CSE543 - Computer and Network Security
Module: Hardware Security

Professor Trent Jaeger
What is Trust?
What is Trust?

- dictionary.com
  - Firm reliance on the integrity, ability, or character of a person or thing.
What is Trust?

- dictionary.com
  - Firm reliance on the integrity, ability, or character of a person or thing.

- What do you trust?
  - Trust Exercise
What is Trust?

• dictionary.com
  ‣ Firm reliance on the integrity, ability, or character of a person or thing.

• What do you trust?
  ‣ Trust Exercise

• Do we trust our computers?
Trust

• “a system that you are forced to trust because you have no choice” -- US DoD
Trust

- “a system that you are forced to trust because you have no choice” -- US DoD

- “A ‘trusted’ computer does not mean a computer is trustworthy” -- B. Schneier
Trusted Computing Base

• Trusted Computing Base (TCB)
  ‣ Hardware, Firmware, Operating System, etc

• There is always a level at which we must rely on trust
Trusted Computing Base

- Helps us enforce security
  - E.g., reference monitor in OS for access control
- Historically, security features have been added to OSes or into programs directly
  - But, may be slow and/or complex enforce security
- How **about adding security features** into the hardware?
  - May still need support from the OS/compilers
  - But maybe we don’t have to trust them…
Buffer Overflows

• Can hardware help prevent buffer overflows from being exploited?
  ‣ How could it help?
Can hardware help prevent buffer overflows from being exploited?
  - How could it help?

One Approach: Intel MPX
  - Instruction set architecture (ISA) extension
  - Set bounds registers - update these from a bounds table
  - Check bounds - check bounds for a pointer
  - Set status - store error code to enable error handling

Approach
  - Store upper and lower bound addresses in a bounds register
  - Use selected bounds register with a pointer use
  - Pointer must be within bounds
Buffer Overflows - MPX

• Of course, somebody needs to setup the bounds information and decide when to check the pointers
  ‣ And deal with violations when they occur

• Operating systems
  ‣ Provides support for memory management for bounds table and exception handling on violation

• Compilers
  ‣ Instruments the original program to track and check bounds

• Runtime libraries
  ‣ Initialize MPX and check bounds before library calls

• Ecosystem for Intel MPX is now available although researchers are just starting to evaluate
Another Use for MPX

- Paper “LMP: Light-Weighted Memory Protection with Hardware Assistance” in ACSAC 2016 used MPX for implementing a shadow stack
- A shadow stack compares return values on stack with expected return values
  - LMP implements such checks by
    - On Call: Copy expected return address to shadow stack
    - On Return: Load expected return address into bounds register and compare to actual return address
  - To protect the shadow stacks, all stores except those in instrumentation are prohibited from accessing shadow stack memory by bounds checks
Control Flow Hijacking

• Can hardware help prevent control flow hijacking using function pointers (call/jmp) and returns?
  ‣ How could it help?
Control Flow Hijacking - PT

- Can hardware help prevent buffer overflows from being exploited?
  - How could it help?

- One Approach: Intel PT
  - Record the control flow decisions made by a program at runtime in a trace buffer
  - Use the trace buffer to evaluate the program control flow to detect errors

- Use for control-flow integrity enforcement
  - Record trace buffers from execution
  - Compare indirect call/jmp targets to expected targets
  - Collect call sites and match returns to expected returns
Can hardware help prevent buffer overflows from being exploited?

One Approach:

- Intel PT
  - Record the control flow decisions made by a program at runtime in a trace buffer
  - Use the trace buffer to evaluate the program control flow to detect errors
  - Use for control-flow integrity enforcement
    - Record trace buffers from execution
    - Compare indirect call/jmp targets to expected targets
    - Collect call sites and match returns to expected returns

**An Example**

- Trace Packets
  - PGE A
  - TNT Taken
  - TNT Not Taken
  - TIP F
  - PGD 0

- Basic Blocks
  - A: jmp D
  - B: jcc E
  - C: call *rax
  - D: jcc B
  - E: ret
  - F: syscall
Control Flow Hijacking - PT

- Can hardware help prevent buffer overflows from being exploited?
  - How could it help?

- One Approach: Intel PT
  - Record the control flow decisions made by a program at runtime in a trace buffer
  - Use the trace buffer to evaluate the program control flow to detect errors

- Use for control-flow integrity enforcement
  - Record trace buffers from execution
  - Compare indirect call/jmp targets to expected targets
  - Collect call sites and match returns to expected returns
Control Flow Hijacking - PT

- Can hardware help prevent buffer overflows from being exploited?

  - How could it help?

- One Approach: Intel PT

  - Record the control flow decisions made by a program at runtime in a trace buffer
  - Use the trace buffer to evaluate the program control flow to detect errors

- Use for control-flow integrity enforcement

  - Record trace buffers from execution
  - Compare indirect call/jmp targets to expected targets
  - Collect call sites and match returns to expected returns
Control Flow Hijacking - PT

• Can hardware help prevent buffer overflows from being exploited?

‣ How could it help?

• One Approach: Intel PT

‣ Record the control flow decisions made by a program at runtime in a trace buffer

‣ Use the trace buffer to evaluate the program control flow to detect errors

• Use for control-flow integrity enforcement

‣ Record trace buffers from execution

‣ Compare indirect call/jmp targets to expected targets

‣ Collect call sites and match returns to expected returns
Control Flow Hijacking - PT

• Can hardware help prevent buffer overflows from being exploited?
  ‣ How could it help?

• One Approach: Intel PT
  ‣ Record the control flow decisions made by a program at runtime in a trace buffer
  ‣ Use the trace buffer to evaluate the program control flow to detect errors

• Use for control-flow integrity enforcement
  ‣ Record trace buffers from execution
  ‣ Compare indirect call/jmp targets to expected targets
  ‣ Collect call sites and match returns to expected returns
Control Flow Hijacking - PT

• Can hardware help prevent buffer overflows from being exploited?
  ‣ How could it help?

• One Approach: Intel PT
  ‣ Record the control flow decisions made by a program at runtime in a trace buffer
  ‣ Use the trace buffer to evaluate the program control flow to detect errors

• Use for control-flow integrity enforcement
  ‣ Record trace buffers from execution
  ‣ Compare indirect call/jmp targets to expected targets
  ‣ Collect call sites and match returns to expected returns
Control Flow Hijacking - PT

• Can hardware help prevent buffer overflows from being exploited?
  ‣ How could it help?

• One Approach:
  ‣ Intel PT
    ‣ Record the control flow decisions made by a program at runtime in a trace buffer
    ‣ Use the trace buffer to evaluate the program control flow to detect errors

• Use for control-flow integrity enforcement
  ‣ Record trace buffers from execution
  ‣ Compare indirect call/jmp targets to expected targets
  ‣ Collect call sites and match returns to expected returns

System Overview

User Space

Kernel Space
Control Flow Hijacking - PT

• Can hardware help prevent buffer overflows from being exploited?
  ‣ How could it help?

• One Approach: Intel PT
  ‣ Record the control flow decisions made by a program at runtime in a trace buffer
  ‣ Use the trace buffer to evaluate the program control flow to detect errors

  • Use for control-flow integrity enforcement
    ‣ Record trace buffers from execution
    ‣ Compare indirect call/jmp targets to expected targets
    ‣ Collect call sites and match returns to expected returns
System Overview

User Space

Kernel Space

Control Flow Hijacking - PT

• Can hardware help prevent buffer overflows from being exploited?
  ‣ How could it help?

• One Approach: Intel PT
  ‣ Record the control flow decisions made by a program at runtime in a trace buffer
  ‣ Use the trace buffer to evaluate the program control flow to detect errors

• Use for control-flow integrity enforcement
  ‣ Record trace buffers from execution
  ‣ Compare indirect call/jmp targets to expected targets
  ‣ Collect call sites and match returns to expected returns
Control Flow Hijacking - PT

- Can hardware help prevent buffer overflows from being exploited?
- How could it help?

- One Approach: Intel PT
- Record the control flow decisions made by a program at runtime in a trace buffer
- Use the trace buffer to evaluate the program control flow to detect errors

- Use for control-flow integrity enforcement
- Record trace buffers from execution
- Compare indirect call/jmp targets to expected targets
- Collect call sites and match returns to expected returns

---

What To Do?

- Depends on the enforced policy
Control Flow Hijacking - PT

• Coarse-grained Policy (any legal target for source)
  ‣ Check if the targets of indirect control transfers are valid
  ‣ Requires decoding the trace packets

• Fine-grained Policy (specific targets for source)
  ‣ Check if the source and destination are a legitimate pair
  ‣ Requires control-flow recovery

• Shadow Stack
  ‣ Check if an indirect control transfer is legitimate based on the reconstructed call stack for entire run
  ‣ Requires sequential processing
Kernel Rootkits

• Can hardware help protect your programs from compromised operating systems?
  ‣ Do you really need to trust the OS?
Kernel Rootkits - SGX

• Can hardware help protect your programs from compromised operating systems?
  ‣ Do you really need to trust the OS?

• One Approach: Intel SGX
  ‣ Define a protected memory “enclave” to run programs
  ‣ Load and run your programs in that enclave
  ‣ Use OS as a untrusted server of resources (encrypted memory and system resources)

• For a program that processes secret data
  ‣ Load program and keys into enclave
  ‣ Read encrypted data from system
  ‣ Decrypt and process that data
Intel SGX

- Challenge - **Side Channels**
- Untrusted operating system can see all the memory accesses from each enclave
- Untrusted operating system can cause memory accesses to occur by unmapping pages
- Researchers have found that such malice can be done on a fine granularity to enable single-stepping of enclaves
- Provides untrusted operating system with a powerful method for detecting the operation of enclaves and possibly leaking data based on their operation
Malware

• Can hardware help protect your systems from running malware?
  ‣ How can hardware help?
Trusted Platform Module

• The Trusted Platform Module (TPM) provides hardware support for *sealed storage* and *remote attestation*

• What else can it do?
  ‣ [www.trustedcomputinggroup.org](http://www.trustedcomputinggroup.org)
Where are the TPMs?
TPM Components

- Non-Volatile Storage
- Platform Configuration Register (PCR)
- Attestation Identity Key (AIK)
- Program Code
- Random Number Generator
- SHA-1 Engine
- Key Generation
- RSA Engine
- Opt-In
- Exec Engine
Tracking State

- Platform Configuration Registers (PCRs) maintain state values.
- A PCR can only be modified through the Extend operation:
  - `Extend(PCR[i], value) :
    - PCR[i] = SHA1(PCR[i] \cdot value)``
- The only way to place a PCR into a state is to extend it a certain number of times with specific values.

Measurement Flow
(Transitive Trust)

- BIOS Self Measurement
- OS Loader Code
- OS Code
- Application Code
Secure vs. Authenticated Boot

• Secure boot *stops execution* if measurements are not correct

• Authenticated boot measures each boot state and lets *remote systems determine if it is correct*

• The Trusted Computing Group architecture uses *authenticated boot*
Using TPMs

• Justify System Integrity

• Attestation Approaches
  ‣ Trusted Platform on Demand (TPoD)
    • IBM Research Tokyo
  ‣ Linux Integrity Measurement Architecture
    • Sailer et. al. (USENIX Security 2004)
  ‣ BIND: A Fine-grained Attestation Service for Secure Distributed Systems
    • Shi et. al. (IEEE S&P 2005)

• Network Authentication
  ‣ Trusted Network Connect (TNC)
    • www.trustedcomputinggroup.org
Integrity Measurement

• IPsec and SSL provide secure communication
  ‣ But with whom am I talking?
Integrity Measurement

Execution Flow

Measurement Flow

TCG-based Integrity Measurement Architecture

Defined by Grub (IBM Tokyo Research Lab)

Defined by TCG (Platform specific)

Platform Configuration Registers 0-23

TCG-based Integrity Measurement Architecture
Basic Idea

Attested System

- Data
- Program
- Config data
- Boot-Process
- Kernel
- Kernel module

Signed TPM Aggregate

- SHA1(Boot Process)
- SHA1(Kernel)
- SHA1(Kernel Modules)
- SHA1(Program)
- SHA1(Libraries)
- SHA1(Configurations)
- SHA1(Structured data)
- ...

Analysis

System-Representation

- Known Fingerprints

System Properties

Measurement

ext. Information
(CERT,...)

CSE543 - Computer and Network Security
Measurement List

- /bin/bash
- SHA1
- Memory Map
- Schedule
- Execve (*file)
- Integrity Value
- Measurement List (Kernel-held)
- Linux Security Module
- Traditional execution path
Some Details

• Kernel Measures
  ‣ Executables, Libraries, Modules

• At
  ‣ Load time only

• Applications May Measure Also
  ‣ Critical input

• **Issues Addressed:**
  • Prevents writing on actively measured files
    ‣ Cannot open for write while file is open
  • Non-deterministic loading
    ‣ Need measurement list