CSE 127: Introduction to Security

Lecture 6: Secure System Design

Nadia Heninger

Winter 2023

Some material from Dan Boneh, Stefan Savage, Deian Stefan, Keegan Ryan

Last time: Principles of secure system design

- Least privilege
- Privilege separation
- Complete mediation
- Fail safe/closed
- Defense in depth
- Keep it simple

Unix file permissions are a simplified ACL

```
nadiah@login:/cse/htdocs/classes/wi21/cse127-a$ ls -l
total 32
-rw-rw-r-- 1 nadiah cse127-a-wi 18660 Jan 14 00:34 index.html
drwxrwxr-x 2 nadiah cse127-a-wi  4096 Jan 13 08:42 pa
drwxrwxr-x 2 nadiah cse127-a-wi  4096 Jan 13 19:57 resources
drwxrwsr-x 3 nadiah cse127-a-wi  4096 Jan 14 00:34 slides
```

- Permissions grouped by user owner, group owner, other
- Operations: read, write, execute

Process UIDs

Process permissions are determined by UID of user who runs it unless changed.

- Real user ID (RUID)
  - Used to determine which user started the process
  - Typically same as the user ID of parent process
- Effective user ID (EUID)
  - Determines the permissions for process
  - Can be different from RUID (e.g. because `setuid` bit on the file being executed)
- Saved user ID (SUID)
  - EUID prior to change

setuid

- A program can have a `setuid` bit set in its permissions
- This impacts fork and exec
    - Typically inherit three IDs of parent
    - If `setuid` bit set: use UID of file owner as EUID

```
-rwsr-xr-x 1 root root 54256 Mar 26  2019 /usr/bin/passwd
```

`setuid`, `setgid`, and sticky bit

There are three bits:

- `setuid`: set EUID of process to ID of file owner
- `setgid`: set effective group ID of process to GID of file
- sticky bit
  - on: Only file owner, directory owner, and root can rename or remove file in the directory
  - off: If user has write permission on directory, can rename or remove files, even if not owner

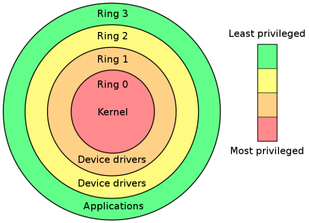
```
drwxrwxrwt  10 root root 12288 Jan 18 20:55 tmp
```

Overview of Unix file security mechanism

- **Pro**: Simple and flexible
- **Con**:
    - Coarse-grained
    - Nearly all system operations require root access.
    - In practice, common to run many services as root. This violates principle of least privilege and increases attack surface.

Kernel isolation

- Kernel is isolated from user processes
  - Separate page tables
  - Processor privilege levels ensure userspace code cannot use privileged instructions
- Interface between userspace and kernel: system calls



Process confinement: system call interposition

Observation: To damage a host system (e.g. make permanent changes), an app must make system calls

- To delete or overwrite files: `unlink`, `open`, `write`
- For network attacks: `socket`, `bind`, `connect`, `send`

Idea: Monitor app's system calls and block unauthorized calls

Key component: Reference monitor

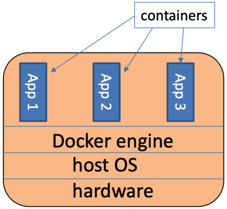
- Mediates requests from applications
    - Enforces confinement
    - Implements a specified protection policy
- Must always be invoked
    - Every application must be mediated
- Tamperproof
    - Reference monitor cannot be killed, or if killed then monitored process is killed too
- Small enough to be analyzed and validated

System Call Interposition in Linux: `seccomp-bpf`

`seccomp-bpf`: Linux kernel facility used to filter process syscalls

- Syscall filter written in the BPF language
- Used in Chromium, Docker containers…

- Container: process-level isolation
- Container prevented from making syscalls filtered by seccomp-bpf



Example: Smartphone OS design

Does the threat model for a smartphone differ from a desktop?

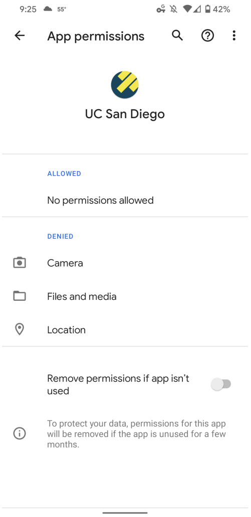
What’s the threat model?

What are the assets?

What security properties do we want to preserve?

Android process isolation

- Android uses Linux and sandboxing for isolation
- Each app runs under its own UID
- Apps can request permissions, which are basically capabilities
- Reference monitor checks permissions on intercomponent communications


9:25 55° 42%

App permissions

UC San Diego

ALLOWED

No permissions allowed

DENIED

Camera

Files and media

Location

Remove permissions if app isn't used

To protect your data, permissions for this app will be removed if the app is unused for a few months.

Software fault isolation (SFI)

Placing untrusted components in their own address space provides isolation, but comes with overhead.

Software fault isolation wants to partition apps running in the same address space.

- Kernel modules shuld not corrupt kernel
- Native libraries shuld not corrupt JVM

Software fault isolation (SFI)

Placing untrusted components in their own address space provides isolation, but comes with overhead.

Software fault isolation wants to partition apps running in the same address space.

- Kernel modules shuld not corrupt kernel
- Native libraries shuld not corrupt JVM

SFI approach: Partition process memory into segments

- Memory isolation: Instrument all loads and stores
- Control flow integrity: Ensure all control flow is restricted to CFG that instruments loads/stores
- Complete mediation: Disallow privileged instructions
- Syscall-like interface between isolated code

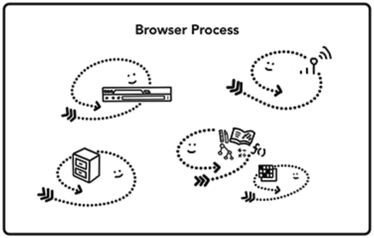
Example: Browser design

What’s the threat model?

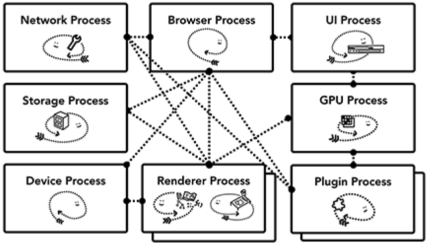
What are the assets?

What security properties do we want to preserve?

Chrome Security Architecture



Pre-2006



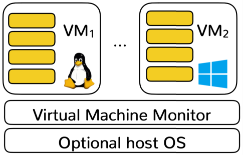
Modern

Modern Browswer Security Model

- Browser process
  - Handles the privileged parts of browser (network requests, address bar, bookmarks)
- Renderer process
  - Handles untrusted attacker content: JS engine, DOM, etc.
  - Communication restricted to remote procedure calls
- Many other processes (GPU, plugin, etc.)

Virtual Machines

- Virtual machines allow a single piece of hardware to emulate multiple machines
- Useful for cloud computing and also for isolation
- Intel has hardware support for x86 virtualization: VMM support in hardware so that operating system can be run in ring 0 without requiring VMM intervention for syscalls



VMs and Isolation

VM Isolation for the cloud:

- VMs from different customers may run on the same machine
- Hypervisor tries to isolate VMs to minimize information leaks

VM Isolation for the end user:

- Qubes OS: A desktop OS where everything is a VM
- Every window frame UI identifies VM source

Hardware isolation: Secure enclaves

- Intel Software Guard eXtensons (SGX)
  - Runs trusted code in an *enclave*
  - Enclave memory encrypted and only decrypted in the CPU
  - Can’t be read even by malicious OS

- Why do we want to protect a program against a malicious OS?

Hardware isolation: Secure enclaves

- Intel Software Guard eXtensons (SGX)
    - Runs trusted code in an *enclave*
    - Enclave memory encrypted and only decrypted in the CPU
    - Can’t be read even by malicious OS

- Why do we want to protect a program against a malicious OS?

  Example applications:
    - DRM (Digital Rights Management)
    - Secure remote computation
    - Protecting crypto keys or sensitive information

iOS Secure Boot

Apple devices use a secure enclave coprocessor as part of its boot chain.

Hardware-based root of trust: code and code-verifying keys baked into boot ROM (read-only memory).

Each step of the boot process verifies that the bootloader, kernel are signed by Apple.

What are the positives and negatives of this kind of design?

Physical isolation: Air gap

To ensure that a misbehaving app cannot harm the rest of the system, you could run it on physically isolated system.

What kinds of systems would you do this for?

What are the downsides?

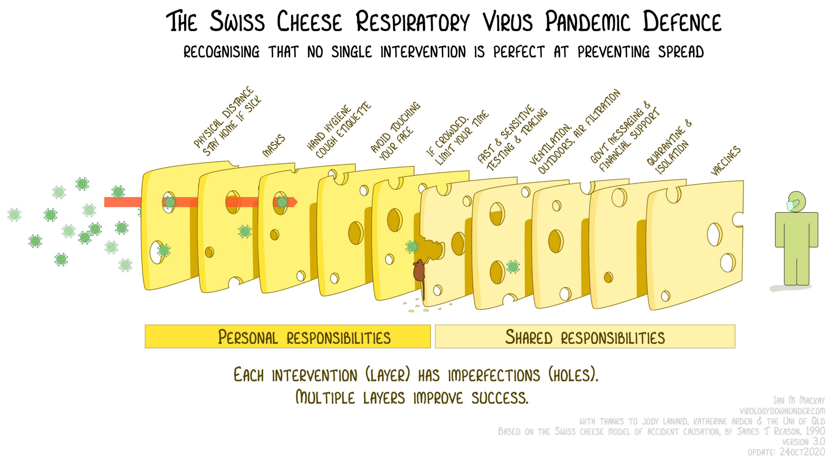
Principles: Fail closed

What’s the problem with failing open?

Why might system designers choose to fail open?

Principles: Defense in depth

We do not expect any of our defenses to be perfect.


The Swiss Cheese Respiratory Virus Pandemic Defence
Recognising that no single intervention is perfect at preventing spread
Physical distance, stay home if sick
Masks
Hand hygiene, cough etiquette
Avoid touching your face
If crowded, limit your time
Fast & sensitive testing & tracing
Ventilation, outdoors, air filtration
Govt messaging & financial support
Quarantine & isolation
Vaccines
Personal responsibilities
Shared responsibilities
Each intervention (layer) has imperfections (holes).
Multiple layers improve success.
Ian M Mackay
virologydownunder.com
With thanks to Jody Lanard, Katherine Arden & the Uni of Qld
Based on the Swiss cheese model of accident causation, by James T Reason, 1990
Version 3.0
Update: 24Oct2020

Principles: Keep it simple

We *have* to trust some components of our system.

In general keeping the Trusted Computing Base small and simple makes it easier to verify.

- In theory a hypervisor can be less complex than a full host operating system.
- A small OS kernel has less attack surface than one with many features.

Software and hardware isolation techniques

- Memory isolation
- Resource isolation and access control
- System call interposition
- Sandboxing
- Containers
- Virtualization
- Secure enclaves
- Physical air gap

Lesson: Complete isolation is often inappropriate; applications need to communicate through regulated interfaces

Principles of secure system design

- Least privilege
- Privilege separation
- Complete mediation
- Fail safe/closed
- Defense in depth
- Keep it simple