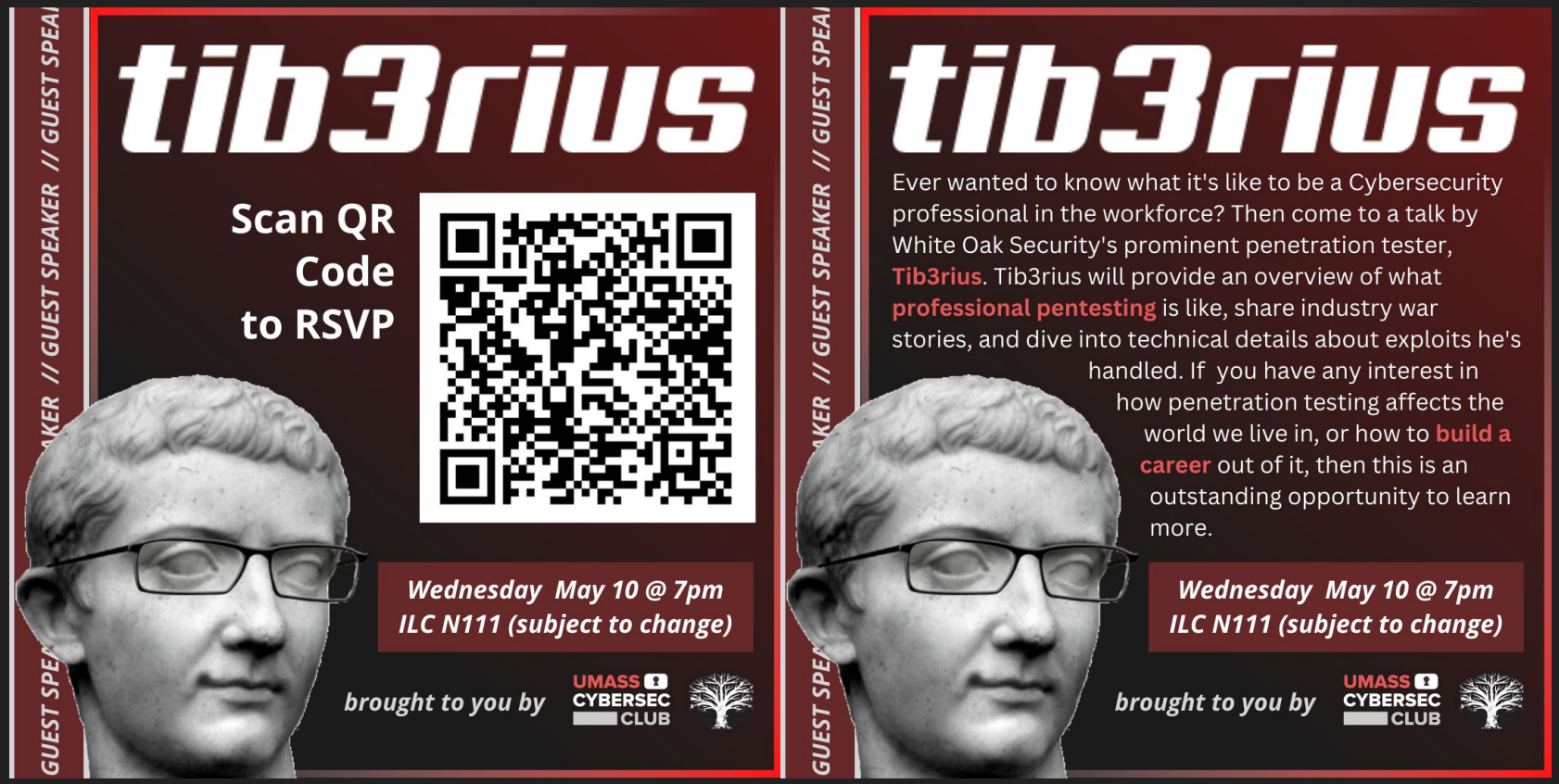
COMPSCI 390R

Emulation

UMASS CYBERSEC CLUB

Topics to Cover:

1. Talk moved to the 17th
2. Presentations starting this Thursday



UMASS CYBERSEC CLUB

What Is Emulation:

A Computer in a Computer:

Emulation is using software to emulate the activities of a target device

How do we accomplish this?

Emulators are usually one of the two forms

- A userland process that doesn't rely on performance too much
- A fully loaded operating system that uses hardware level tools

UMASS CYBERSEC CLUB

Common Use Cases of Emulation

Chip-8/Toy Languages:

Many people experiment writing their own toy programming languages which are interpreted in a way that's basically system emulation.

CHIP-8 is a fairly common cpu architecture that was created for the sake of people learning how to write an emulator

https://en.wikipedia.org/wiki/CHIP-8

Games & Backwards Compatibility:

Emulation is very common in operating systems and consoles. Windows is built off of the principle of backwards compatibility, and it’s very common for older video game consoles to be able to emulate the last generation hardware.
The PS3 has a separate clock counter to emulate the clock of the PS2

People write emulators to copy consoles also

https://github.com/yuzu-emu/yuzu

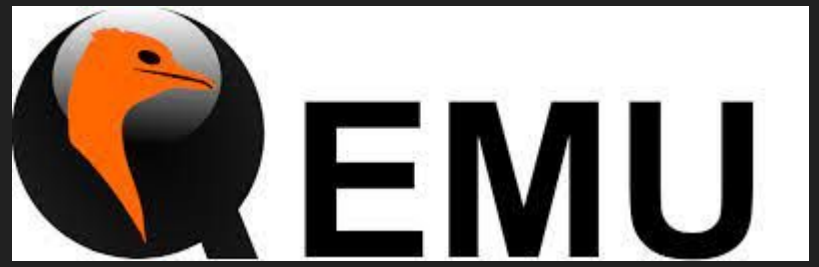
QEMU:

QEMU is an Open-Source Emulation Engine

- Can run multiple architectures & different CPU models for each architecture
- Can run process's compiled for other machines on your native hardware, or even entire operating systems using KVM
- The company that made QEMU gets paid to model systems in QEMU

https://www.qemu.org/

UMASS CYBERSEC CLUB



Snapshots & Debugging:

Given that we are debugging an emulated system which is ***software*** if we can properly emulate a hardware device or piece of software for another architecture, we can apply all the same analysis we've learned so far.

We can take "snapshots" of our device, (freeze memory, cpu state, disk) and can rewind the device whenever needed to test different executions

Emulation for Security

UMASS CYBERSEC CLUB

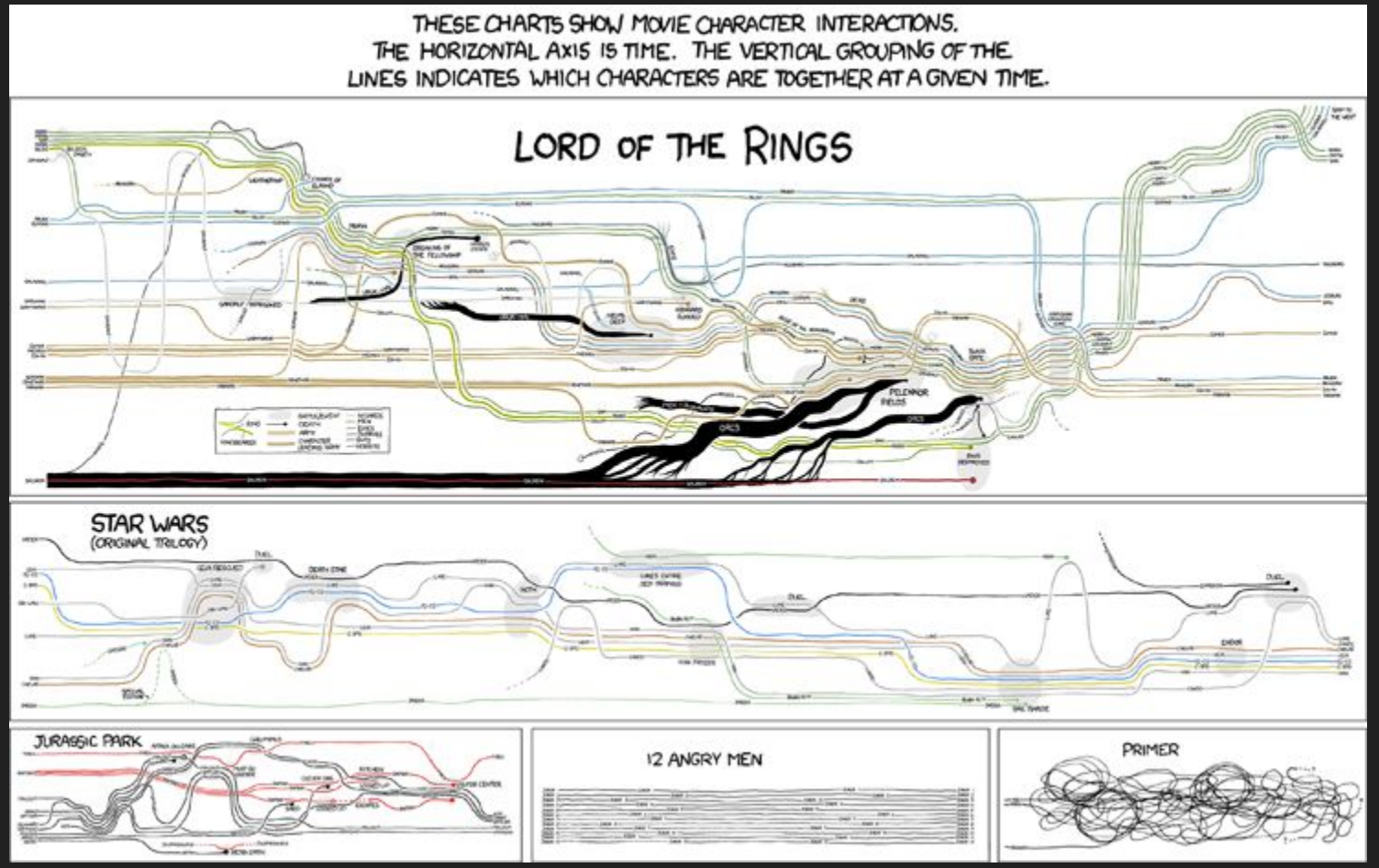
Emulation vs Traditional Debugging:

Modern Emulation systems built for vulnerability research are focused around more introspection and “full-system” analysis.

We can always run gdb on single processes, or windbg on Windows, but newer internal tools can rip out entire kernel structures, analyze them, an see how every part of the kernel interacts together.

Bug Hunting with Emulation:

Modern emulators use what's called Time-travelling Snapshots



UMASS CYBERSEC CLUB

Bug Hunting with Emulation:

1. Take an initial snapshot and setup of your machine
2. Fuzz the entire system focusing on the part you want to test (could be OS, hardware device, etc)
3. Use coverage fuzzing as possible “timelines”, take new snapshots, and continue this process branching.
4. If bugs or new paths are found, we can “rewind” the snapshots to see if we can make better paths or introduce bugs earlier in execution

UMASS CYBERSEC CLUB

Pitfalls of Emulation:

It’s very expensive:

- Without a JIT, emulating cross platform systems is very expensive
- Modern hardware devices can have multiple architectures running all different clocks, and they communicate so you need some form of syncing
- Building these models is also very expensive, a base emulation system is fine but you need the firmware of any chips or implementations of ASICs built out now

The Real World:

The government is very interested in this

Emulation based fuzzing is a big avenue of research for the government right now. Both offensive capabilities and defensive research are looking into this as potential methods for quick research

https://sam.gov/opp/f198c2a5f9ca4dcab48dea65dc5f2c0b/view

How do we do this?

Distributed Time-Traveling Emulation & Fuzzing

1. Given target device, model all components in your emulation software
2. Run each model using time traveling emulation with periodic snapshots. You can split of “branches” with different fuzzing inputs, then let each one run
3. Have a global time sync program which syncs all the devices together, so when they need to communicate (cpu querying gpu), the timeline can sync by rewinding, and all devices and branching can resume

Emulating A Battleship

UMASS CYBERSEC CLUB