...
svchost.exe	2,196 K	8,820 K	880 Host Process for Windows S...	Microsoft Corporation	System	NT AUTHORITY\LOCAL SERVICE	...
svchost.exe	2,584 K	11,388 K	1,204 Host Process for Windows S...	Microsoft Corporation	System	NT AUTHORITY\NETWORK SERV...	...
svchost.exe	6,980 K	14,988 K	1,944 Host Process for Windows S...	Microsoft Corporation	System	NT AUTHORITY\LOCAL SERVICE	...
AvastSvc.exe	0 K	120,004 K	47,940 K	2080 Avast Antivirus Service	AVAST Software	System	NT AUTHORITY\SYSTEM
svchost.exe	4,344 K	12,872 K	2,244 Spooler SubSystem App	Microsoft Corporation	System	NT AUTHORITY\SYSTEM	...
svchost.exe	1,656 K	5,704 K	636 Host Process for Windows S...	Microsoft Corporation	System	NT AUTHORITY\LOCAL SERVICE	...

	Increased Security	Same Security	Decreased Security	Severely Broken
TLS Security				
Client Security Products	0/20	2/20	18/20	10/20
Middleboxes	0/12	1/12	6/12	5/12

# Defense: Anti-Virus

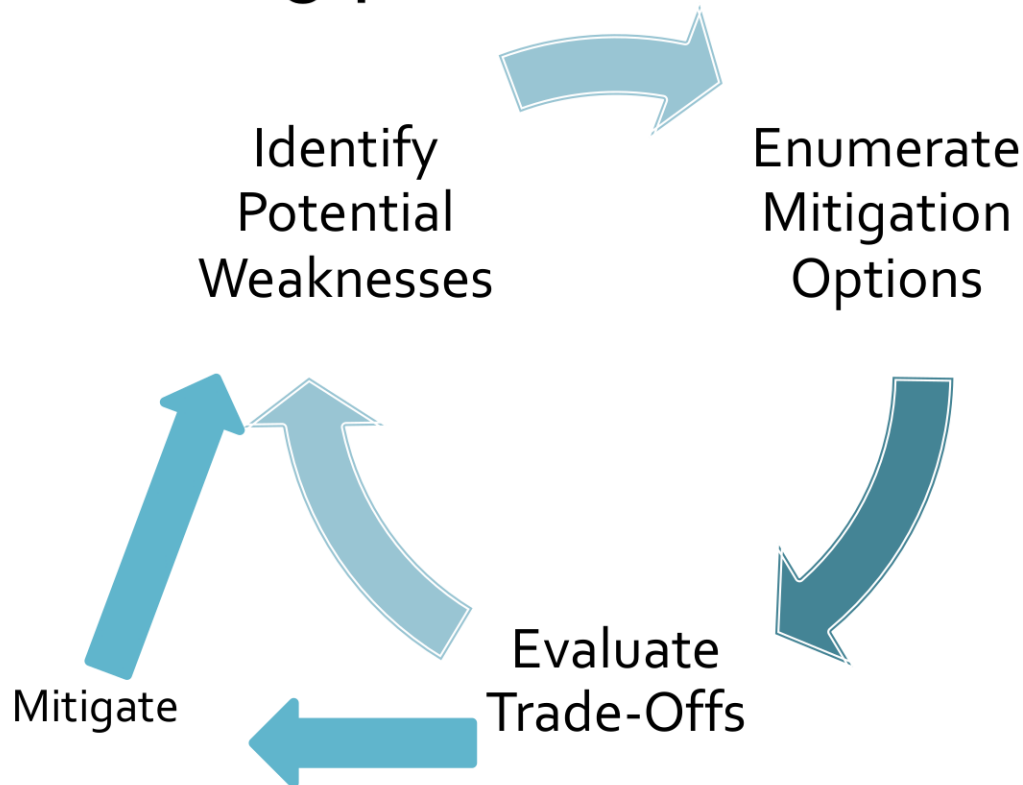


The image shows a screenshot of a Yahoo search results page for the query 'antivirus'. The browser address bar shows the URL: [https://search.yahoo.com/yhs/search?hspart=iba&hsimp=yhs-1&type=sdds\\_5343\\_CRW\\_CA&p=antivirus](https://search.yahoo.com/yhs/search?hspart=iba&hsimp=yhs-1&type=sdds_5343_CRW_CA&p=antivirus). The search results include several entries, such as 'Antivirus | Kaspersky™ Antivirus - 2019 Edition - Save 60%' and 'Top 10 Best Antivirus'. A large, semi-transparent advertisement is overlaid on the page. The advertisement is split into two vertical panels: a red panel on the left and a green panel on the right. The red panel features a white shield with a red 'M' inside, surrounded by a dotted white border and two pairs of blue scissors positioned as if cutting the shield. The green panel contains the following text in bold, red font: 'McAfee', '50% OFF', 'HURRY UP', 'EXPIRE SOON!', and 'Click Here' (underlined). The background of the advertisement is semi-transparent, allowing the search results to be partially visible through it.

# Threat Modeling



A systematic approach to analyzing and understanding potential weaknesses.



# Trusted Computing Base



The **Trusted Computing Base (TCB)** is the collection of all components within a system critical to providing security properties.

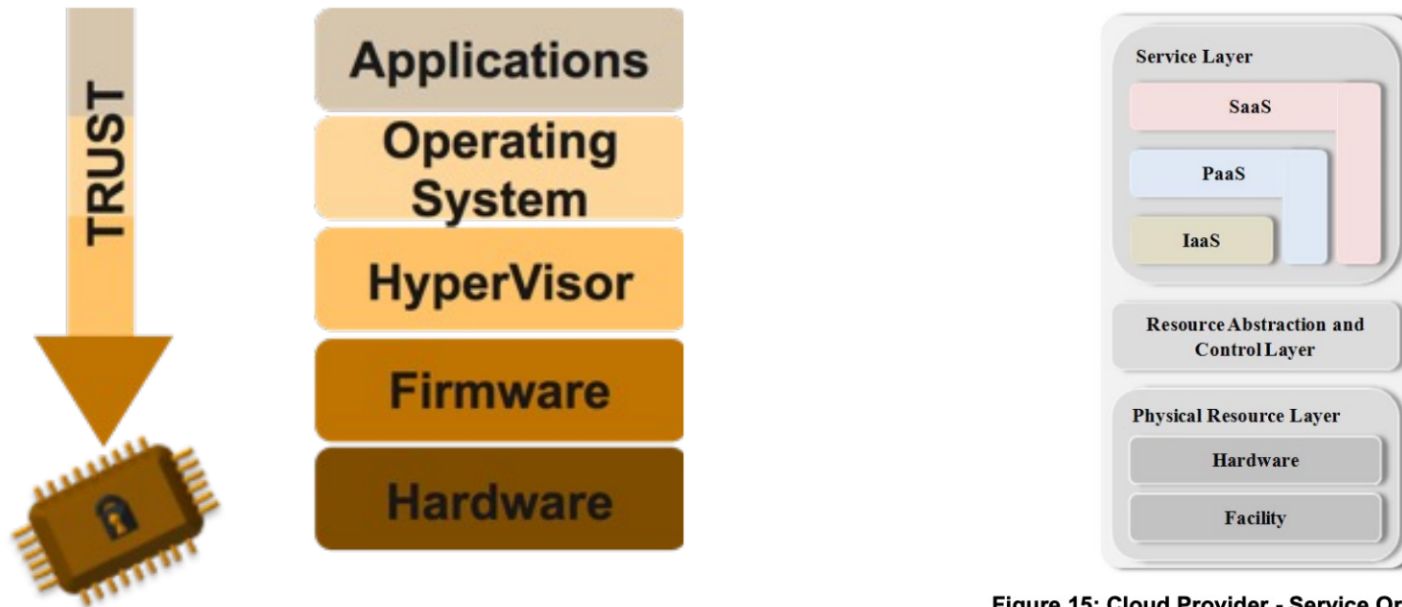
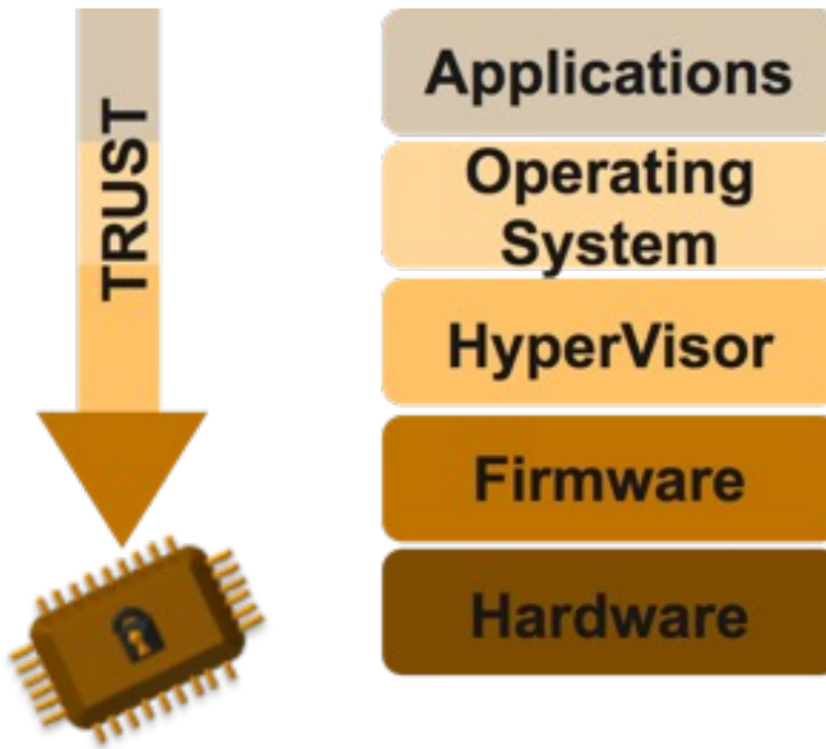


Figure 15: Cloud Provider - Service Orchestration

# Local TCB



- Everything needed to run the application safely
- Each layer relies on the layers below it to behave correctly

# Turtles All The Way Down



- Attack surface is exponentially larger b/c malicious lower-levels
- Level N bug means levels  $\geq N$  are untrustworthy
- Bugs and vulns mimic each other due to abstraction

# Computer and Network Security

## Lecture 14: OS Security & Isolation

COMP-5370/6370  
Fall 2024

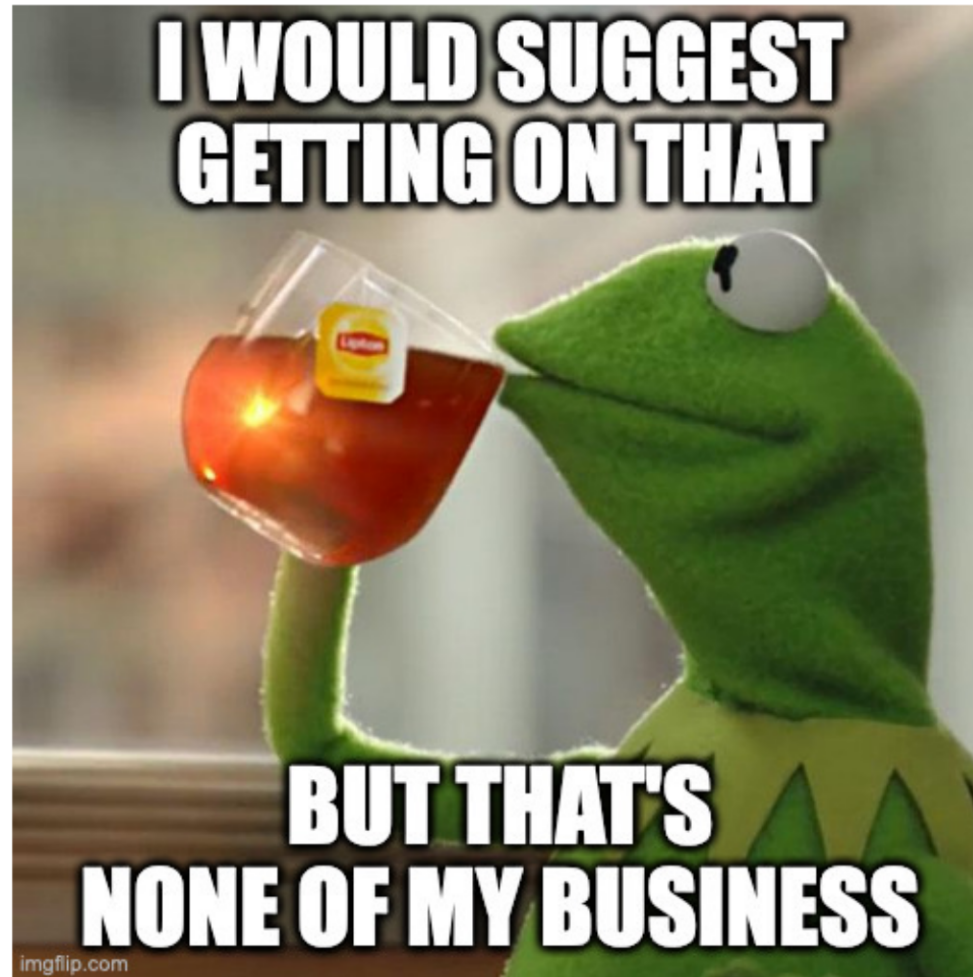




# Course Notes



- Project 2 due **next Sunday** (13Oct2024)
  - If you haven't setup the VM...



# Computer and Network Security

## Lecture 14: OS Security & Isolation

COMP-5370/6370  
Fall 2024

