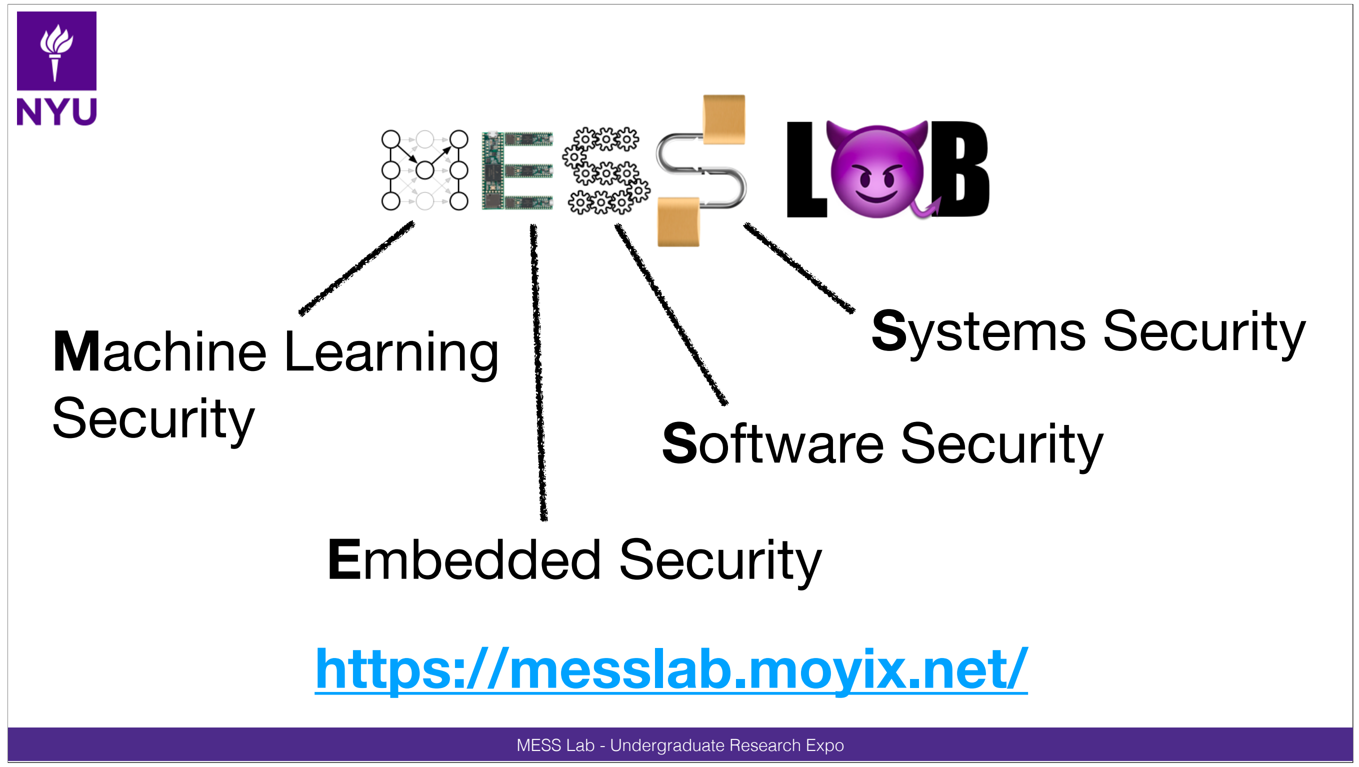
This is the MESS Lab – it's actually an acronym that I'll explain, but it also reflects something about how I view research: it might be messy and you might not always know what to do or what direction to go in, but you can still have fun and build some cool things.



We’re a security lab, so we do research in security across a bunch of different areas: ML, Embedded, Software Security, and Systems Security. I’m going to try to give a 10,000 ft view of some of our work in each area, and then go a bit deeper into one project at the end that’s a bit newer and where we’re actively looking for help.

NYU

Machine Learning Security

- Can ML models be maliciously trained to include **backdoors**?
  - How could we defend against that?
- Do AI programming models like GitHub Copilot write secure code? (**No**)
  - How can we make them more secure?
- How can ML help us find, exploit, and fix vulnerabilities in traditional software?

Backdoors: we trained a street sign recognition system to include a backdoor that will treat **stop signs** as **speed limit** signs when a yellow sticky note is placed on the stop sign.
AI programmers: we measured how frequently GitHub Copilot will generate vulnerable code – **40%** of the suggestions were vulnerable!
And finally, how can we use ML to improve traditional software security, for example by detecting vulnerabilities in code or helping to decompile and reverse engineer programs.

NYU

Embedded Security

- Embedded and IoT devices are **everywhere**
  - Their security? **Not great**
- How can we efficiently **find** and **fix** bugs in embedded devices?
- Can we make it easier to develop **emulators** for embedded devices that let us test them more effectively?
- How can we **protect** existing embedded devices and reduce their exposed attack surface?

There are tons of embedded devices, including lots in this room right now. That camera and that projector probably run some flavor of Linux or a specialty embedded system. So they're *everywhere*, but they often don't have features like automatic updates or modern security protections. So we want to find ways of protecting them.

NYU

Software Security

- How can we make it easier to reverse engineer and understand complex software?
  - We built **PANDA**, a whole-system emulator for reverse engineering with *time travel debugging*
- Can we create highly realistic **synthetic vulnerabilities** to help test and improve bug-finding systems?
- How can we find and fix bugs in complex device drivers like WiFi drivers – *without* the actual hardware?
  - Our prototype fuzzer has found serious bugs in the Linux kernel's WiFi drivers, including two CVEs

PANDA: Platform for Architecture-Neutral Dynamic Analysis

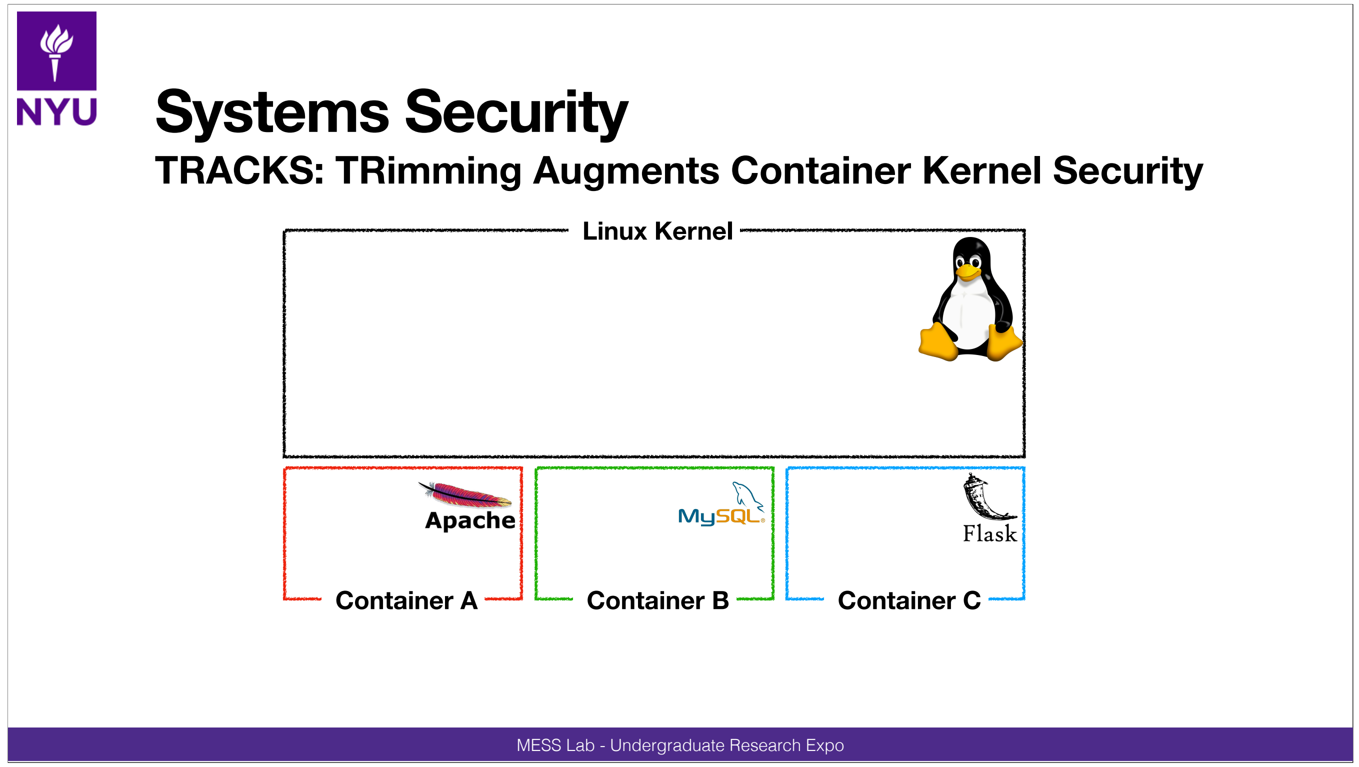
LAVA

LAVA: Large-scale Automated Vulnerability Addition

We also do research in traditional software security. Today's software systems are **big**: millions of lines of code, multiple collaborating processes, and so on. We've built some tools to help reverse engineer and analyze these complex systems, like PANDA, a whole-system emulator with some cool features like *time travel debugging*, which lets you record the execution of a system and then step back in time to see what happened at each step in detail.

Programmers usually try to *reduce* the number of bugs in their software. We've been looking at the opposite problem: how do you automatically add realistic vulnerabilities to a program? Why? Well, one reason is that you might want to see how good a bug-finding system in, and that's hard because we don't know where bugs are to start with.

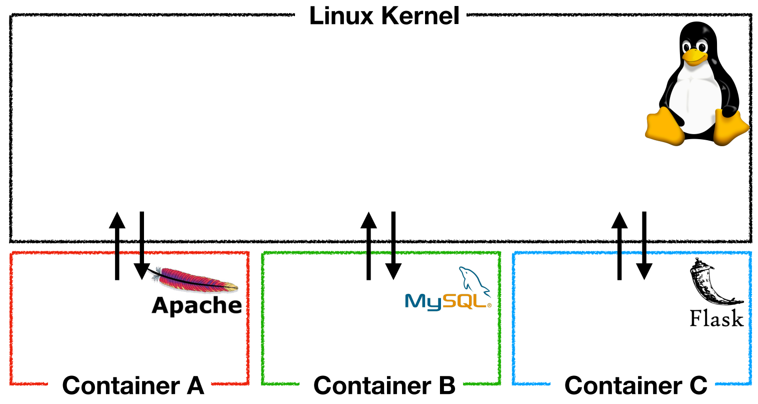
Finally, we've been looking at how to find vulnerabilities in complex and hard-to-test software. If you've ever tried to get a sound card or wifi adapter working in Linux, you might have noticed these drivers can be complicated and finicky. And to test them you usually need to have an actual, physical device. We've been working on ways to test this code *without* the actual hardware, and we've been able to find and fix some serious vulnerabilities in the Linux kernel.



This is a project in collaboration with Justin Cappos's Secure Systems Lab. It's pretty new, but we think it could have the potential to really improve the security for millions of real-world users. In a modern cloud computing environment, applications run in containers, which are designed to give the illusion that each application is running in its own isolated environment.

Systems Security

TRACKS: TRimming Augments Container Kernel Security

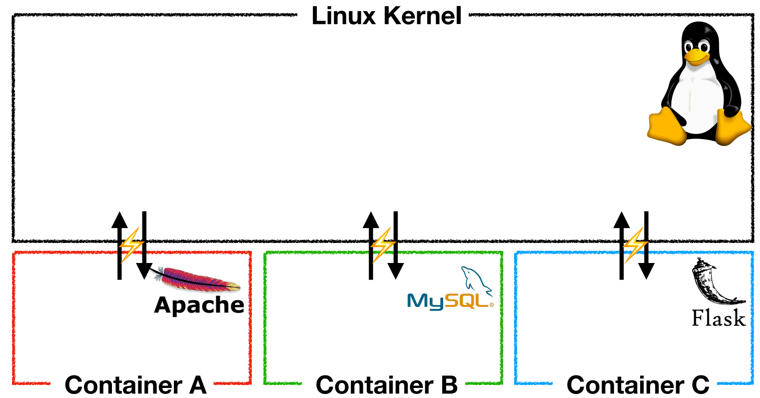


Containers talk directly to the Linux kernel, just like normal processes running on a Linux system

NYU

Systems Security

TRACKS: TRimming Augments Container Kernel Security

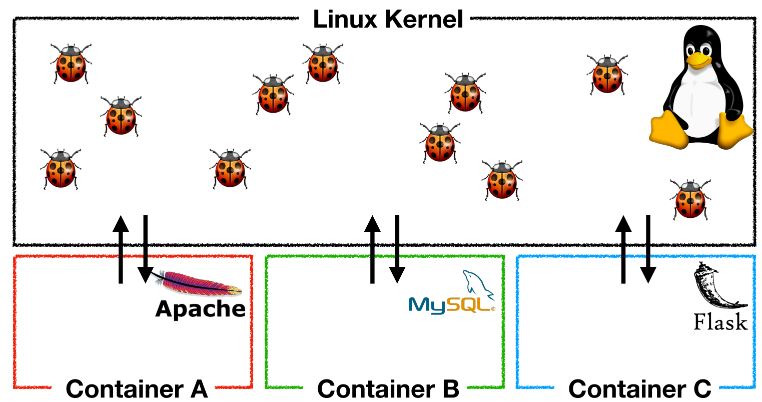


This makes containers *fast* and *lightweight*

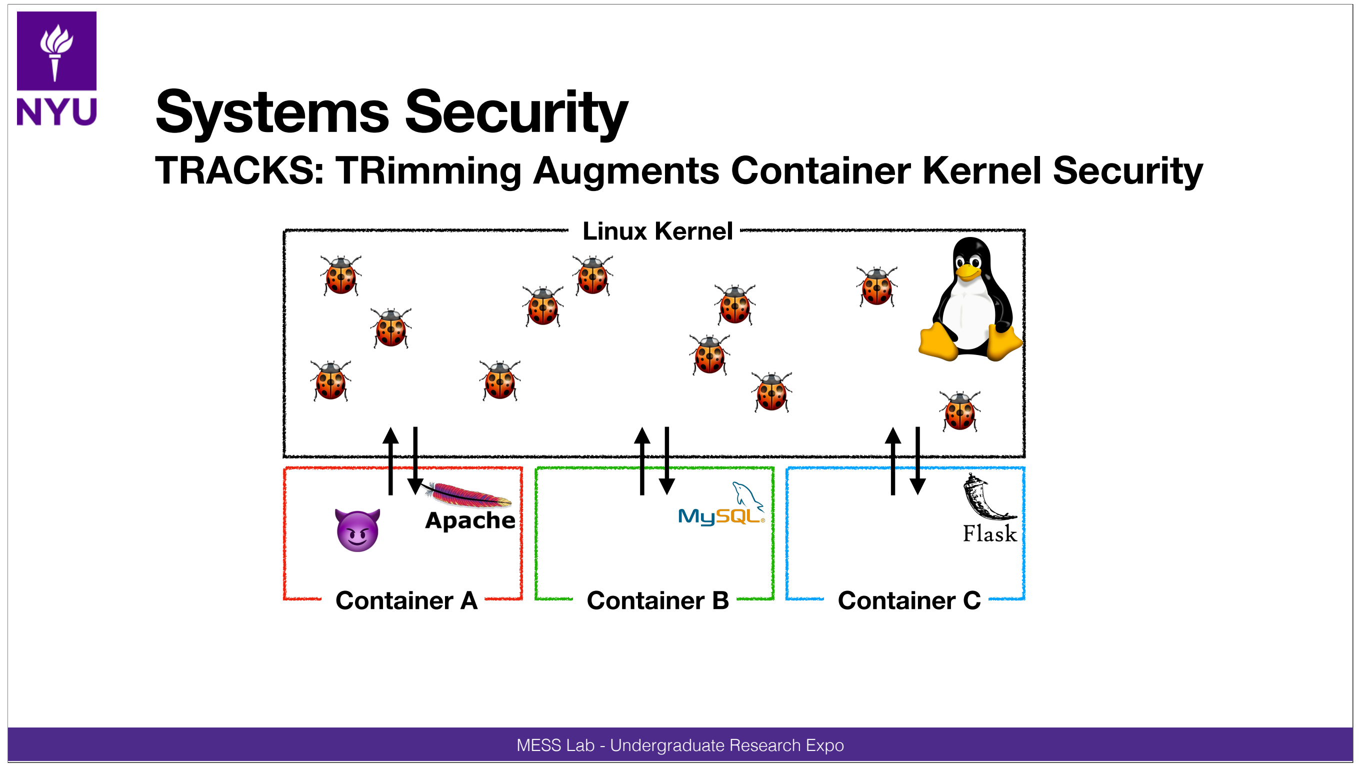
NYU

Systems Security

TRACKS: TRimming Augments Container Kernel Security



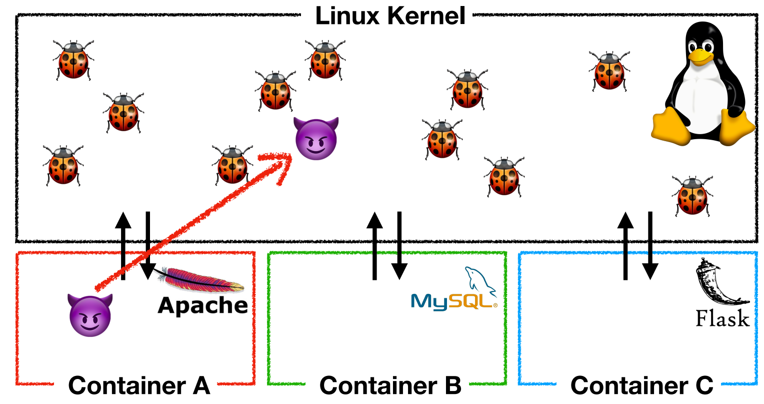
Unfortunately, the Linux kernel has **bugs** – and lots of them! 149 vulnerabilities discovered so far this year



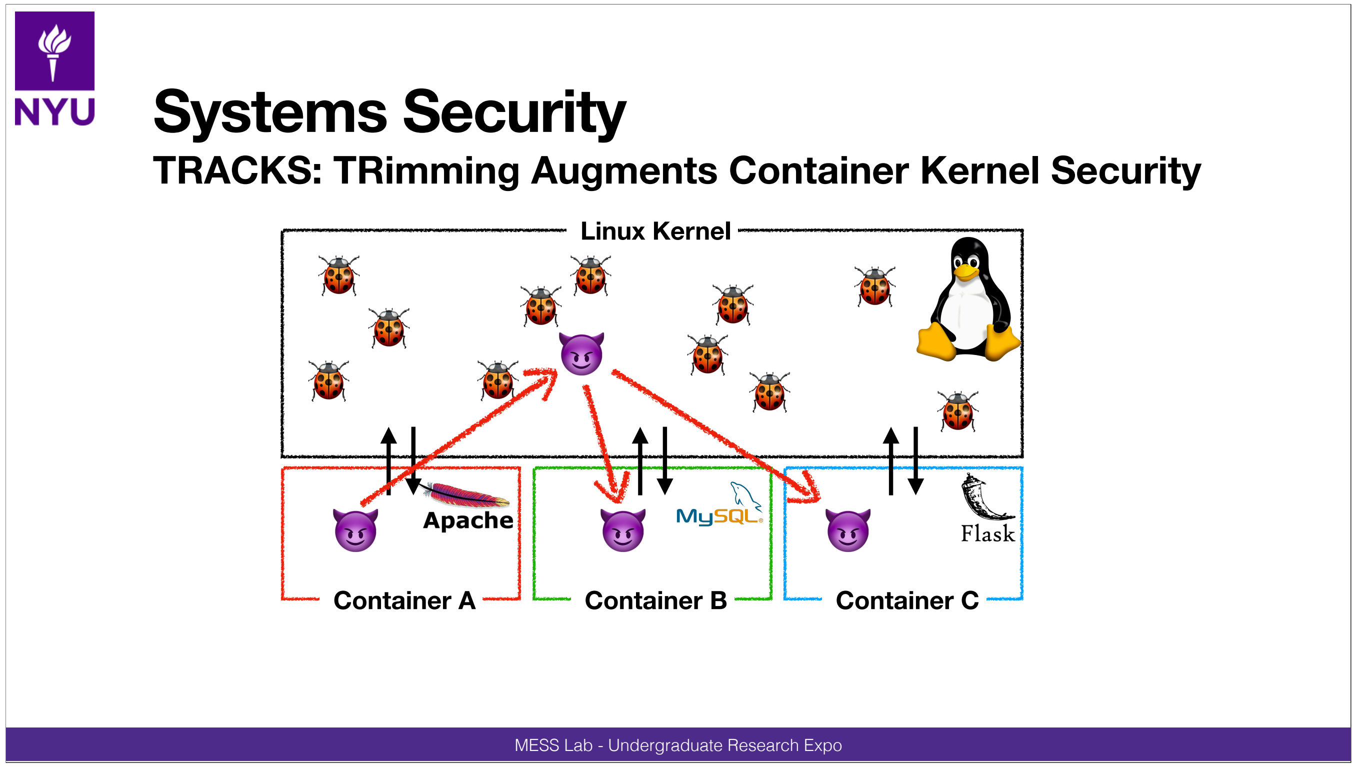
This means that if a container is malicious or compromised...

Systems Security

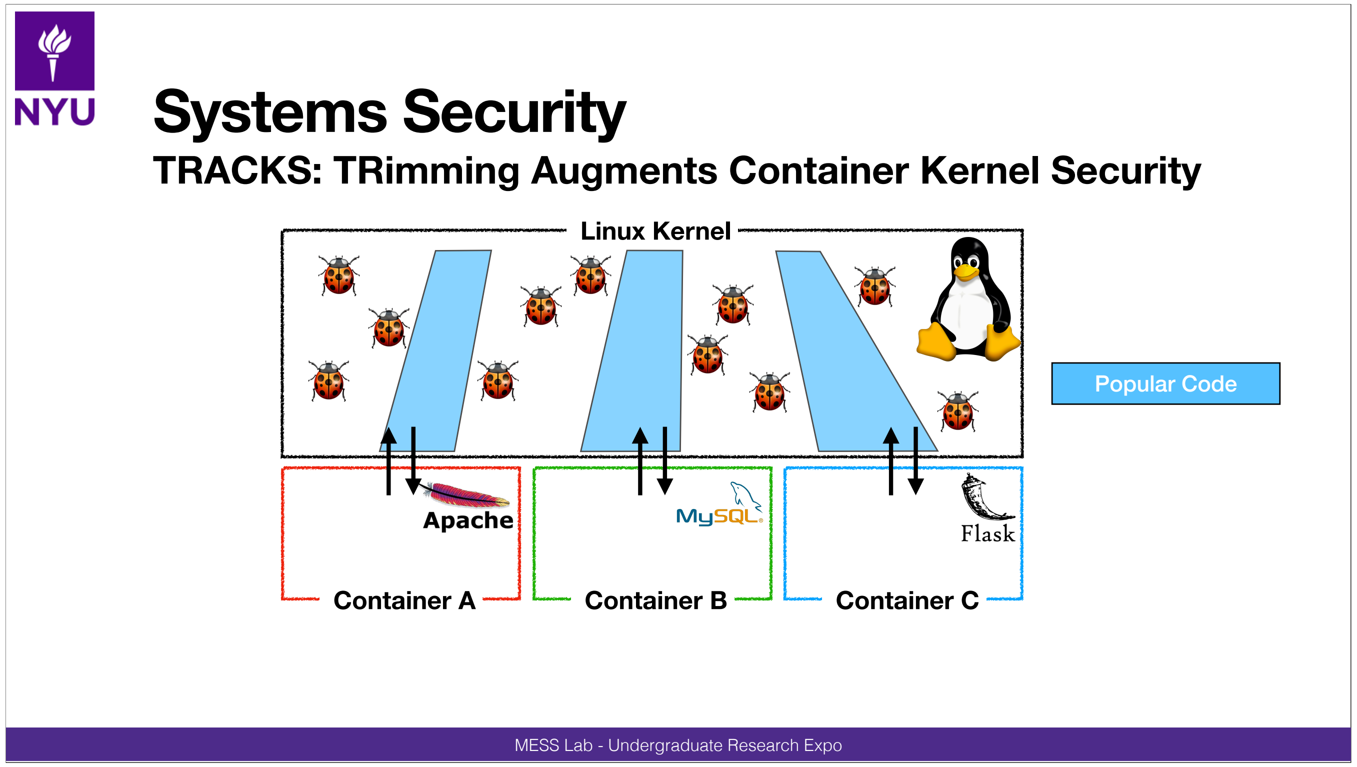
TRACKS: TRimming Augments Container Kernel Security



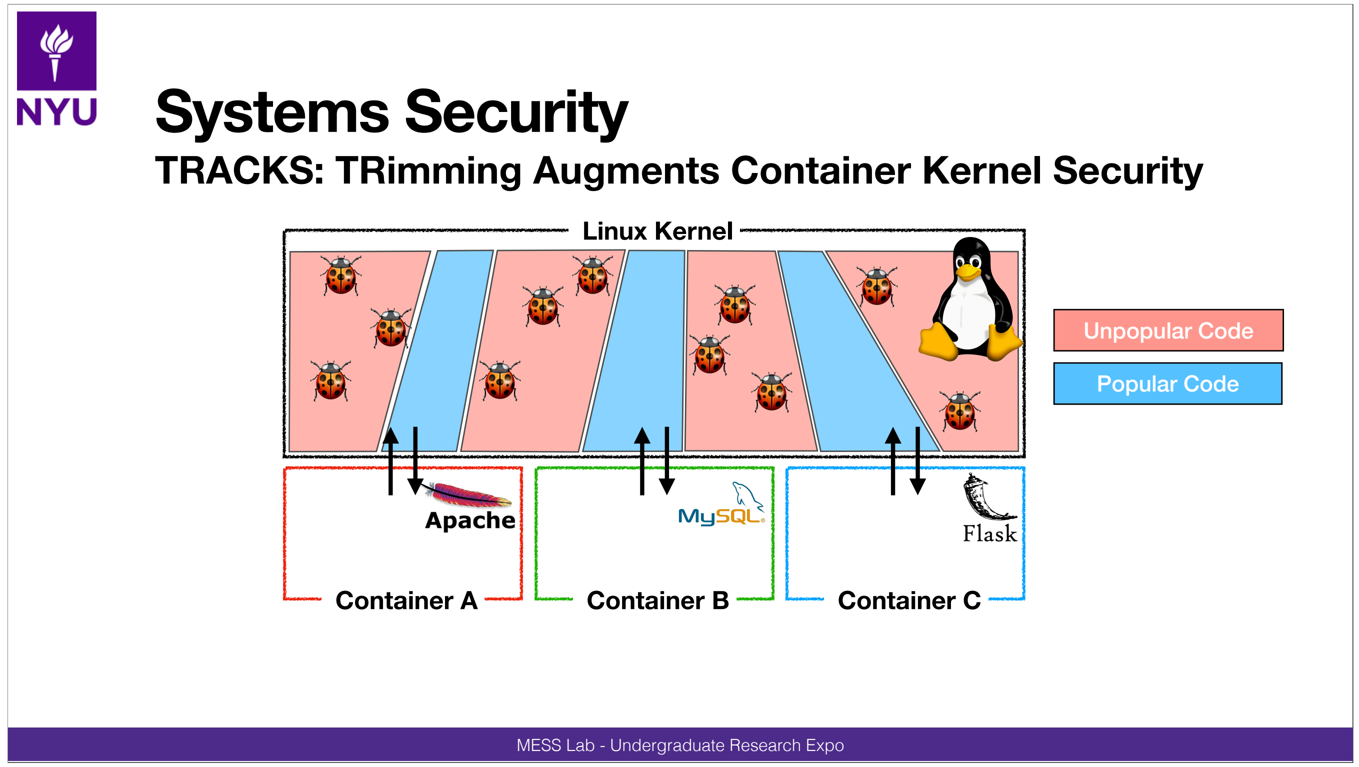
...it can launch exploits against the Linux kernel...



...and then take over other containers, steal their data, etc.



But the Linux kernel is big. Most programs only use a small part of the kernel in normal usage. We call these the **popular paths**.

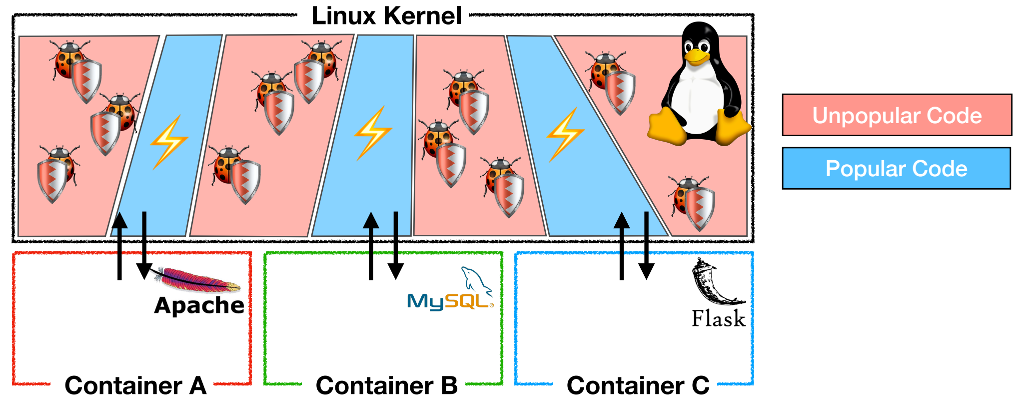


We discovered that these popular paths contain many fewer vulnerabilities! **95%** of kernel vulnerabilities were found in “*unpopular*” code.

NYU

Systems Security

TRACKS: TRimming Augments Container Kernel Security



This means we can turn on **extra security checks** on the **unpopular code**, giving *enhanced protection* without hurting the performance of applications in normal usage!

Join Us!

MESS LAB

- Learn more about our lab and projects at: https://messlab.moyix.net/
- Get in touch via email: brendandg@nyu.edu
- Please include:
  - What topics are you interested in?
  - What experience do you have (classes, research, etc.)?
  - A link to code you've written on GitHub

Don't feel ready yet? Build your skills in the OSIRIS security lab!
https://www.osiris.cyber.nyu.edu/