Trusted Computing

Tom Chothia
Computer Security

Introduction

- Trusted Computing
- The Trusted Platform Module (TPM)
- Case study: the Sony PlayStation 3

The Idea

- Through out this module, I have shown you that:
  - If attackers have physical access to the hardware they own it.
  - We cannot be sure what software a remote party is running.
  - Once you distribute software you loose control of it.

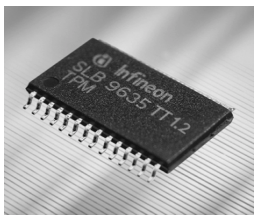
Wouldn't it be "nice" if this wasn't the case?

Basic Remote Attacks

- It's almost impossible to read a value from a chip.
  - Can take it apart and learn the circuit,
  - but not the data
- "Trusted computing" tries to use this to let hardware providers control a computer, not the user.

The Trusted Platform Module

- TPM is in most good laptops
- The TPM allows:
  - Secure storage,
  - Platform integrity reporting,
  - Platform authentication,
- Remote parties can establish a secure connection (which you can't observe).



The Trusted Module Platform

Trusted computing can in theory:

- Give you a remote guarantee of the identity of a machine.
- Tell you what software a remote machine is running.
- Provide secure tamper proof storage.

Key TPM functions

- Can create keys, and use them to decrypt and encrypt.
  - TPM can create RSA keys,
  - Keys stored securely on the TPM,
  - Protected by “auth data” (like password)
  - Too slow to guess the key by brute force.
- Removes the issue of finding the key in memory.
- Stops hard disk being removed from computer.

Key TPM functions

- Stores a unique endorsement key (EK),
  - Signed by the manufacturer
  - Keys never leave the TPM chip
- Attestation identity key pairs (AIKs)
  - Bound to the EK by a proof.
  - They let TPM prove its identity.
  - Anyone can encrypt data just for this TPM.

Uses for AIKs?

The PCR Registers

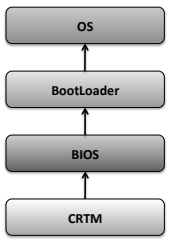
- 16+ Platform Configuration Registers (PCR).
- These registers can be extended by being hashed with a value.
  - New register value = SHA1(old value | input).
- These store a “one way value”.
- N.B. a remote party can use a TPM session to extend a value into a register.

Platform Integrity

1 of the PCRs is used to identify the system.

Systems with TPMs have a non-writable part of the BIOS called: *Core Root of Trust for Measurement* (CRTM).

This hashes the BIOS code and extends a PCR with that value.



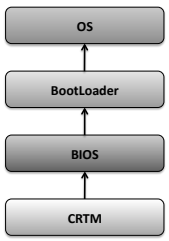
Platform Integrity

The BIOS hashes the Boot Loader and extends the PCR with this.

Boot Loader hashes the OS and extends the PCR again.

The PCR can be checked by a remote party.

The PCR value tells the remote party exactly what software is installed.



Uses for Platform Integrity Checking?

Ethical Issues

- I run someone else's software on my computer, who owns it?
- Trusted computing removes control from the computer's owner and gives it to the creator.
- Often this takes control from the user and gives it to big companies that want to make money from them.

Case Study: PlayStation 3

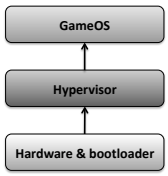
- PS3 use powerful "Cell Broadband Engine Architecture
  - 8 CPUs
- These are worth more than you pay for the console.
  - Sony makes money off the games
- Sony wants to stop piracy and stop people using the hardware for other purposes.

Platform Integrity

The hardware and bootloader loads and runs a "Hypervisor".

This must be signed by Sony, and the bootloader checks this signature.

The Hypervisor then loads and runs "GameOS", also signed by Sony.

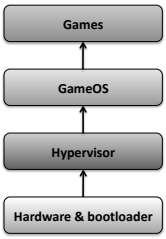


Platform Integrity

The GameOS provides the operating system for games.

The GameOS will only load games into memory and run them if they are signed by Sony.

This is an example of a "Chain of Trust", just the same as the TPM.



PlayStation 3 Security

- Keys are securely stored in hardware.
  - Sony's signature verification key.
  - Unique key for each console.
- Only Sony OS can be run.
- PS3 uses full disk encryption
  - Can't read data directly off the disk

PlayStation 3 Security

- On paper this looks like great security.
  - To break it you would have to rebuild almost all of the PlayStation hardware and software.
- From a risk analysis point of view, Sony have done their job well.
  - They are ISO 27001 certified.

PlayStation 3 Security

- On paper this looks like great security.
  - To break it you would have to rebuild almost all of the PlayStation hardware and software.
- From a risk analysis point of view, Sony have done their job well.
  - They are ISO 27001 certified.
- BUT … they wrote their own crypto.

Disk Encryption

- Symmetric Key (can't read it because Hypervisor won't let us).
- User can access files they made themselves not other files (e.g. the games).
- Sony doesn't use cipher block chaining
  - no hashing or integrity check!
- Result: you can decrypt anything you like!

Disk Encryption

1. Make a copy of the disk.
2. Write a file to the disk.
3. Find the bit of disk that changed (that's your encrypted file).
4. Over write this with what you want to decrypt.
5. Restart the PlayStation and ask for your file back.
6. PlayStation decrypts the file and gives you the plain text.

Jailbreak: Running our own code

Best way to run arbitrary code on a secure system?

Buffer overflow attack!

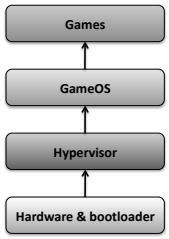
Unnamed Hackers found a buffer overflow in the USB handlers.
- PS3 trusts USB stick to tell it the size of the data.

Piracy

- This can be used for piracy.
- Put image of game on USB stick as payload (disk encryption broken).
- Add code to USB stick to perform attack and start the game.
- See websites.

But we want to run Linux!

- Nothing so far interferes with the Hypervisor or GameOS.
- Fail0verflow group found a flaw in the chain of trust.



Replacing GameOS

- The Hypervisor loads the OS, but it will only load an OS signed by Sony.
- Sony use elliptic curve encryption for their signatures.
- Like RSA, but more efficient.

Elliptic curve signatures.

- To sign program p, you need:
  - Elliptic curve parameters G
  - m : random number (different every time)
  - k : signing key
- Signatures of p is pair (R,S):

  $R = (mG)_x$, $S = (\#(p) + k.R)/m$

Epic Fail

Sony use the same random numbers each time:

$R = (mG)_x$, $S_1 = (\#(p1) + k.R)/m$

$R = (mG)_x$, $S_2 = (\#(p2) + k.R)/m$

Then $S_1 - S_2 = (\#(p1) - \#(p2)) / m$

$\Rightarrow$ $m = (\#(p1) - \#(p2)) / S_1 - S_2$

$\Rightarrow (m.S_1 - \#(p1)) / R = k$

xkcd

```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

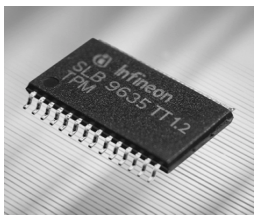
`java.util.Random` isn't good enough for encryption you need to use: `java.security.SecureRandom`

Running Linux

- With the signing key "k" Fail0verflow signed a Linux kernel that can then be loaded and run on a PS3.
- About 10 months ago, George Hotz, released code and the key.
  - Sony sued.

Recommended Reading

- "Introduction to the TPM" by Allan Tomlinson.
- Mark Ryan's notes on Trusted Computing
- Both linked to from website.

Conclusion

- TPM is in most good laptops
- TPM provides:
  - Secure storage,
  - Platform integrity reporting,
  - Platform authentication,
- Remote parties can establish a secure connection (which you can't observe).