5. Trusted Computing
ENEE 757 | CMSC 818V

Prof. Tudor Dumitraș
Assistant Professor, ECE
University of Maryland, College Park

http://ter.ps/757
https://www.facebook.com/SDSAtUMD

Today’s Lecture

• Where we’ve been
  – Introduction to network and distributed system security
  – Statistical inference and machine learning
  – Crypto review

• Where we’re going today
  – Trusted computing

• Where we’re going next
  – Passwords
Pilot Project

• Report due by **end of today**

• 1–2 page report
  – **Hypothesis**
  – **Volume**: how much data (e.g. number of rows, columns, bytes)
  – **Velocity**: how fast is the data updated
  – **Variety**: how to access/analyze the code programmatically
    • JSON/CSV/DB dump, screen scrape, etc.;
    • What language / library to use to read the data
  – **Data quality**

Paper Presentations

• Sign up for presentations at [https://docs.google.com/spreadsheets/ccc?key=0ArRU_2H7IDkwdG5SYktnbUi5OEM4MDIKLVZldlBCOXc#gid=2](https://docs.google.com/spreadsheets/ccc?key=0ArRU_2H7IDkwdG5SYktnbUi5OEM4MDIKLVZldlBCOXc#gid=2)

• Each student must sign up to
  – Present one paper **AND**
  – Scribe one in-class discussion **AND**
    • Present one more paper **OR**
    • Scribe 5 more papers

• Sign up by **EOD Wednesday**
Homework Submissions

- Use the submit command on GRACE
  - Log into grace.umd.edu
  - submit <year> <semester> <college> <course> <section> <assignment> <filename>
    - Example: submit 2014 fall ene 757 0101 1 passwords.bib
    - For more information on GRACE: http://www.grace.umd.edu/

- Submit BibTeX files in plain text
  - No Word DOC, no RTF, no HTML!
  - Do not remove BibTeX syntax (e.g. the @ sign before entries)
    - This confuses my parser and I may think that you did not submit the homework if I don’t catch the error!

- Submission deadline: 6 pm the day before class

Who Can You Trust?

- Where is the request “from”?
  - The user? The workstation? The application? The network channel?
    - All of the above?
  - Which of these actors do you trust?
Trusted Computing Base (TCB)

- **TCB** = the collection of hardware and software that **security depends on**
  - Thus, **TCB** should be as small as possible
    - Must be carefully tested, analyzed and protected
    - **“Trust”** is something that is **bad** and must be avoided when possible

- Trusted components may operate alongside untrusted components
  - Example: you cannot trust every host connected to the Internet

- A **trustworthy** system or component is one that will not fail
  - “Fail secure” or “fail safe”: abort operation when uncertain where the request or inputs come from
  - “Trusted computing” really means “trustworthy computing”

Trusted Computing Group (TCG)

- Historically, PC manufacturers have chosen flexibility over security
  - User can modify the PC in any way she likes
  - PC does not have hardware protection for boot procedure, does not validate O/S before loading it, does not validate other programs

- Today this is changing with efforts like the Trusted Computing Group (TCG)
  - [www.trustedcomputinggroup.org](http://www.trustedcomputinggroup.org)
  - Alliance formed in Jan 1999 by Compaq, HP, IBM, Intel & Microsoft
  - More than 150 companies by 2002
  - Developing a standard for a “**trusted platform**” (TP)
  - Scope of specs is at hardware, O/S and BIOS levels
    - Main spec released in Aug 2000 (v1.0) and Feb 2001 (v1.1)
    - PC-specific spec released in Sep 2001
Example: TCG

- Some goals of TP
  - Enable local and remote users to obtain reliable information about the software running on the platform
  - Provide a basis for secure key storage
  - Enable conditional release of secret information to the TP based on the software running

- TP enabled by a “trusted processing module” (TPM)
  - A hardware processing component that is isolated from software attacks and at least partially resistant to hardware tampering

- Each TPM is equipped with a different private key \( K_{TPM}^{-1} \)
  - And a certificate signed by the manufacturer (more on this later)

TCG “Roots of Trust”

The standard specifies two logical “roots of trust”

- **Root of trust for measurement (RTM)**: A platform-dependent component that starts *measurement* of software running
  - Measurement means the platform state, e.g. the applications started & configuration files used
  - In practice, this means that the RTM must run first (or everything that is run before it is trusted)
    - e.g., BIOS boot block, called the “core root of trust for measurement” (CRTM)

- **Root of trust for reporting (RTR)**: A platform-independent component that stores measurements as they happen, in such a way that measurements cannot be *undone*
  - RTR is implemented by the TPM
TPM Platform Configuration Registers

- TPM (version 1.1) contains sixteen 20-byte **platform configuration registers** (PCRs)
  - 20 bytes in order to store a SHA-1 hash value
- Each PCR records the last in a chain of hashes of the software that has been loaded and executed

![Diagram of TPM Platform Configuration Registers]

- PCR is updated before newly loaded software gets control
  - This is called extending the PCR
- PCR cannot be erased except by reboot (or SKINIT / GETSEC[SENTER] instruction in v1.2 TPMs)
- In this way, PCR contains record of software running

TCG Authenticated Boot

![Diagram of TCG Authenticated Boot]

- OS components
TCG Secure Boot

- Non-volatile “data integrity registers” (DIRs) are loaded with expected PCR values
  - DIRs are contained within TPM and require owner authorization to write

- If a PCR value, when computed, doesn’t match corresponding DIR value, then boot is canceled

TCG Integrity Challenge and Response

- Remote machine can query TPM for contents of PCRs

- TPM responds with signed PCR values
  - Signed with Attestation Identification Key (AIK) $K^{-1}_{AIK}$
  - The AIK pair is generated by the TPM
    - The AIK is endorsed by a certification authority

- TP additionally responds with records (hints) of what is “summarized” in the PCR values
  - Records could contain software itself, but more likely contains name, supplier, version, and URL for software
  - Enables remote machine to reconstruct and check PCR values
  - Records not trusted and so are stored outside TPM
TPM Sealed Storage

• Data encrypted with RSA key stored in the TPM
  – Slow public-key crypto

• **TPM seal**: bind data to current software state
  – Content of PCRs is encrypted along with the data

• **TPM unseal**: decrypt the sealed data
  – TPM will decrypt the data only if the PCRs contain the same values
  – Ensures that encrypted data can be trusted

• Applications:
  – Full Disk Encryption (FDE)
  – Digital Rights Management (DRM)

Full Disk Encryption: BitLocker Architecture

• **Static root of trust**: measurements of early boot components
Disk Layout and Key Storage

**Operating System Volume**

Contains
- Encrypted OS
- Encrypted page file
- Encrypted temp files
- Encrypted data
- Encrypted hibernation file

**Where’s the Encryption Key?**

1. SRK (Storage Root Key) contained in TPM
2. SRK encrypts the VMK (Volume Master Key)
3. VMK encrypts FVEK (Full Volume Encryption Key) – used for the actual data encryption
4. FVEK and VMK are stored encrypted on the Operating System Volume

**Attacks Against Software FDE**

[Halderman et al., 2008]

- Cold boot attack on secrets in memory
- DRAM does not immediately lose its value upon power-off
- This can be used to extract BitLocker keys if done skillfully
After 5 seconds

Source: Lest We Remember: Cold Boot Attacks on Encryption Keys. J. Alex Halderman et al. 2008 USENIX Security Symposium

---

After 30 seconds
After 60 seconds

After 5 minutes
Late Launch (TPM v. 1.2)

- **Static root of trust**: must reboot to restart measurement
- **Dynamic root of trust** (late launch): properties similar to reboot
  - Without a reboot!
  - Removes many things from TCB
    - BIOS, boot loader, DMA-enabled devices, ...
    - Long-running OS and Apps if done right

- Modern CPUs provide special instructions that interact with TPM chip
  - AMD SVM: **SKINIT** instruction
  - Intel TXT: **GETSEC[SENTER]** instruction

- Initial intent: virtualization support
  - Virtual machine monitor (VMM) can be measured (MVMM)
  - Integrity of loaded code can be attested
  - Untrusted legacy OS can coexist with trusted software

Password Reuse

- People often use 1 password for 2+ websites
- Banking, social networking, file sharing, ...
Password Exposure

• Password provided to compromised web server

Password Verification

• What if...
  – A compromised OS cannot learn the password
  – Only essential code can access password
    • Decrypt SSL traffic
    • Salt and hash password
    • Compare with stored hash
    • Output MATCH or FAILURE
  – Can remotely verify this is so

• Requires strong system security
**TCB Reduction with Flicker**

[McCune et al., 2008]

Today, TCB for sensitive code S:
- Includes App
- Includes OS
- Includes other Apps
- Includes hardware

With Flicker, S’s TCB:
- Includes Shim
- Includes some hardware

**What is S?**

- Self-contained code in an application
- Data secrecy and integrity requirements
- General-purpose computing
- Some examples
  - Manages a private key for web server or CA
  - Manages Access Control List (ACL)
  - Is a compute client in distributed setting
  - Manages a device for secure I/O
**Flicker Contributions**

- Isolate security-sensitive code execution from all other code and devices
- Attest to security-sensitive code and its arguments and nothing else
- Convince a remote party that security-sensitive code was protected
- Add < 250 LoC to the software TCB

**Adversary Capabilities**

- Run arbitrary code with maximum privileges
- Subvert any DMA-enabled device
  - E.g., network cards, USB devices, hard drives
- Perform limited hardware attacks
  - E.g., power cycle the machine
  - Excludes physically monitoring/modifying CPU-to-RAM communication
**Architecture Overview**

- **Core technique**
  - Pause current execution environment
  - Execute security-sensitive code with hardware-enforced isolation
  - Resume previous execution

- **Extensions**
  - Preserve state securely across invocations
  - Attest only to code execution and protection
  - Establish secure communication with remote parties

---

**Execution Flow**

![Execution Flow Diagram](chart.png)

- **OS**
- **CPU**
- **RAM**
- **Module**
- **App**
- **SKINIT**
- **TP**
- **PCRs**

**Inputs**

**Outputs**

STOP 0 0 0

PCR: 0 0 0 ...
Attestation

TPM

PCRs: 0 0 0

K,1

Shim S Inputs

Outputs

Attestation

What code are you running?

0

versus

App 1

App S

Sign(Os, K,1)
Context Switch with Sealed Storage
- Seal data under combination of code, inputs, outputs
- Data unavailable to other code

Functionality
- Shim can execute arbitrary x86 code but provides very limited functionality
  - No system calls, device drivers, etc.
  - Must exit back to OS and properly apply crypto to keep secrets safe

- Fortunately, many security-sensitive functions do not require much
  - E.g., key generation, encryption/decryption, password check

- Functionality can be added to support a particular security-sensitive operation
Application: Rootkit Detector

- Administrator can check the integrity of remote hosts
  - E.g., only allow uncompromised laptops to connect to the corporate VPN

Sources
- Various slides from Mike Reiter and Jonathan McCune
Review of Lecture

• What did we learn?
  – Trusted Platform Module (TPM)
  – Remote attestation
  – Sealed storage
  – Minimizing the TCB with Flicker

• What’s next?
  – Passwords

• Deadline reminders
  – Post pilot project reports by end of today
  – Sign up for paper presentations by EOD Wednesday
  – Submit first homework (paper review) by 6 pm Tuesday