University of BRISTOL

Reference Monitors

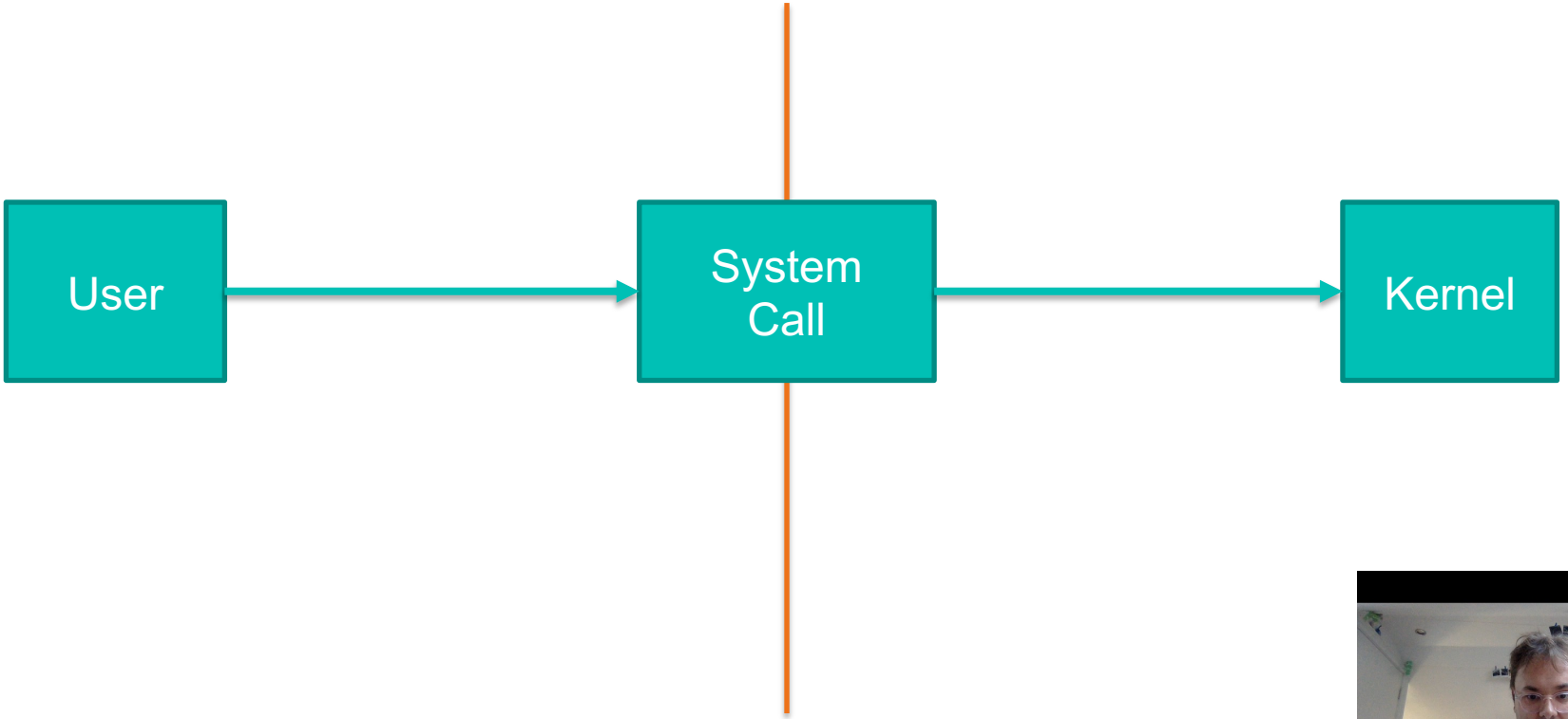
Joseph Hallett

What is it?

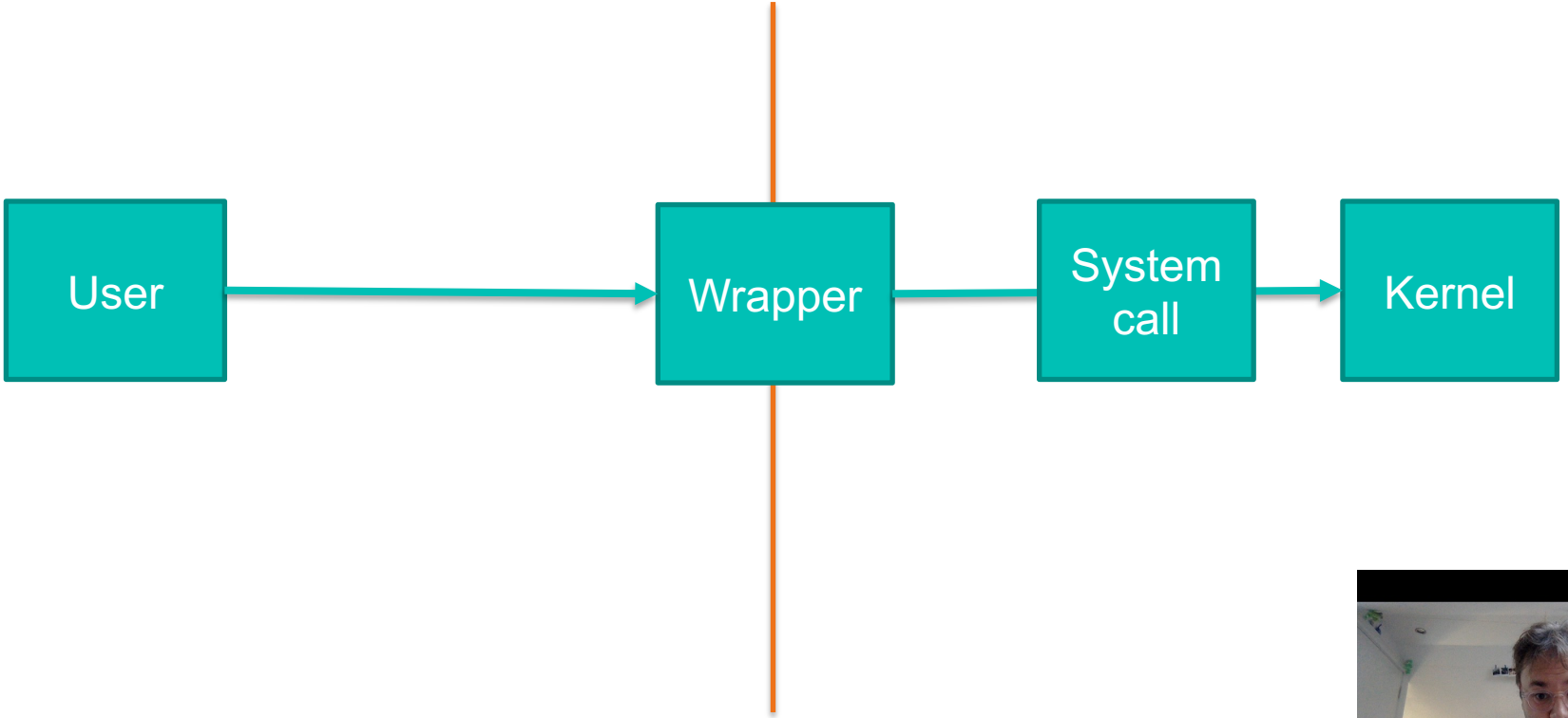
- A reference monitor is *a secure, always-used and fully-testable module that controls all software access to data objects or devices*

- Operating systems control access to privileged resources (e.g. memory, files, processes, hardware…)
- Security policies say *who* should be able to do *what* with *whom*
- The reference monitor helps enforce that policy by controlling what the kernel will allow any user to do.

What is it



What is it



TSEC

Reference monitors are required for systems to meet security standards…
Lots of work implementing them for Linux (SELinux) and other systems…

Implementations have been bitten repeatedly by concurrency and race-condition bugs…

Example

Consider an `open` system call…

- `open("path", O_RDONLY);`

Kernel and userland is separate, so the kernel copies the path into it's memory

- `strncpy_from_user(kname, filename, EMBEDDED_NAME_MAX);`

Does the system call…

Example (with Refernce Monitor)

Consider an open system call…

- `open("path", O_RDONLY);`

Reference monitor checks the call and decides whether to allow or deny (or modify). Copies arguments into kerneland.

- `strncpy_from_user(kname, filename, EMBEDDED_NAME_MAX);`

System call goes ahead, so the kernel copies the path into it's memory as it would without the reference monitor

- `strncpy_from_user(kname, filename, EMBEDDED_NAME_MAX);`

Does the system call…

Example (with Refernce Monitor)

Consider an open system call…

- `open("path", O_RDONLY);`

Reference monitor checks the call and decides whether to allow or deny (or modify). Copies arguments into kerneland.

- `strncpy_from_user(kname, filename, EMBEDDED_NAME_MAX);`

System call goes ahead, so the kernel copies the path into it's memory as it would without the reference monitor

- `strncpy_from_user(kname, filename, EMBEDDED_NAME_MAX);`

Does the system call…

Two time-sensitive copies from userland (which an attacker controls…

Exploiting Concurrency Vulnerabilities in System Call Wrappers

Robert N. M. Watson
Computer Laboratory
University of Cambridge
robert.watson@cl.cam.ac.uk

Abstract

System call interposition allows the kernel security model to be extended. However, when combined with current operating systems, it is open to concurrency vulnerabilities leading to privilege escalation and audit bypass. We discuss the theory and practice of system call wrapper concurrency vulnerabilities, and demonstrate exploit techniques against GSWTK, Systrace, and CerbNG.

1 Introduction

System call interposition is a kernel extension technique used to augment operating system security policies without modifying the underlying code. It is widely used in research systems and commercial anti-virus software despite research suggesting security and reliability problems. Garfinkel [14], Ghormley [15], and the author [23] describe the potential for concurrency vulnerabilities in wrapper systems. However, these discussions are cursory – the vulnerability of wrappers to concurrency attacks deserves further exploration. We investigate vulnerabilities and exploit techniques for real-world systems, demonstrating that inherent concurrency problems lead directly to exploitable vulnerabilities. We conclude that addressing these sys-

All experiments and measurements were performed on a 3.2 GHz Intel Xeon.

2 Kernels and Concurrency

Operating system kernels are highly concurrent programs, consuming concurrency services internally and offering them to applications. Most desktop and server systems support multiprocessing and threading, as do many embedded systems, traditional bastions of minimalism.

In the operating system kernel, hardware sources of concurrency are interrupts and multiprocessing. Kernels provide internal threading facilities to kernel subsystems (file systems, network stacks, etc) and expose concurrency to applications via processes, threading, signals, and asynchronous I/O. Application writers employ these to mask I/O latency and exploit hardware parallelism. Concurrency is a fundamental operating system feature, and must be considered carefully in any security services.

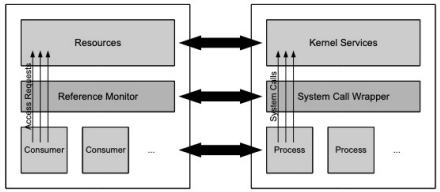


Figure 1: Misleading visual congruence of the reference monitor and system call wrapper models.

3 Wrappers for Security

System call interposition imposes a reference monitor on kernel services by intercepting system calls (Fig. 1). Anderson [4] states that a reference monitor must be tamper proof, always invoked, and small enough to subject to