Threat Modeling for Modern System Firmware

Robert P Hale, Vincent Zimmer
Intel® Corp.
Agenda

• Introducing threat modeling
• The big BIOS assets
• Next steps
Why a threat model?

• “My house is secure” is almost meaningless
  – Against a burglar? Against a meteor strike? A thermonuclear device?
• “My system is secure” is almost meaningless
  – Against what? To what extent?
• Threat modeling is a process to define the goals and constraints of a (software) security solution
  – Translate user requirements to security requirements
• In this presentation we summarize the results of the threat modeling effort for our UEFI / PI codebase
  – We believe the process and findings are applicable to driver implementations as well as UEFI implementations in general
Defining, using a threat model

• A Threat Model (TM) defines the security assertions and constraints for a product
  – Assets: What we’re protecting
  – Threats: What we’re protecting it against
  – Mitigations: How we’re protecting our Assets
• Use TM to narrow subsequent mitigation efforts
  – Don’t secure review, fuzz test all interfaces
  – Select the ones that are critical
• TM is part science, part art, part experience, part nuance, part preference
  – Few big assets vs lots of focused assets
We don’t always get to choose our Assets

SMM

BIOS Flash

Security “Researchers”

UEFI, TCG, OSV

Internal Research

UEFI Summer Summit – July 2013

www.uefi.org
Flash**

- NIST SP800-147 says
  - Lock code flash except for update before Exit Mfg Auth
  - Signed update (>= RSA2048, SHA256)
  - High quality signing servers
  - Without back doors ("non-bypassability")

- Threats
  - PDOS – Permanent Denial of Service
    - System into inefficient room heater
  - Elevation of privilege
    - Owning the system at boot is an advantage to a virus

- Known attacks
  - CIH / Chernobyl 1999-2000
  - Mebron 2010

- Mitigations include
  - Reexamining flash protection methods – use the best even if its new
  - Using advanced techniques to locate and remove (un)intentional backdoors

Make core flexible enough for a variety of hardware-based protections
• SMM is valuable because
  – It’s invisible to Anti Virus, etc
  – SMM sees all of system RAM
  – Not too different from PCI adapter device firmware
• Threats
  – Elevation
    • View secrets or own the system by subverting RAM
• Known attacks
  – See e.g. Duflot’s Security Issues Related to Pentium System Management Mode **
• Mitigations include
  – Validate “external” / “untrusted” input
  – Remove calls from inside SMM to outside SMM
Resume from S3

• ACPI says that we return the system to the S5 → S0 configuration at S3 → S0
  – Must protect the data structures we record the cold boot config in
• Threats
  – Changing data structures could cause security settings to be incorrectly configured leaving S3
  – Reopen the other assets’ mitigated threats
• No known attacks
• Mitigations include
  – Store data in SMM -or-
  – Store hash of data structures and refuse to resume if the hashes don’t compare
Tool chain

• Tools create the resulting firmware
  – Rely on third party tools and home grown tools
  – Incorrect or attacked tools leave vulnerabilities
• Threats
  – Disabled signing, for example
• Known attacks
  – See e.g. Reflections on Trust, Ken Thompson**
• Mitigation
  – Difficult: For most tools, provided as source code
  – Review for correct implementation
  – Use static, dynamic code analysis tools
    • PyLint for Python, for example

** CACM, Vol 27, No 8, Aug, 1984, pp. 761-763
Boot flow

• Secure boot
  – Authenticated variables
  – Based on the fundamental Crypto being correct
  – Correct location for config data
• Threats
  – Run unauthorized op roms, boot loaders
  – PDOS systems with bad config variables
• Known attacks
  – Researchers
• Mitigations include
  – Sanity check config vars before use, use defaults
  – Reviews, fuzz checking, third party reviews, etc.
TM to Modules: Boot flow

Authenticated Variables
boot control variables
Setup Variables

Var Svcs → Auth Var Svcs
PE/Coff Loader → Crypto

Authenticated Variables

File System

USB Stack
ATAPI Stack
SCSI Stack
LAN Stack

USB Dev
ATAPI Dev
SCSI Dev
LAN NIC

Buff o’flow to pwn sys. Sanity check config pkts.

Classic net attacks Fuzz test packets
Assets or not?

• Variable content sanity checking?
  – If you randomly fill in your Setup variables, will your system still boot?
  – Fit in as a part of boot flow

• ACPI? We create it but don’t protect it

• TPM support? We fill in the PCRs but don’t use them

Quality ≠ Security
Call to action

• What are your assets?
  – How will they attack you?
  – How will you mitigate their attacks?
  – How will you verify that you’ve done your job well?
• If you use someone else’s code you are implicitly using their threat model
  – Does their threat model match yours?
• For more info, see
  – Books, e.g. *Threat Modeling*, Swiderski and Snyder, Microsoft Press
  – Presentations from e.g. CanSecWest, Blackhat
  – Websites, e.g. Microsoft’s SDL site**

Thanks for attending the UEFI Summerfest 2013

For more information on the Unified EFI Forum and UEFI Specifications, visit http://www.uefi.org

Presented by

intel®
Backup
Secure Development Lifecycle must be a part of any modern firmware project

Threat modeling is the first major activity and drives the rest of SDL

This presentation is the result of the ongoing Threat Modeling activity Intel does on its UEFI/PI codebase

The results are, we believe, applicable to UEFI implementations in general

The methodology has proved useful to e.g. driver implementations as well
Assets or not?

- ACPI? We create it but don’t protect it
- TPM support? We fill in the PCRs but don’t use them
- Variable content sanity checking?
  - If you fuzz your setup variables, will your system boot?
  - Fit in as a part of boot flow
System firmware assets

• We don’t always have a choice as to the assets to protect
• NIST says protect your flash
• Researchers say protect your SMM
• TCG, OSVs say protect your boot flow
• Our research says protect your S3 script
• Build tool chain