The foundation is rotting and the basement is flooding: A deeper look at the implicit trust relationships in your organization

Mr. Jacob I. Torrey
TROOPERS’15
@JacobTorrey
Who am I?

• Senior security researcher at Assured Information Security
  – Leads Denver, CO office
  – Leads the low-level computer architectures group
  – Plays in:
    • SMM
    • VMM
    • BIOS

• LangSec Cultist

• Avid outdoorsman/fitness nut
• Introduction
• Background
  – Threat modeling
  – Low-level attack surface
  – Technical Debt
• Who you trust, and don’t realize you’re trusting
  – Mapping your trusted computing base (TCB)
  – An example of pivots
  – Less is more
• Selling InfoSec
  – Win themes
  – “InfoSec debt”
• Conclusions
• Information security is always seen as a cost to doing business, not a way to help achieve business goals

• I have been collaborating with a number of CISOs/Dir. Of IT Security in recent months and provide an “adversarial mindset”

• By bringing an attacker’s perspective to the table, you can identify threats to business and provide a better ROI
  – Focus on supporting business, not Infosec as be-all, end-all
Background: Threat Modeling

• “Chess vs. Poker”
  – InfoSec research focuses on elegance
  – Attackers and users of technology focus on ease-of-use/convenience or ROI

• As InfoSec researcher/professional
  – I am rewarded for “neat tricks” and elegant exploits – synthetic/fabricated environments
  – You are rewarded for deploying solutions and new defenses – not making things more “secure”
• Everyone who uses technology is an information security practitioner: “cyber civilians”
  – Just want to accomplish their goal

• Attackers are motivated by ???
  – Money? – The easiest way to make money will provide highest ROI – just like you!!!
  – Revenge? – Their goal is destruction, not hiding their tracks
  – Fun? – Aims for soft targets and moves on
Metric used to track growing gap between product in reality and in a “perfect world”
- If you accept no technical debt, you will get to market late
- If you take on too much technical debt, your product will be unstable and impossible to maintain

Helpful concept to sell product investors on development
- A little more $ now will save $ later
- Maintenance over life-cycle may outstrip initial development costs if too much technical debt is taken on
• The body of code that executes as part of “privileged” container
• Privilege can be defined as:
  – Administrator privilege
  – OS/Kernel privilege
  – Hypervisor privilege
  – Access to sensitive data ← commonly overlooked!
  – Humans with access
• Goal: Shrink this as small as possible
• Measure/protect this codebase as other code running will not be able to access sensitive data…
Mapping the TCB

• Unfortunately, this is extremely hard?
• Example: Intel legacy boot process
  – BIOS is loaded from ROM into RAM ← BIOS vendor
    • Generally hashed and checked
  – BIOS loads ISA/PCI option ROMs off of devices
    • Video card ← GPU vendor
    • NIC ← NIC vendor
    • RAID controller ← RAID vendor
  – BIOS loads OS from disk ← OS vendor
    • Can be hashed and checked
    • Could be run under virtualization ← Hypervisor vendor
• You are trusting each of these vendors before your application is even run!
• Now that your application is running:
  – Libraries/tool-kits you link against
  – Drivers for every device you install
    • Plug in a USB device, run a driver wrote by vendor or individual!
  – Everything running with more privilege than your application:
    • Anti-virus solution
    • DLP
    • OS
    • Hypervisor
    • BIOS/firmware
• A bug in any of these could be the entry point for attacker!
  – Or consider a malicious developer at XYZ corporation adds back-door to your printer driver!
• Do you vet these vendors via typical vendor evaluation process?
Next couple slides show a few low-level attack at x86 architecture level

Unlikely to be used against your organization

Highlight that there is always a way in for a sufficiently determined attacker
  – If you’ve got one of those, you are already failing

You want to ensure your organization is not the target of a sophisticated attacker
Stepping p3wns(2013) – A. Cui et al.

Showed that pivoting through printer would allow remote shell from behind firewall

Just by printing a document, printer was infected

Could infect IP phones and smart switches for stronger foothold into network
Background: Low-level attack surface

  - Showed that the reference implementation for most modern systems’ firmware was vulnerable
  - Most firmware vendors copied reference implementation
  - Could escalate from user application to firmware
Background: Low-level attack surface


- Showed that x86 hardware could be misused to hide malware from OS protections and anti-virus
  - Bypass anti-kernel patching
  - Anti-virus could not detect modifications to code

- Split view of memory data vs. code
  - Reading memory gives different output than executing it
  - No way to measure what is running
Virtual Memory Security

- Paging/virtual memory is a protective feature/promise
  - First code in will be able to control system – usually BIOS/OS

- Unless you can access the pages tables, you are locked out (until now)
  - Can’t add mappings to page tables unless you have a mapping to the page table

- Protects against certain classes of attack
Cluck Cluck Goal

- **Goal:** Map in arbitrary physical memory
  - Requires modifying page tables – need to know where they are in virtual memory

- Can be kernel shell-code, live memory forensics, etc.

- Have ring-0 access, but confined to OS-controlled mappings
  - Cannot access MMIO devices for example

- **OS independent**
Problem

- Only know where in physical memory (CR3) the page tables are

- Cannot map in the page tables without having the page tables mapped in already
  - The OS usually has a hard-coded value (0xC0000000 in many Windows systems)
  - OS-specific attacks are lame, let’s exploit the architecture!

- You do not know where your code is executing since you cannot access the page tables
Need control over just 32-bits of memory at a known physical address
  ◦ This is the crux
  ◦ Can bootstrap a recursive mapping

Enhanced Configuration Access Mechanism
  ◦ PCIe has more configuration space per device
  ◦ Port I/O is slow
  ◦ Need a way to access it faster

ECAM shadows device configuration space into physical memory
  ◦ Base address is stored in PCIEXPBAR register
Solution

- **Construct a PDE that maps in the page directory (recursive entry)**
  - Use the CR3 physical address and mark it as present/RW/PS

- **Utilize Port IO to insert new PDE into PCI configuration space**
  - We have just modified what the CPU thinks is physical memory through port IO!

- **Determine physical location**
  - MCH stores the PCI base address in a configuration register (port IO again!)
Solution II

But where can our PDE go?
- Can’t trash random registers or system may crash!

Thank you Intel for the SCRATCHPAD DATA register
- “This register is for software use, it has no functionality”
- 32-bits of beautiful storage right in the MCH (D0:F0)
- Port I/O access to physical memory, write that PDE!

Determine physical location
- MCH stores the PCI base address (PCIXBAR) in a configuration register (port IO again!)
Solution III

- **Change CR3 to point to PCI configuration space**
  - Kernel code is marked as Global, thus the TLB will cache the code segment, so the box won’t crash
  - The CPU doesn’t know that it’s doing anything wrong (using PCI config like this is wrong) and the MCH doesn’t know how the CPU is using the memory!!!

- **Scan the ‘real’ page directory (we know where it is now) for an empty entry and put our PDE there**

- **Switch CR3 back (yes this works!)**

- **Profit! All in a few lines of ASM**
  - You have a virtual pointer to the page tables!
Caveats

- **Alignment** – PDE and CR3s are not aligned, requires some bitwise operations

- Needs PCI registers that are OK to be trashed (like the MCH’s scratchpad register)
  - There are plenty of options on modern systems

- This technique requires Ring-0 and global pages
  - Can be done from ring-3 with IOPL
Design Flaws

- Classic case of feature creep
- PCIe ECAM is for higher performance
- Violates assumptions

- This has happened before
  - SMM caching bug
  - Virtual Machine side-channels
  - Etc…
Pivots in an attack

- Attackers are lazy
  - Aim to accomplish goals as easily and quickly as possible
  - Easier to attack a legacy service running under an employee’s desk than fully patched and firewalled server in NOC

- Will aim for soft targets first, perform recon of network, and pivot to goal systems
  - May happen multiple times

- Most organizations focus on perimeter defense
  - Hard exterior with a soft, gooey filling (vendors!!!)
  - Once perimeter has been breached, game over
Less is more

• The less you have in your network’s TCB, the better!

• Hosting on the cloud (or with cloud model) can de-privilege your organization’s network
  – Move from this:
Less is more

• To this:
• You have now de-privileged the majority of your organization!
  – Least privilege principle
• Shrinking TCB to only include the cloud applications
• Penetrating your organization’s office network much less beneficial to attacker
  – OSINT less valuable
• Less trust of unknown entities (other than cloud provider)
• Million dollar question (literally!):
  – How do you communicate the value-add that security brings to an organization when it is constantly seen as a cost

• Need:
  – Common language to speak to other organizational stakeholders
  – Holistic view of threats and adversary
  – Metric(s) to track progress and ROI
  – Knowledge of when enough is enough
Selling InfoSec: Win Themes

• Need to steer dialogue towards positive: create “win themes” for your security practice

• By implementing less is more, can slim operations and minimize costs in long-run

• Why? You as CISO/defender are the most impacted in breach
  – Company: A-OKAY!
  – Customers: Grumble, but OK
  – You: Checking out indeed.com
• In order to properly protect your organization, you need to know what from:
  – Low-hanging-fruit attackers (automated, script kiddies, etc…)
  – Everyday thieves (looking for profit, don’t care about your company in particular)
  – Advanced, targeted threat (targeting your company, invested in successful exploitation)
Selling InfoSec: Model of Adversaries II

• This is not “threat intelligence”, or pen-testing
  – Different goals

• This is looking at your organization and imagining your adversary’s incentives
  – Are you one-of-many or do you stand out
  – What motivates them, and how do you shift their behavior?
• Know your organization is not monolithic
  – By implementing least privilege principle and breaking network into logical units that are mutually untrusting you may find savings

• Research competitors to compare
  – When running from a (non-targeted) bear, you only need to outrun the other guy, not the bear!
Selling InfoSec: Metrics

• You cannot sell something you cannot measure!
• Metrics **must** be understandable to all stakeholders
• “InfoSec Debt” – Use similar model to translate technical details into a fiscal model that is easy to align with business goals
  – Remember you are there to support the business goals!
Selling InfoSec: InfoSec Debt

• Inventory network
  – Each device is a risk, remember the implicit trust in unknown entities that each device brings
  – Each device has a maintenance cost: patching, IT support, monitoring, log data

• Inventory data
  – Data can be a liability if breached
  – “Crown jewels” are worth a substantial % of company value
• Predict costs out 1, 3, 5 years
  – Just like technical debt, costs over time could exceed short-term savings
    • Ex. Product A costs $10,000 more, but automatically is patched and doesn’t open ports for debugging, Product B is cheaper, but will require specialized attention over its lifetime.

• Find balance between security and usability
  – No InfoSec debt = a turned-off computer in a safe
• How much debt is too much? Too little?
  – Never invest more than your data/network is worth
  – Will you save more technical debt at the expense of business goals? Sometimes a worthy trade-off, now you can measure and compare apples to apples!

• Remember, just like normal debt, it grows over time
  – InfoSec Debt is variable rate:
    • Exploits make their way into kits
    • Automated scanners can detect your weaknesses
    • Maintaining a legacy appliance gets more costly as it stops being able to support new protocols/methods
    • Vendors stop patching old appliances (look at Android phones!)
Concluding Remarks

• A sufficiently determined attacker will be able to find a way into your network
  – Need to model risk & adversary and protect accordingly
  – If you’re being specifically targeted, you’re already failing

• InfoSec doesn’t need to only be a cost
  – Can provide an ROI
  – Bolster brand

• Measuring and tracking “InfoSec Debt” allows you to defend security costs to organizational stakeholders
  – Track progress and improve buy-in
  – Compare vendors objectively
Questions

- Thank you

- Any questions?