Security: Xbox

Emmett Witchel

CS380L

Xbox faux quiz

- How many security mistakes did the Xbox designers make?
- What links are in the Xbox chain of trust that ensures a valid OS boots and only MS-approved applications can run?
- What is attestation?
- Why is the Xbox initial OS in ROM, and not in flash? Why are there two bootloaders?
- What is control flow integrity? How can it be violated? How can it be enforced?
- Why can’t Xbox store the 2bl and OS in plaintext?
- Why is memory “unstable” during early boot of Xbox? What impact does this have on crypto?
- Linux hackers recovered RC4 key, could encrypt Linux bootloader as 2bl: why didn’t they?
- Why is the special purpose VM + xcode purported to be secure?

Security

What does it mean for a computer system to be secure?

Security

What does it mean for a computer system to be secure?

All of these properties are security properties

- P – Privacy
- C – Confidentiality
- A – Authentication
- A – Authorization
- A – Availability
- I – integrity
- N – non-repudiation

Two security ideas

- Privacy – Sensitive data is not read
  - My health and financial records should remain private
  - Do I want Facebook to have my family photos?
  - Data breaches are privacy/confidentiality problems
- Integrity – Sensitive data is not written
  - Taking over computer system is an integrity attack
  - Then steal data, or steal money, or send email as you
  - What devices/software do you trust? with what data?

- Privacy gets more press, but distinguish privacy from integrity

Discretionary & Mandatory Access Control

- Discretionary access control (DAC)
  - Access control policy controlled by user
  - E.g., Linux
  - Usability a plus, but always getting broken
- Mandatory access control (MAC)
  - Access control policy controlled by administrator
  - E.g., SELinux, AppArmor
  - Difficult to configure, brittle, but more secure

- Trend toward usable MAC

Why is computer security hard?

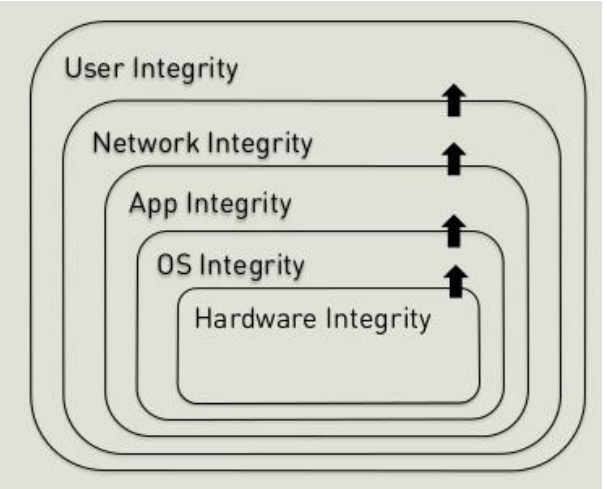
- Because managing secrets and trust is hard
  - Who keeps your secrets?
  - Who do you trust for your health care decisions?
  - Who can debit from your bank account?
- Trust benefits from societal and legal support
  - Can't testify against your spouse
- Computers make it easier to exploit
  - Millions of users' privacy from one event

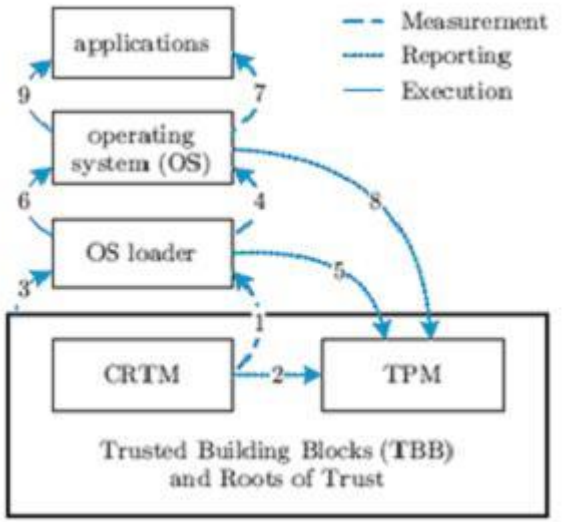
- Authentication story

Chain of Trust

Use hardware and cryptography to ensure:

- system starts in a known, good state
- proceeds to load uncompromised software
- ***Attests*** to the start state of software
  - often not enough to ensure security during operation, but it is a start.





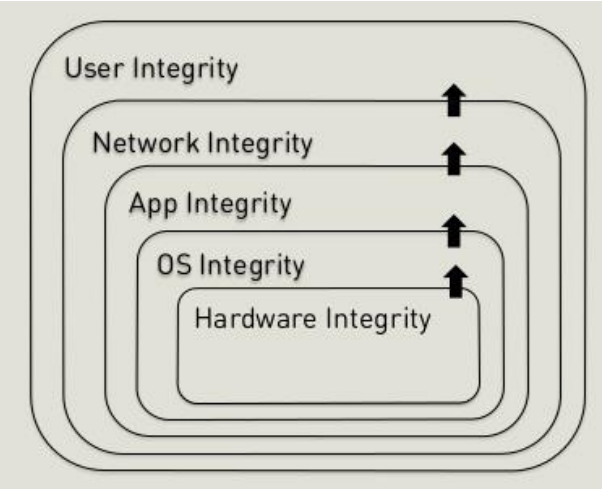
Microsoft obviously made a lot of mistakes. But it would be too easy to just attribute all these to stupid engineers. There have been good (and different) reasons for most of these mistakes, and one can learn a lot from them.

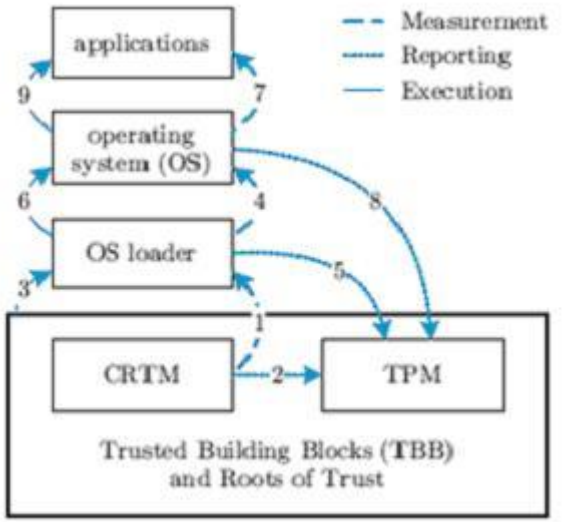
--Michael Steil

Chain of Trust

Use hardware and cryptography to ensure:

- system starts in a known, good state
- proceeds to load uncompromised software
- ***Attests*** to the start state of software
  - often not enough to ensure security during operation, but it is a start.





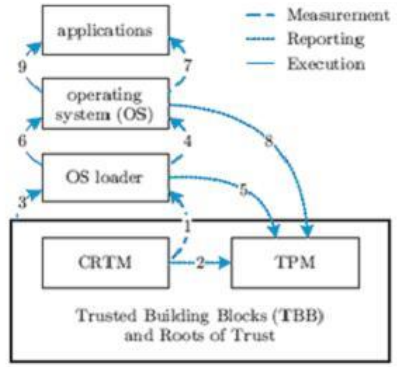
Why isn't attestation "enough"?

too
ve
s,

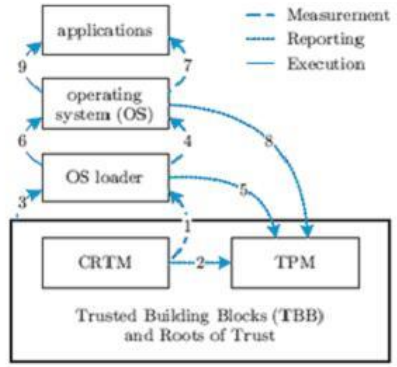
and one can learn a lot from them.
--Michael Steil

Chain of Trust in Xbox

- Want initial OS in ROM (why?)
  - But execution starts in flash, flash size >> ROM size



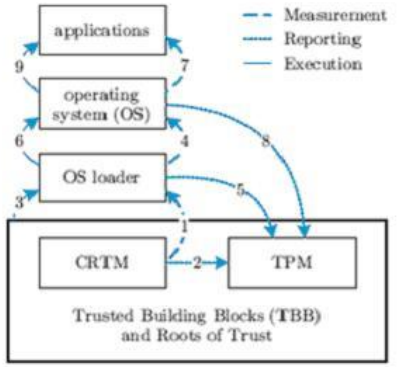
Chain of Trust in Xbox

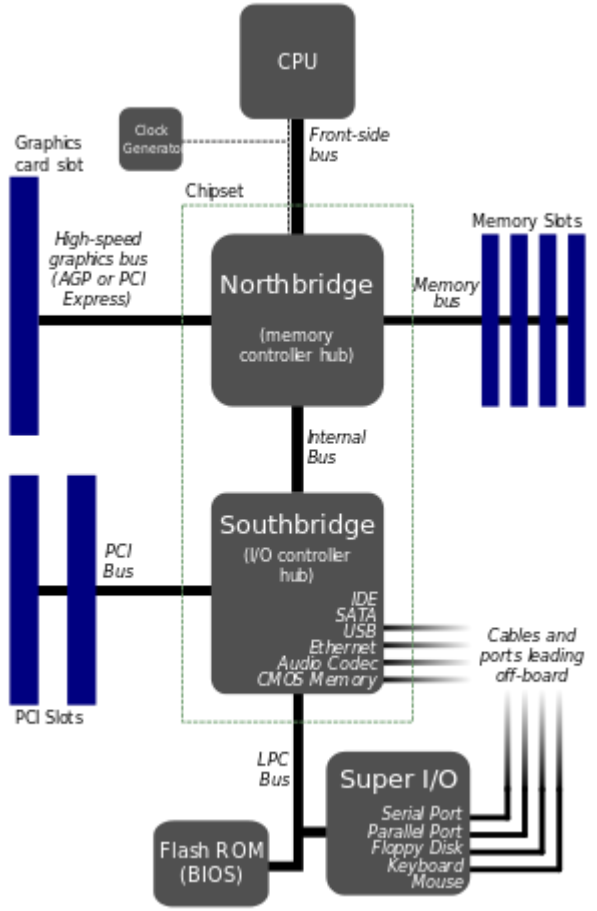


- Want initial OS in ROM (why?)
  - But execution starts in flash, flash size >> ROM size
- ROM program loads and verifies OS in flash
  - But…where to put the ROM?
  - in Southbridge (huh?)

Chain of Trust in Xbox

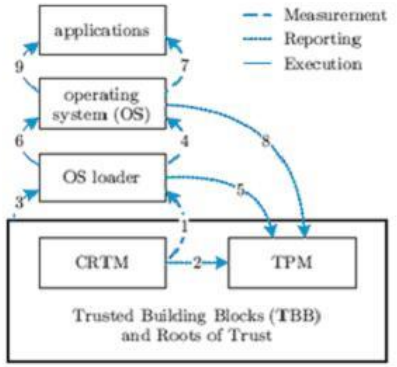
- Want initial OS in ROM (why?)
  - But execution starts in flash, flash size >> ROM size
- ROM program loads and verifies OS in flash
  - But…where to put the ROM?
  - in Southbridge (huh?)

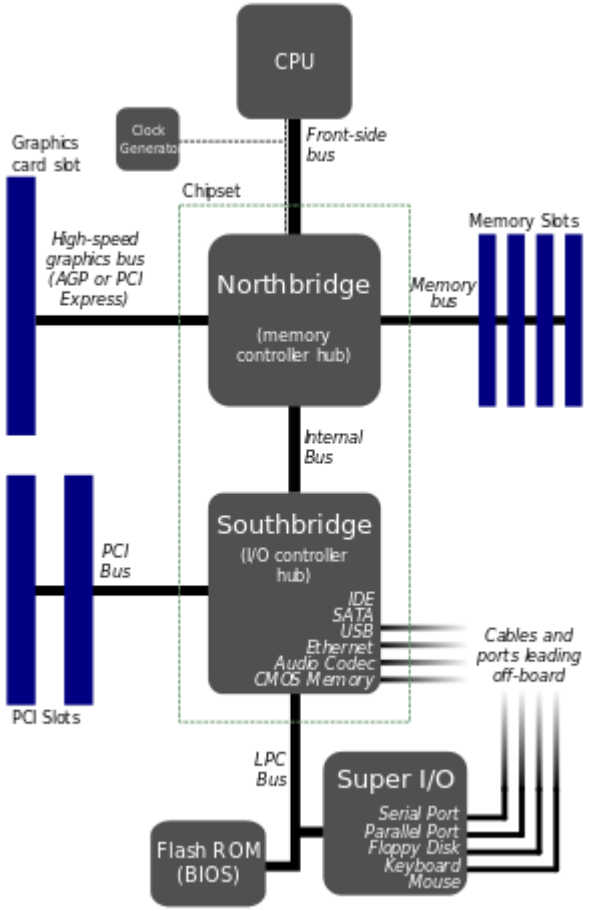




Chain of Trust in Xbox

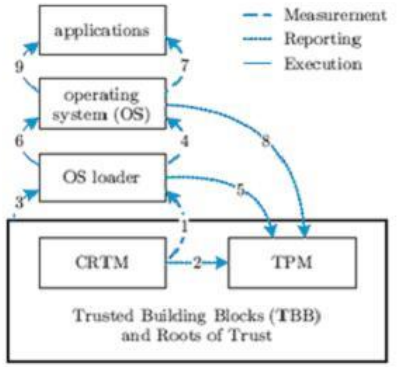
- Want initial OS in ROM (why?)
  - But execution starts in flash, flash size >> ROM size
- ROM program loads and verifies OS in flash
  - But…where to put the ROM?
  - in Southbridge (huh?)
- ROM check: H(flash memory) == 0xFFBBC46...34 →
  - Hash can’t change, so OS can’t be updated
  - So introduce a second boot loader (2bl)

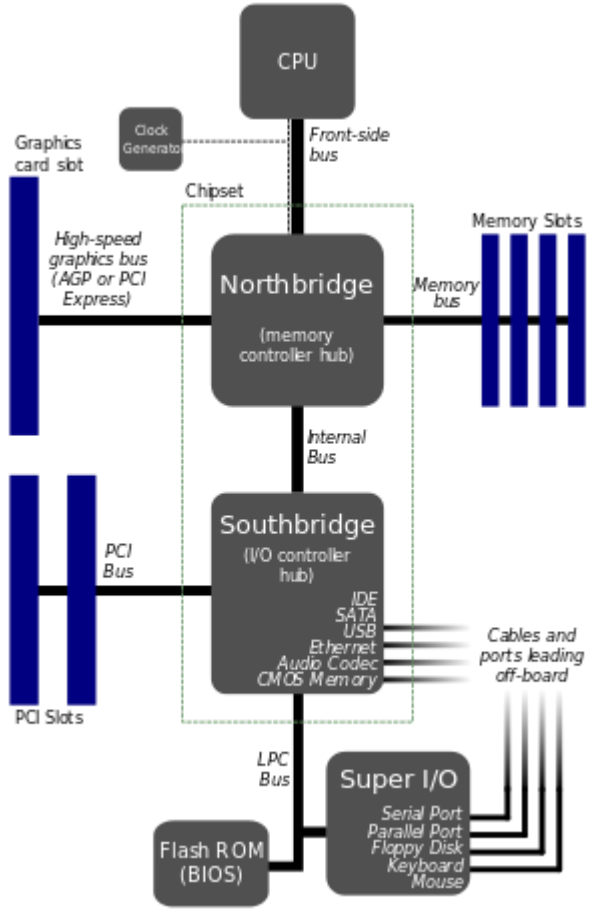




Chain of Trust in Xbox

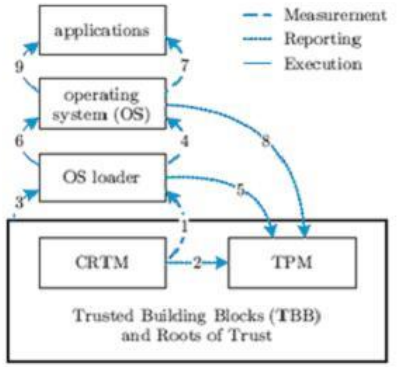
- Want initial OS in ROM (why?)
  - But execution starts in flash, flash size >> ROM size
- ROM program loads and verifies OS in flash
  - But…where to put the ROM?
  - in Southbridge (huh?)
- ROM check: H(flash memory) == 0xFFBBC46...34 →
  - Hash can’t change, so OS can’t be updated
  - So introduce a second boot loader (2bl)
- CPU executes ROM,

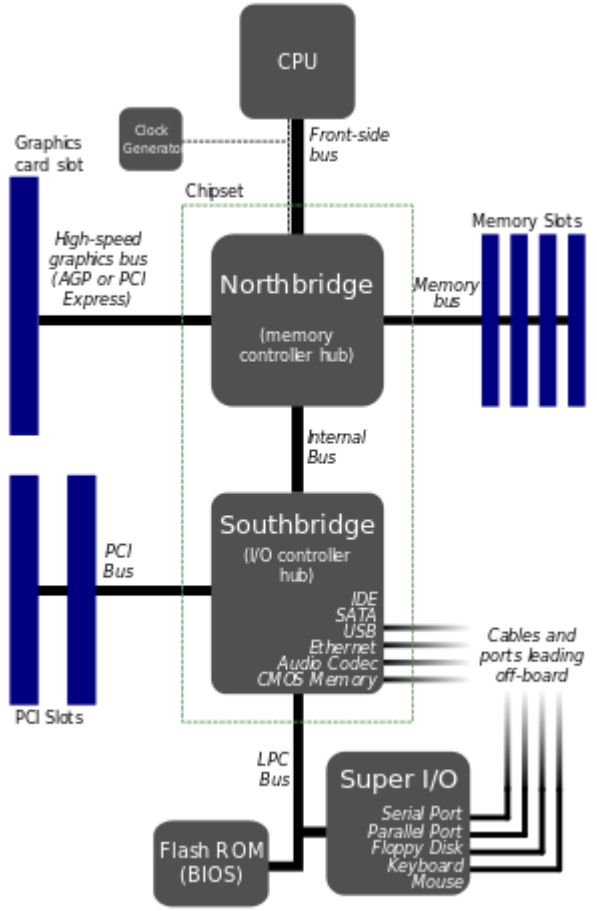




Chain of Trust in Xbox

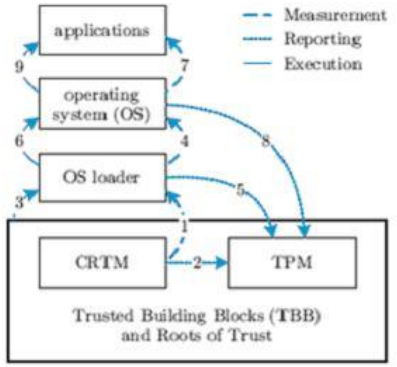
- Want initial OS in ROM (why?)
  - But execution starts in flash, flash size >> ROM size
- ROM program loads and verifies OS in flash
  - But…where to put the ROM?
  - in Southbridge (huh?)
- ROM check: H(flash memory) == 0xFFBBC46...34 →
  - Hash can’t change, so OS can’t be updated
  - So introduce a second boot loader (2bl)
- CPU executes ROM,
- ROM check: H(2bl (in flash)) == 0xFFBBC46...34,

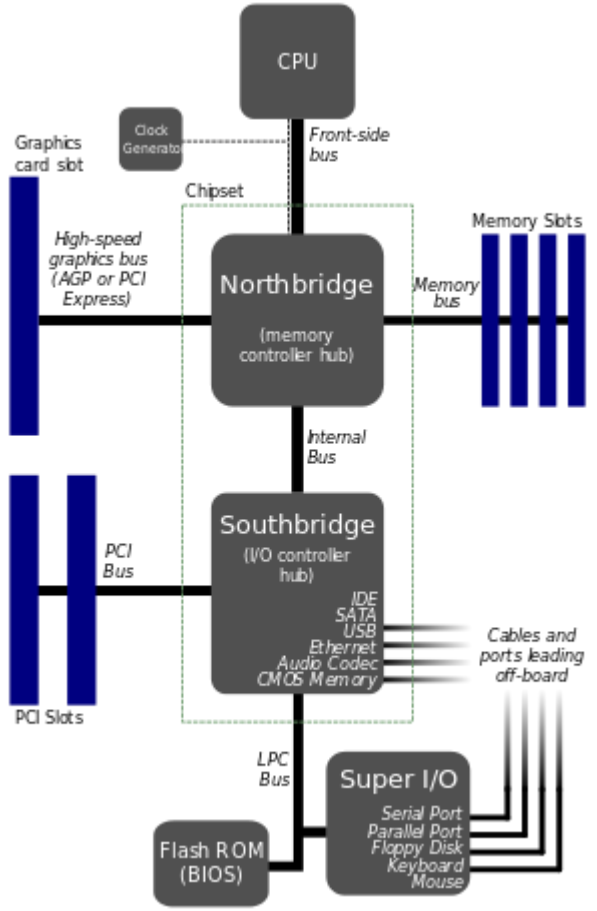




Chain of Trust in Xbox

- Want initial OS in ROM (why?)
  - But execution starts in flash, flash size >> ROM size
- ROM program loads and verifies OS in flash
  - But…where to put the ROM?
  - in Southbridge (huh?)
- ROM check: H(flash memory) == 0xFFBBC46...34 →
  - Hash can’t change, so OS can’t be updated
  - So introduce a second boot loader (2bl)
- CPU executes ROM,
- ROM check: H(2bl (in flash)) == 0xFFBBC46...34,
- 2bl check: Verify(OS (in flash), FS secret-key (in ROM)).

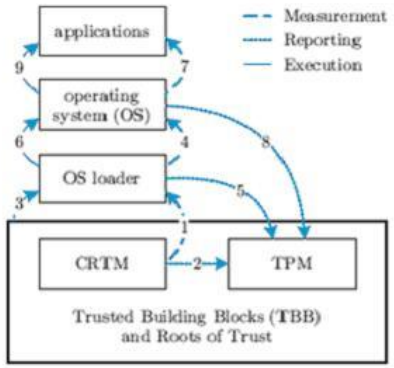


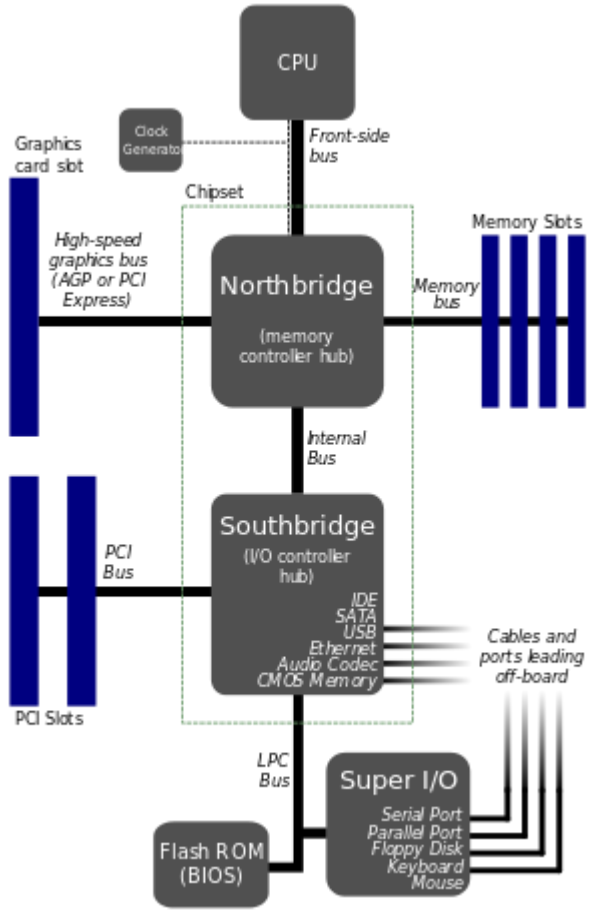


Chain of Trust in Xbox, cont.

BUT wait! (again…)

- Can 2bl and OS be in plain text?

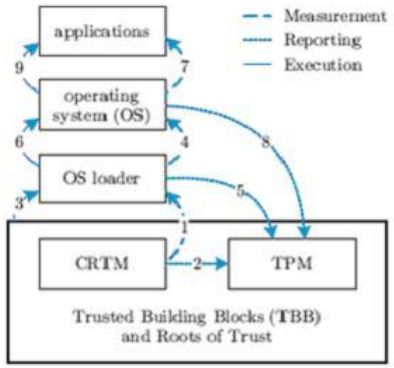


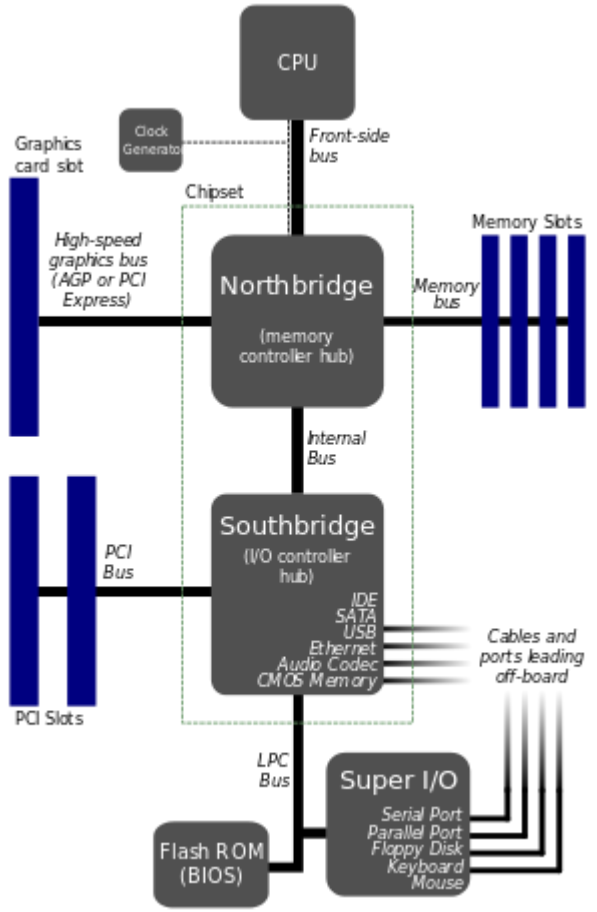


Chain of Trust in Xbox, cont.

BUT wait! (again…)

- Can 2bl and OS be in plain text?
- Can’t use crypto

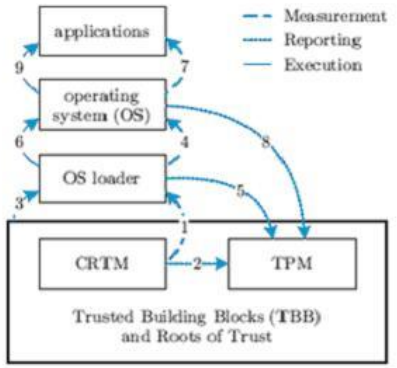


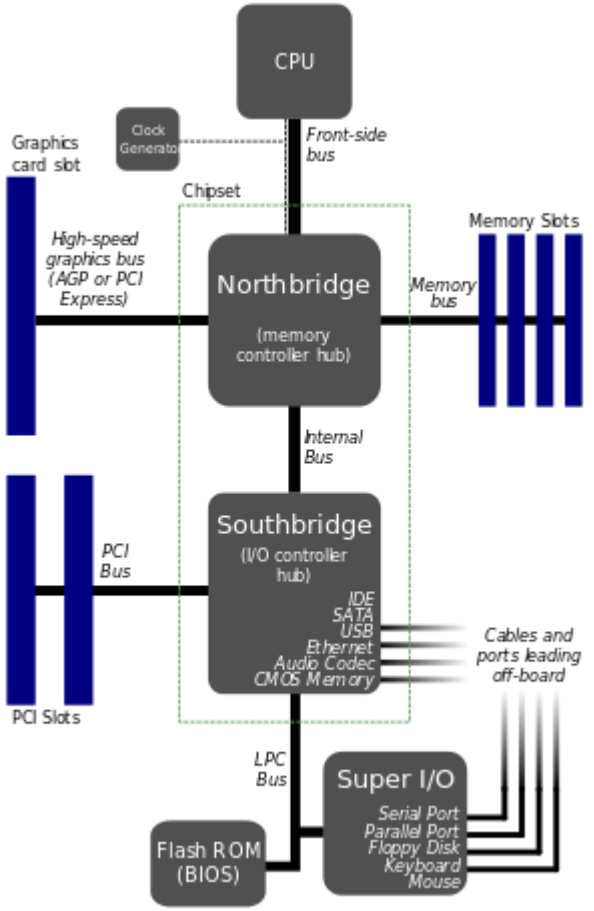


Chain of Trust in Xbox, cont.

BUT wait! (again…)

- Can 2bl and OS be in plain text?
- Can’t use crypto
  - RAM init + decrypt + hash needs 2KB, only 512B available

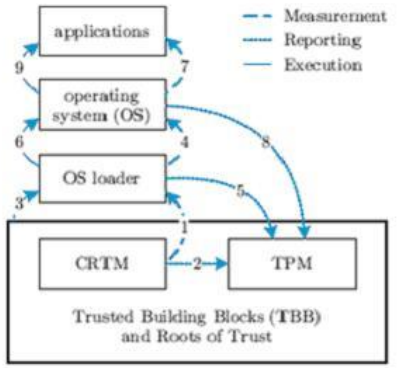


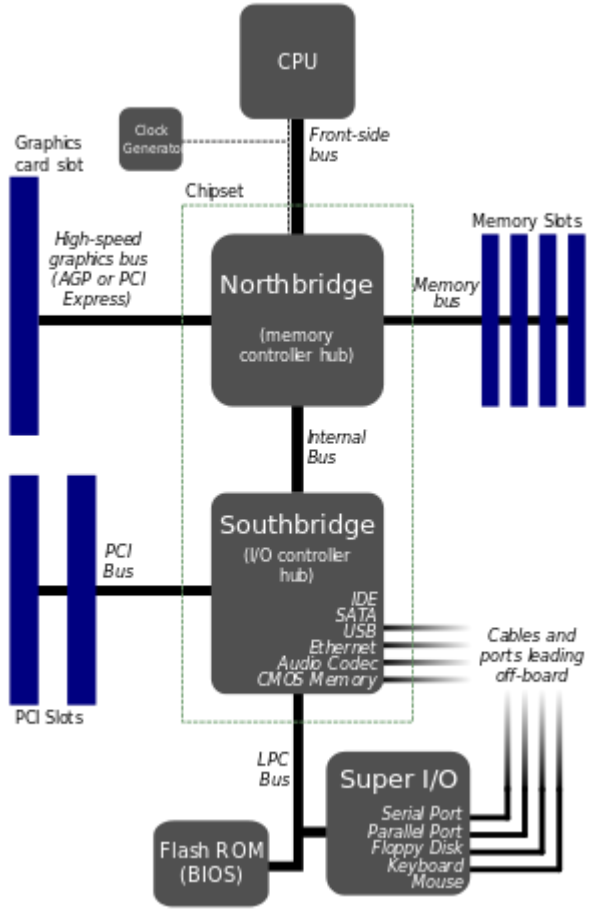


Chain of Trust in Xbox, cont.

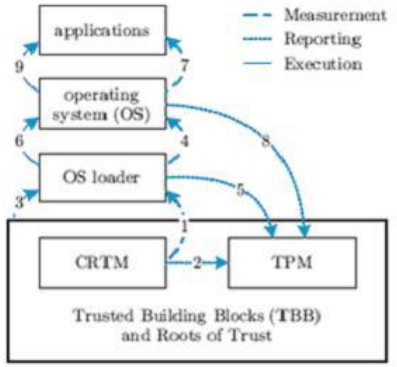
BUT wait! (again…)

- Can 2bl and OS be in plain text?
- Can’t use crypto
  - RAM init + decrypt + hash needs 2KB, only 512B available
- Solution: VM+xcode



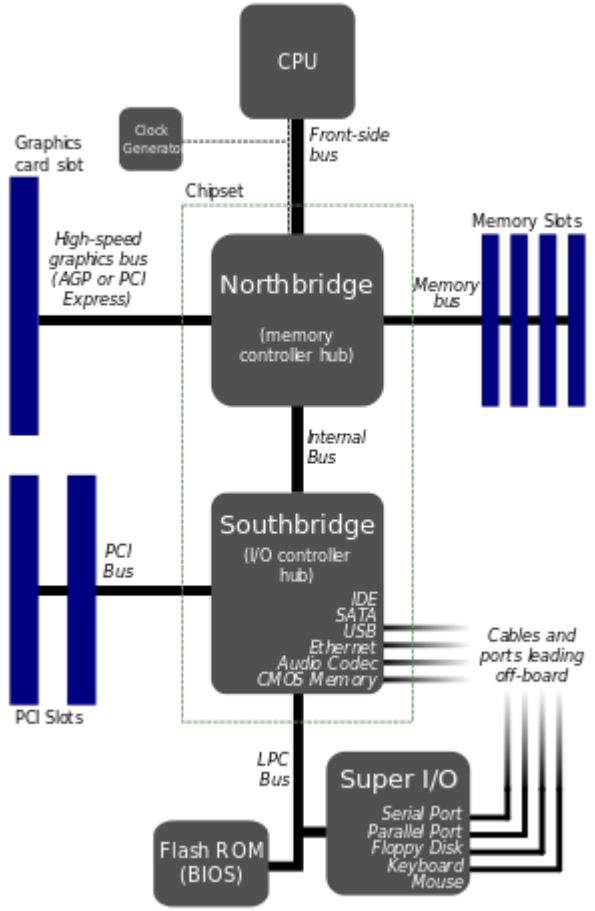


Chain of Trust in Xbox, cont.



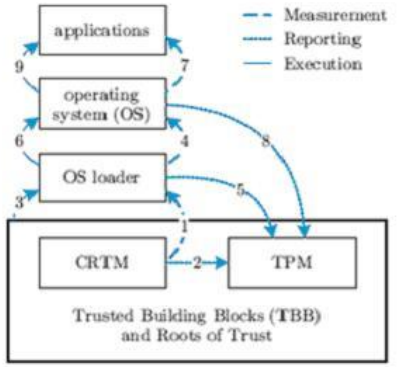
BUT wait! (again…)

- Can 2bl and OS be in plain text?
- Can’t use crypto
  - RAM init + decrypt + hash needs 2KB, only 512B available
- Solution: VM+xcode



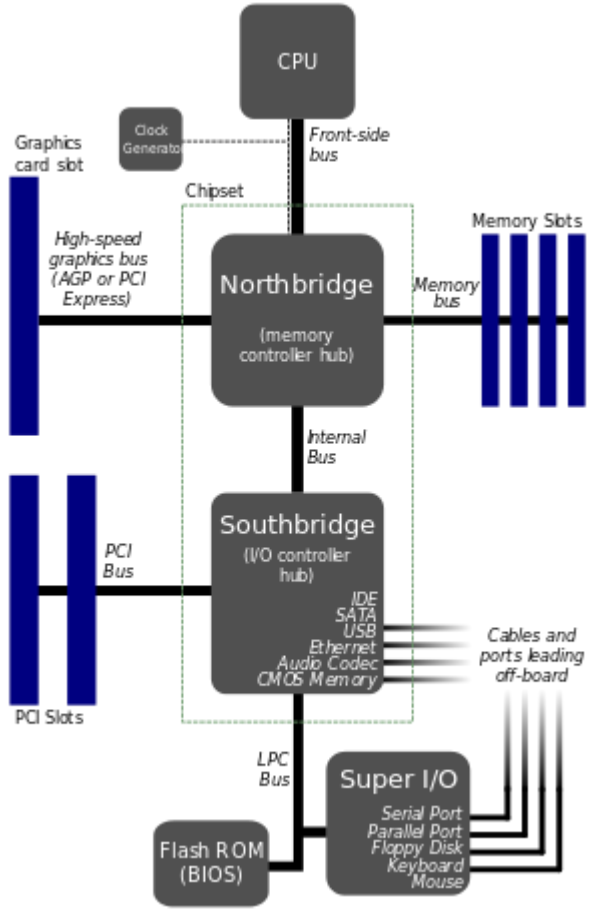
Why is VM+xcode secure?

Chain of Trust in Xbox, cont.



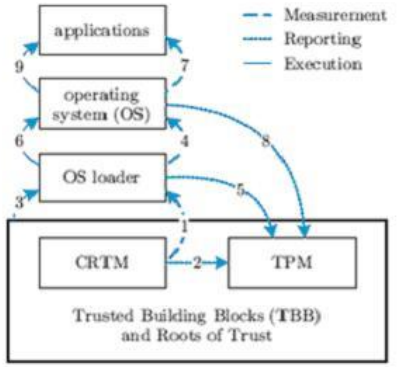
BUT wait! (again…)

- Can 2bl and OS be in plain text?
- Can't use crypto
  - RAM init + decrypt + hash needs 2KB, only 512B available
- Solution: VM+xcode
  - VM not encrypted (no space) or hashed (so can be updated)



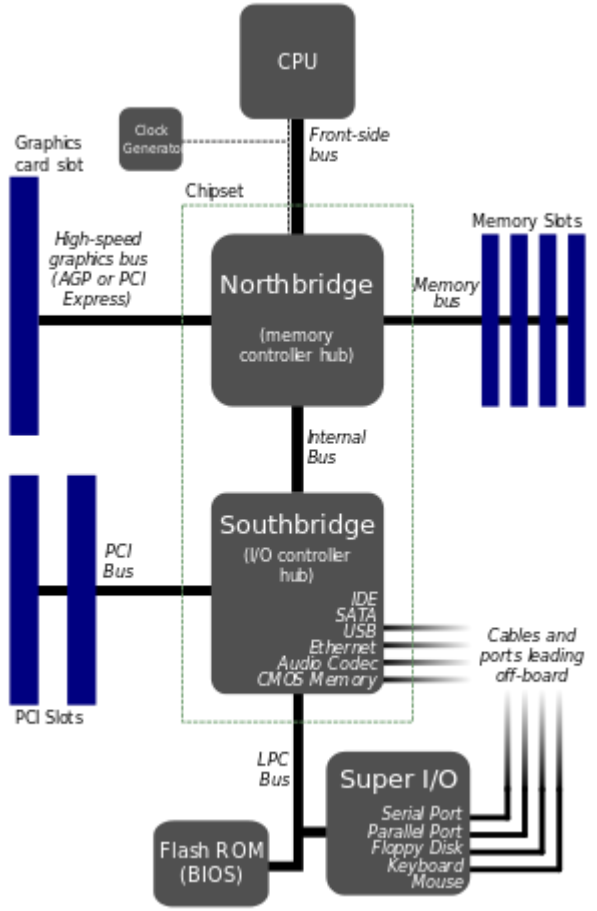
Why is VM+xcode secure?

Chain of Trust in Xbox, cont.



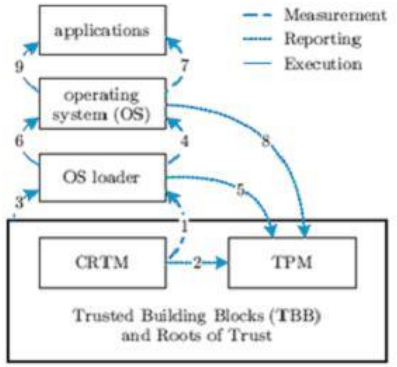
BUT wait! (again…)

- Can 2bl and OS be in plain text?
- Can’t use crypto
  - RAM init + decrypt + hash needs 2KB, only 512B available
- Solution: VM+xcode
  - VM not encrypted (no space) or hashed (so can be updated)
  - Limited instruction set



Why is VM+xcode secure?

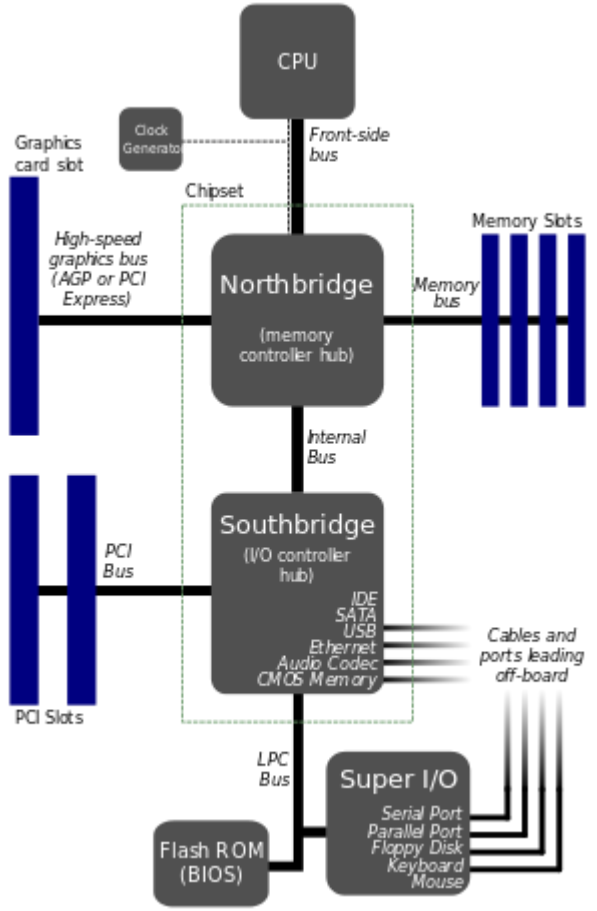
Chain of Trust in Xbox, cont.



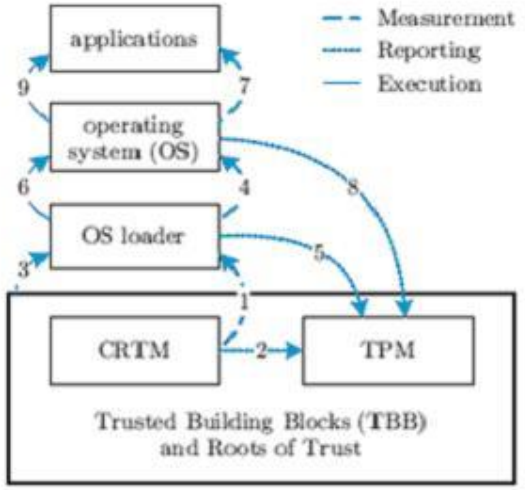
BUT wait! (again…)

- Can 2bl and OS be in plain text?
- Can’t use crypto
  - RAM init + decrypt + hash needs 2KB, only 512B available
- Solution: VM+xcode
  - VM not encrypted (no space) or hashed (so can be updated)
  - Limited instruction set
  - …therefore (ostensibly) incapable of malicious operations

Why is VM+xcode secure?

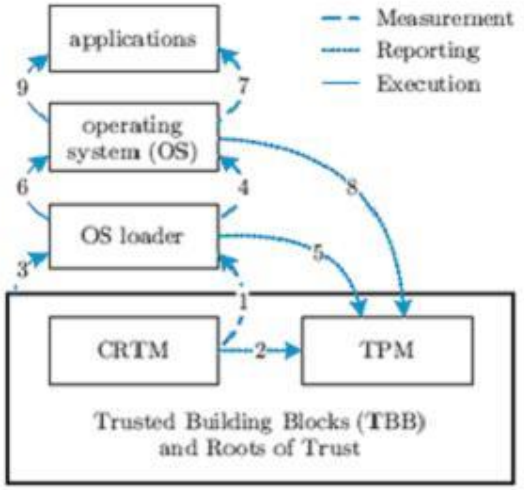


Confining Xcodes, Hide secret ROM



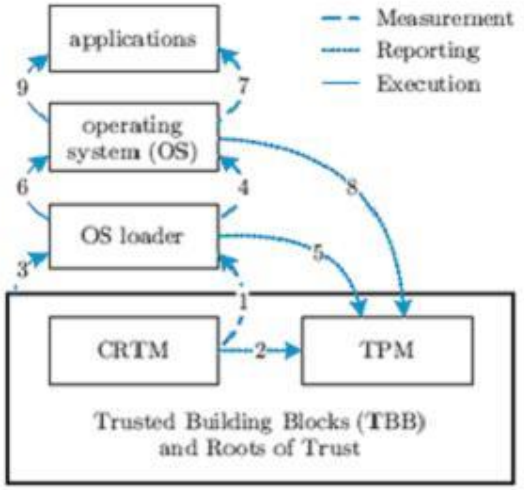
Confining Xcodes, Hide secret ROM

- Mask top 4 bits of every address,
  - xcodes cannot read upper 512b of address space



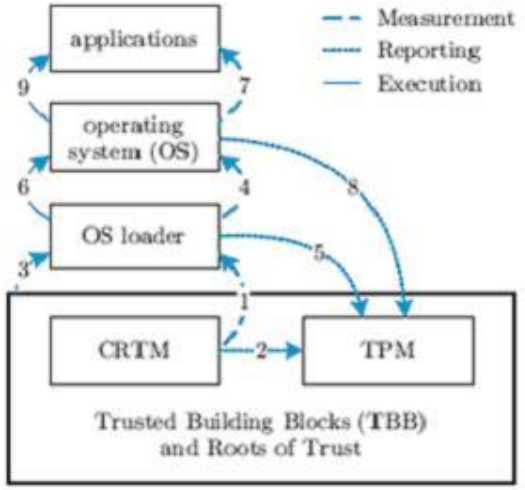
Confining Xcodes, Hide secret ROM

- Mask top 4 bits of every address,
  - xcodes cannot read upper 512b of address space
- Cannot allow xcodes to turn off secret ROM,
  - or interpreter “falls down” into untrusted flash memory.



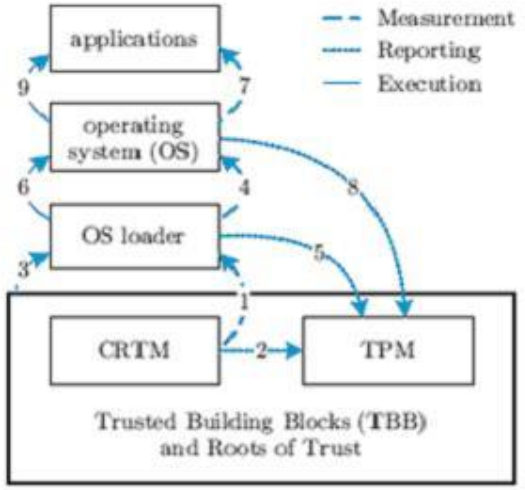
Confining Xcodes, Hide secret ROM

- Mask top 4 bits of every address,
  - xcodes cannot read upper 512b of address space
- Cannot allow xcodes to turn off secret ROM,
  - or interpreter “falls down” into untrusted flash memory.
- BUT, when 2bl starts, it **must** turn off the secret ROM,
  - otherwise compromised game could see it.
  - So clear bit in PCI config register every time it is written.



Confining Xcodes, Hide secret ROM

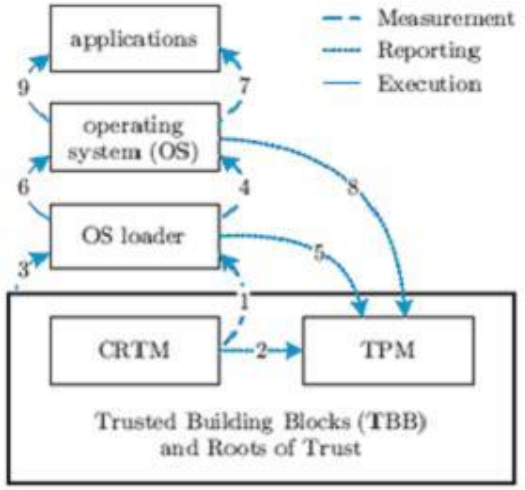
- Mask top 4 bits of every address,
  - xcodes cannot read upper 512b of address space
- Cannot allow xcodes to turn off secret ROM,
  - or interpreter “falls down” into untrusted flash memory.
- BUT, when 2bl starts, it **must** turn off the secret ROM,
  - otherwise compromised game could see it.
  - So clear bit in PCI config register every time it is written.
- Resulting design:
  - Execute secret ROM, set up segment descriptors.
  - Interpret untrusted xcodes (set up RAM)
  - Decrypt and hash 2bl, store it in RAM
  - If H(2bl) == 0x7854794A, execute 2bl, otherwise panic.
  - 2bl decrypts and verifies Windows kernel.
  - Windows kernel verifies media and checks RSA signature of games.



Confining Xcodes, Hide secret ROM

Why doesn't this version work?

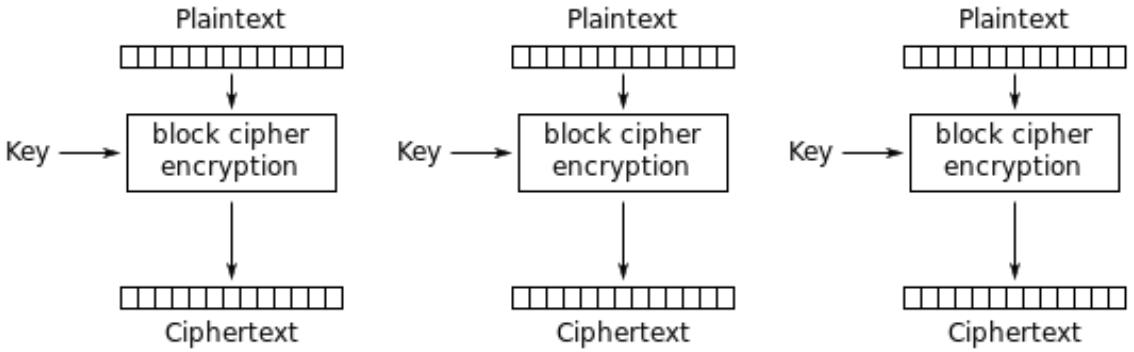
- Mask top 4 bits of every address,
  - xcodes cannot read upper 512b of address space
- Cannot allow xcodes to turn off secret ROM,
  - or interpreter "falls down" into untrusted flash memory.
- BUT, when 2bl starts, it **must** turn off the secret ROM,
  - otherwise compromised game could see it.
  - So clear bit in PCI config register every time it is written.
- Resulting design:
  - Execute secret ROM, set up segment descriptors.
  - Interpret untrusted xcodes (set up RAM)
  - Decrypt and hash 2bl, store it in RAM
  - If H(2bl) == 0x7854794A, execute 2bl, otherwise panic.
  - 2bl decrypts and verifies Windows kernel.
  - Windows kernel verifies media and checks RSA signature of games.



Block Cipher Crypto Background

Electronic Codebook Mode

- Fast, parallelizable
- insecure (Why)?

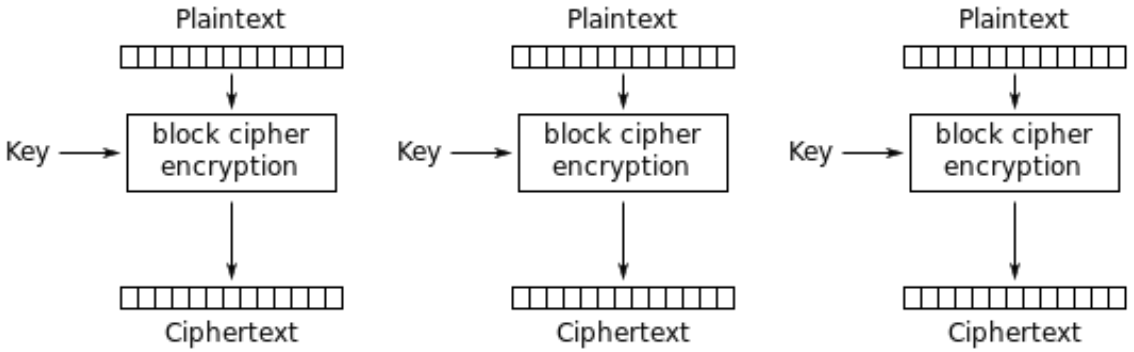


Electronic Codebook (ECB) mode encryption

Block Cipher Crypto Background

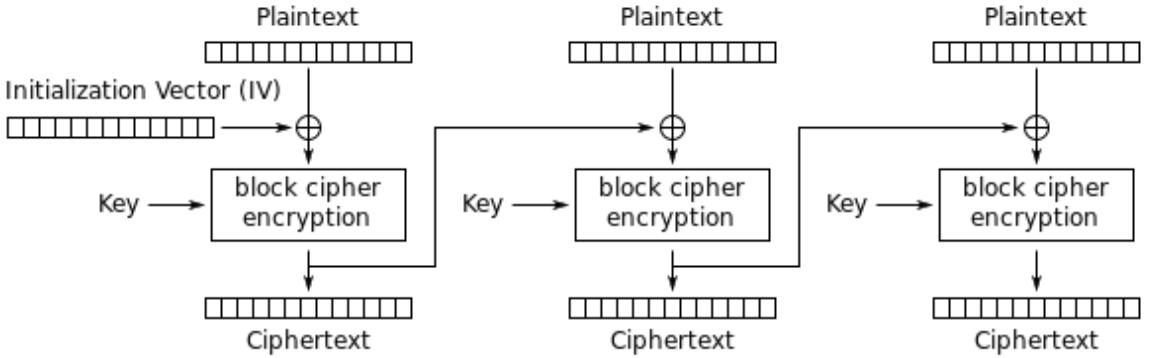
Electronic Codebook Mode

- Fast, parallelizable
- insecure (Why)?



Electronic Codebook (ECB) mode encryption

Cipher Block Chaining

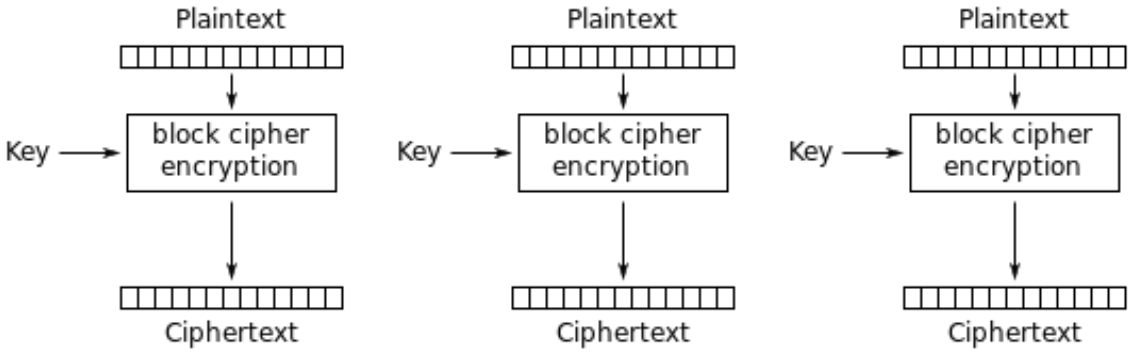


Cipher Block Chaining (CBC) mode encryption

Block Cipher Crypto Background

Electronic Codebook Mode

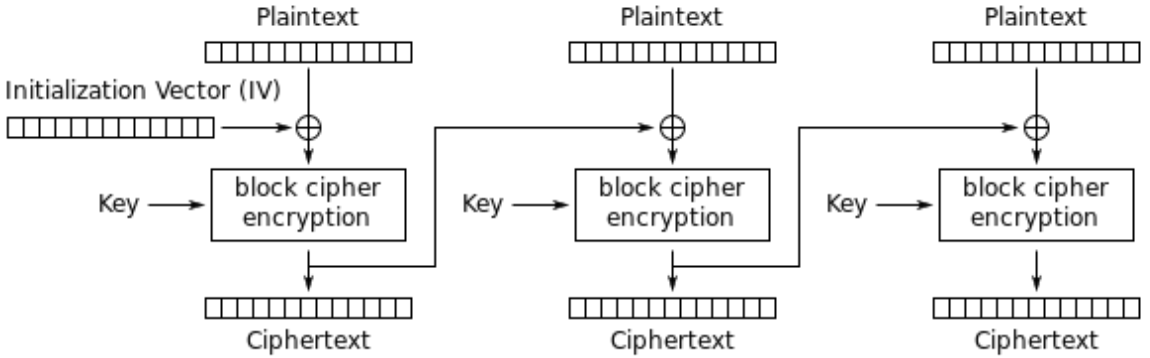
- Fast, parallelizable
- insecure (Why)?



Electronic Codebook (ECB) mode encryption

Cipher Block Chaining

- e.g. xor(enc(prev-block), cur-block)

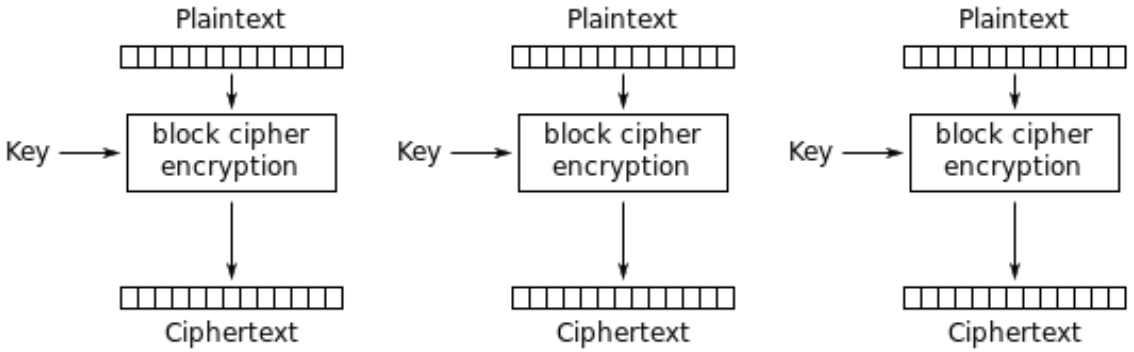


Cipher Block Chaining (CBC) mode encryption

Block Cipher Crypto Background

Electronic Codebook Mode

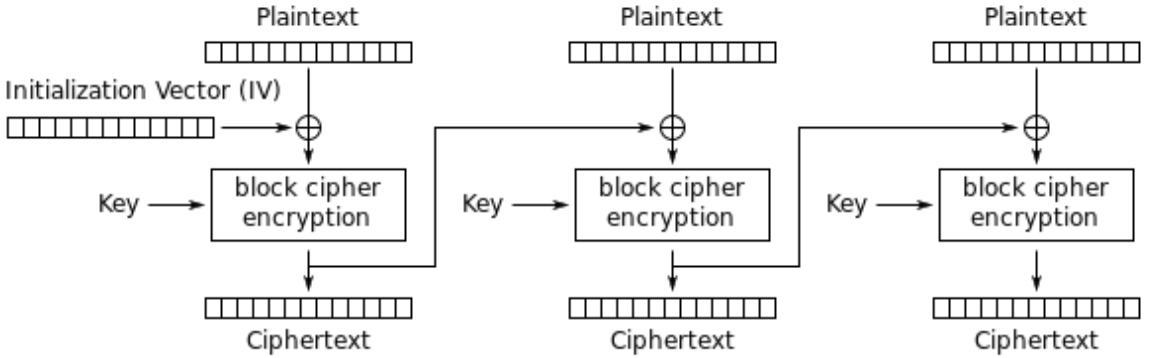
- Fast, parallelizable
- insecure (Why)?



Electronic Codebook (ECB) mode encryption

Cipher Block Chaining

- e.g. xor(enc(prev-block), cur-block)
- need initialization vector (IV).

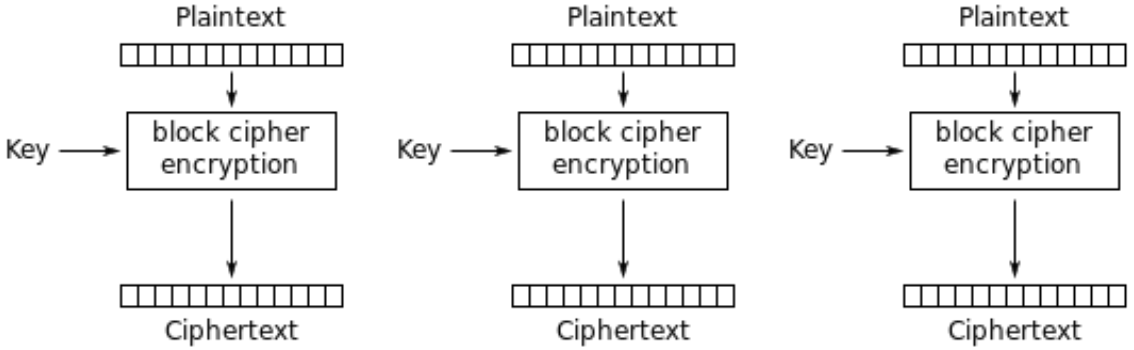


Cipher Block Chaining (CBC) mode encryption

Block Cipher Crypto Background

Electronic Codebook Mode

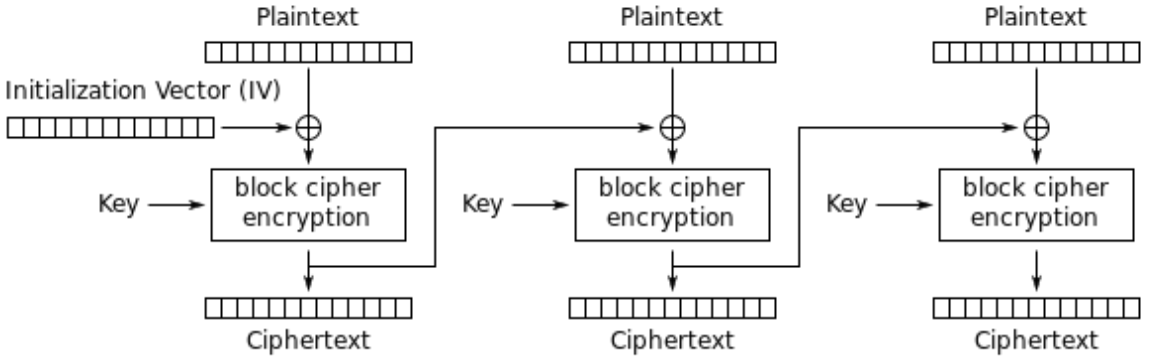
- Fast, parallelizable
- insecure (Why)?



Electronic Codebook (ECB) mode encryption

Cipher Block Chaining

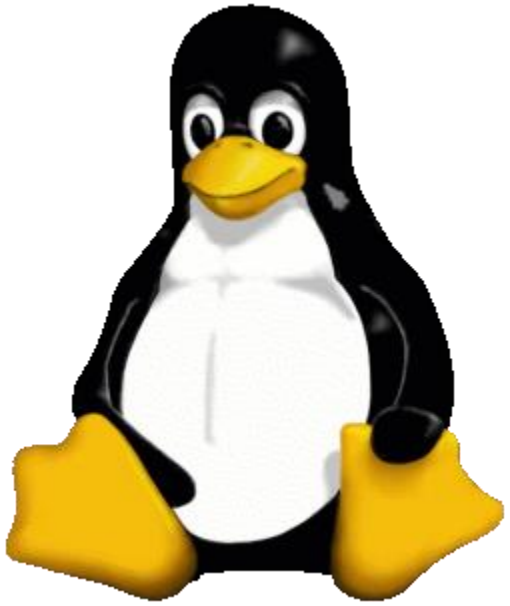
- e.g. xor(enc(prev-block), cur-block)
- need initialization vector (IV).
- Enc(bit) is changed → propagates



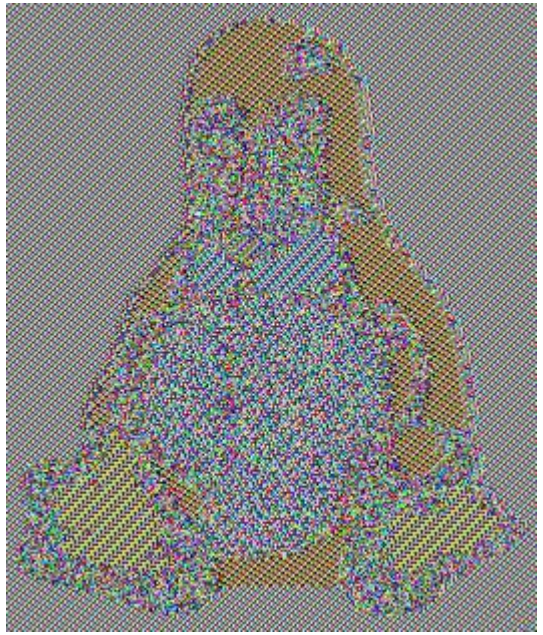
Cipher Block Chaining (CBC) mode encryption

ECB Penguin

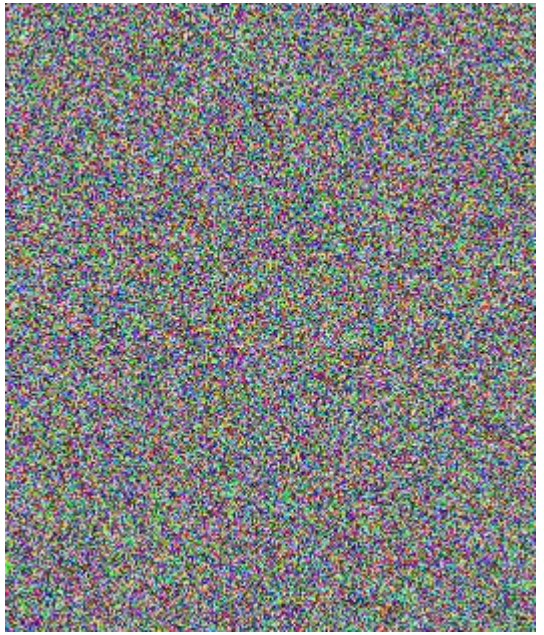
Original image



ECB encrypted



CBC encrypted



Signaling Errors

If the hash check of 2bl fails, what do we do?

Signaling Errors

If the hash check of 2bl fails, what do we do?

- ideal: blink the LED to indicate error

Signaling Errors

If the hash check of 2bl fails, what do we do?

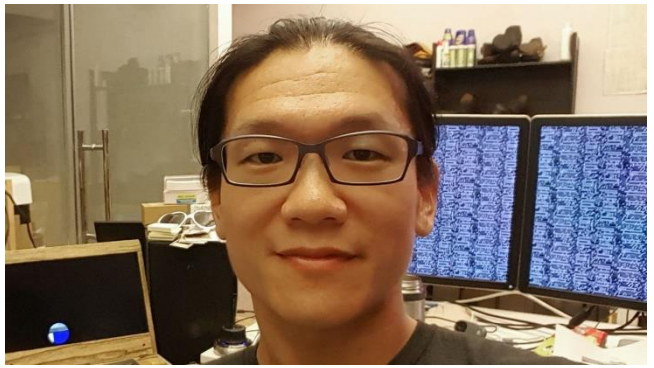
- ideal: blink the LED to indicate error
  - But must shut off ROM
  - Else hacker can read it from the Southbridge, while LED blinks.
  - But turn it off before the LED blinks → fall down into flash.
  - If halt the CPU before turning off the ROM, ROM left on
  - Can’t put halt in RAM, (vulnerable to attack (e.g., disabled by xcodes))

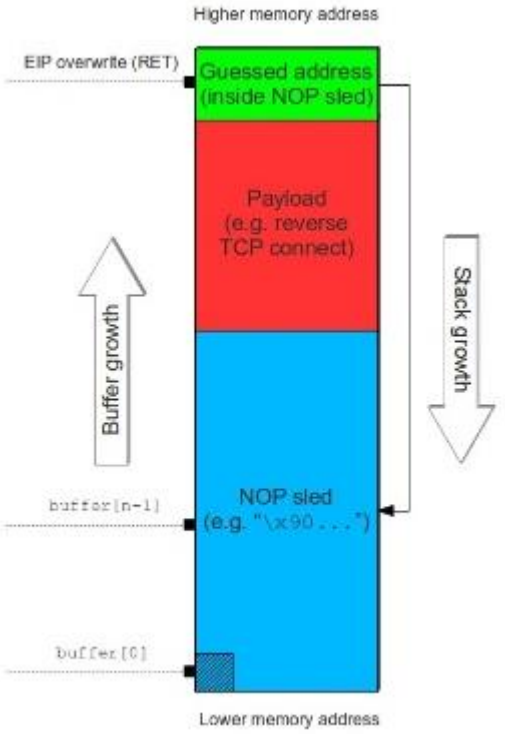
Signaling Errors

If the hash check of 2bl fails, what do we do?

- ideal: blink the LED to indicate error
  - But must shut off ROM
  - Else hacker can read it from the Southbridge, while LED blinks.
  - But turn it off before the LED blinks → fall down into flash.
  - If halt the CPU before turning off the ROM, ROM left on
  - Can't put halt in RAM, (vulnerable to attack (e.g., disabled by xcodes))
- Solution
  - turn off ROM as last instruction in the address space,
  - PC rolls over to 0, causing exception, double fault → halts the machine

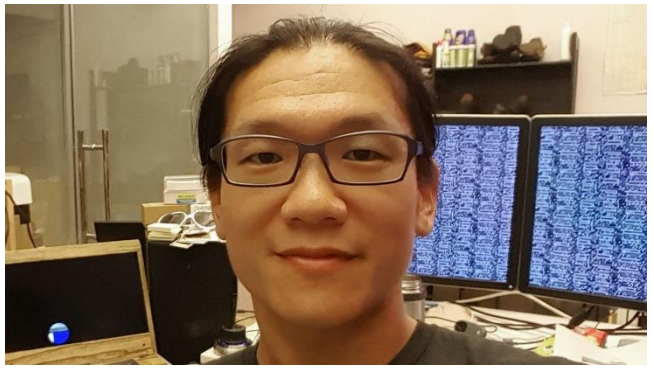
Exploit: Job 1 == read secret ROM

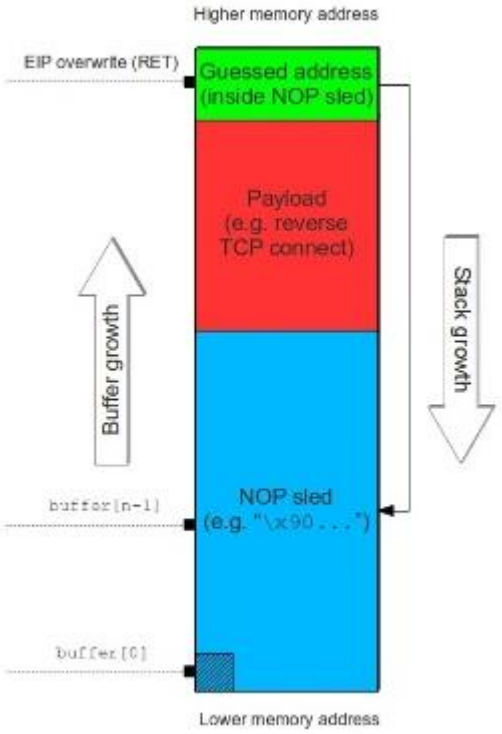




Exploit: Job 1 == read secret ROM

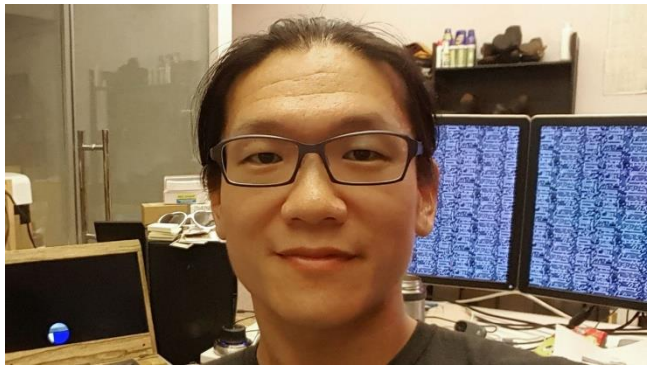
- OOPs: MS build tools include old secret ROM at end of flash!
  - Bunnie Huang uses MIT equipment to sniff HyperTransport contents of secret ROM.

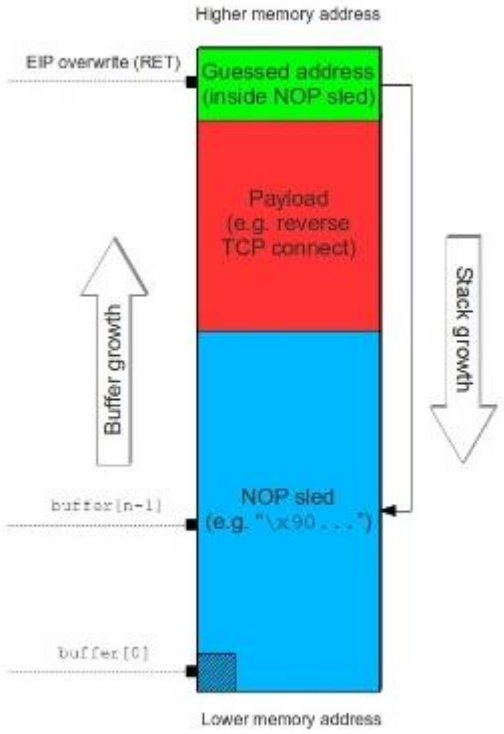




Exploit: Job 1 == read secret ROM

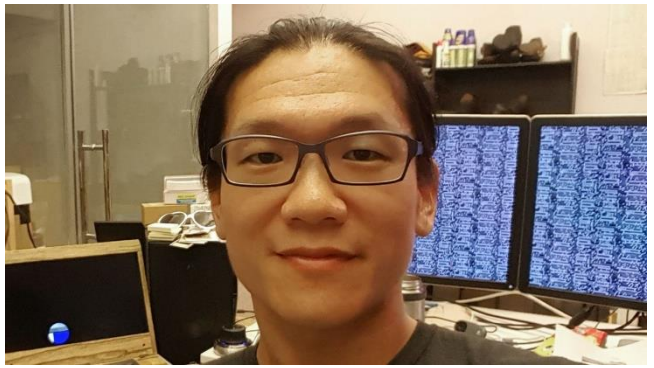
- OOPs: MS build tools include old secret ROM at end of flash!
  - Bunnie Huang uses MIT equipment to sniff HyperTransport contents of secret ROM.
- RC4 used in ECB mode
  - → testing the last 32-bit does not guarantee integrity of entire 2bl.
  - → 2bl is hacked.

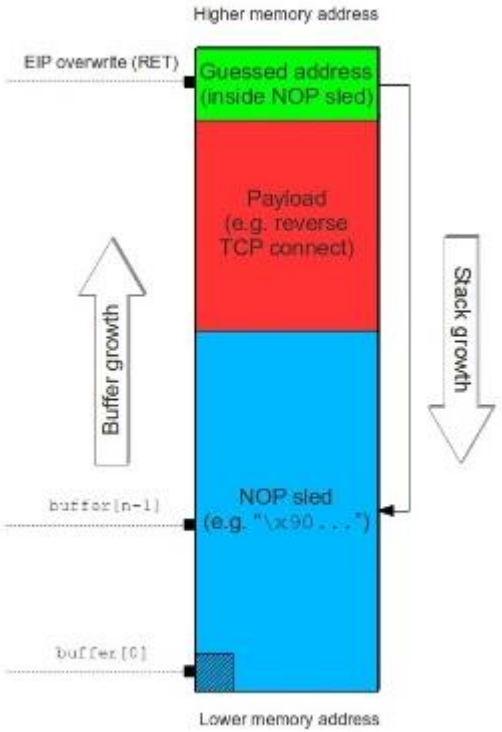




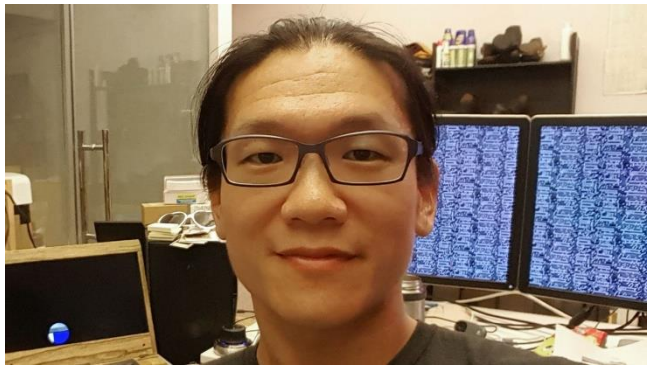
Exploit: Job 1 == read secret ROM

- OOPs: MS build tools include old secret ROM at end of flash!
  - Bunnie Huang uses MIT equipment to sniff HyperTransport contents of secret ROM.
- RC4 used in ECB mode
  - → testing the last 32-bit does not guarantee integrity of entire 2bl.
  - → 2bl is hacked.
- Linux hackers could encrypt their kernel and install it as a 2bl
  - would require use of the secret ROM key → not in the spirit of the GPL

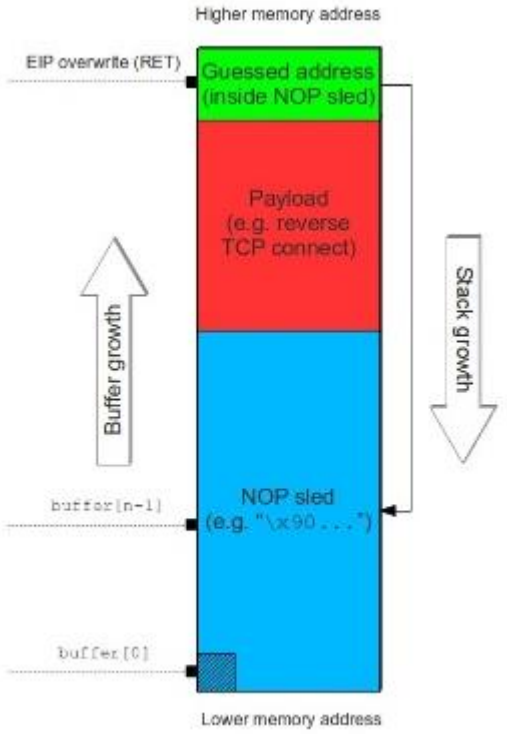




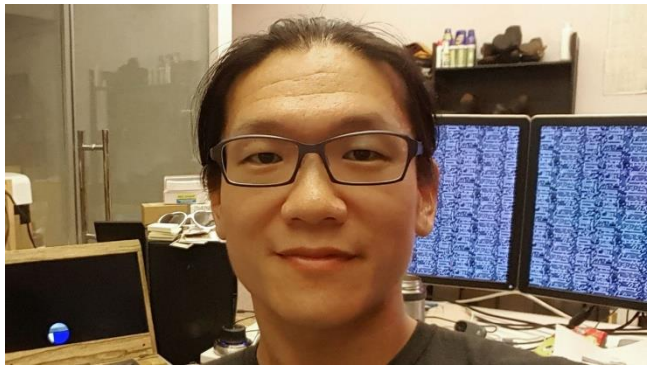
Exploit: Job 1 == read secret ROM



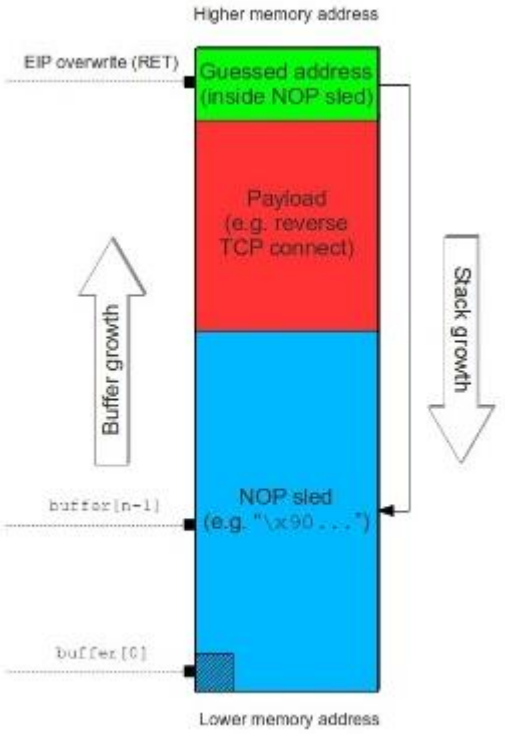
- OOPs: MS build tools include old secret ROM at end of flash!
  - Bunnie Huang uses MIT equipment to sniff HyperTransport contents of secret ROM.
- RC4 used in ECB mode
  - → testing the last 32-bit does not guarantee integrity of entire 2bl.
  - → 2bl is hacked.
- Linux hackers could encrypt their kernel and install it as a 2bl
  - would require use of the secret ROM key → not in the spirit of the GPL
- Intel CPUs don't fault on PC rollover.
  - → add additional xcode to write an instruction at 0 to jump to flash → provide zero 2bl (so decryption fails)
  - PCs start executing at high addresses, (Long ago PC makers wanted to map ROM at address 0)
  - Intel starts execution by reading from non-existent physical addresses, which generates 0xFFFF, or x86 nop.
  - CPU nops its way to the top of the address space, and rolls over to 0.
  - AMD does not implement this behavior, and AMD was the original provider for the xbox.



Exploit: Job 1 == read secret ROM



- OOPs: MS build tools include old secret ROM at end of flash!
  - Bunnie Huang uses MIT equipment to sniff HyperTransport contents of secret ROM.
- RC4 used in ECB mode
  - → testing the last 32-bit does not guarantee integrity of entire 2bl.
  - → 2bl is hacked.
- Linux hackers could encrypt their kernel and install it as a 2bl
  - would require use of the secret ROM key → not in the spirit of the GPL
- Intel CPUs don't fault on PC rollover.
  - → add additional xcode to write an instruction at 0 to jump to flash → provide zero 2bl (so decryption fails)
  - PCs start executing at high addresses, (Long ago PC makers wanted to map ROM at address 0)
  - Intel starts execution by reading from non-existent physical addresses, which generates 0xFFFF, or x86 nop.
  - CPU nops its way to the top of the address space, and rolls over to 0.
  - AMD does not implement this behavior, and AMD was the original provider for the xbox.
- xcode restriction on disabling secret ROM via PCI register write: flawed.
  - checks against a single constant. put a nop sled at the top of flash,
  - disable secret ROM with xcodes, slide down sled to jump to beginning of (modified) flash.
  - Defense code made attack easier.



Xbox v1.1 Fixes

Xbox v1.1 Fixes

- Supposed Fixes
  - Changed the decryption key in secret ROM.
  - RC4 → TEA
  - Accessing ROM while off → crash (no fall downs).
  - remove pins in the low pin count (LPC) bus.
  - makes modchips harder (but not much) to install

Xbox v1.1 Fixes

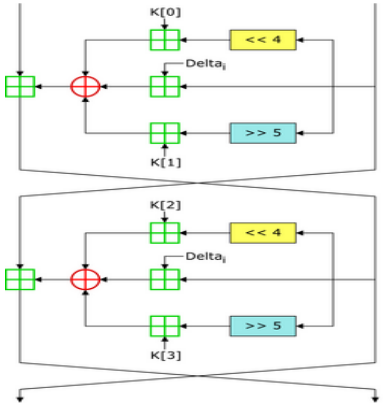
- Supposed Fixes
  - Changed the decryption key in secret ROM.
  - RC4 → TEA
  - Accessing ROM while off → crash (no fall downs).
  - remove pins in the low pin count (LPC) bus.
  - makes modchips harder (but not much) to install
- TEA

Xbox v1.1 Fixes

- Supposed Fixes
  - Changed the decryption key in secret ROM.
  - RC4 → TEA
  - Accessing ROM while off → crash (no fall downs).
  - remove pins in the low pin count (LPC) bus.
  - makes modchips harder (but not much) to install
- TEA
  - bad hash function
  - allows mods to xcodes that still match hash.
  - Hackers patch a jump in 2bl to enter flash.

TEA

- Operates on 2 uint32_t's 128-bit key
- Key-schedule identical for each cycle
- Magic constant
  - used to combat symmetry across rounds
  - = 2^32 / golden-ratio
- Each key equivalent to 3 others
  - Poor as hash function
  - Breakable with 2^23 chosen plaintexts

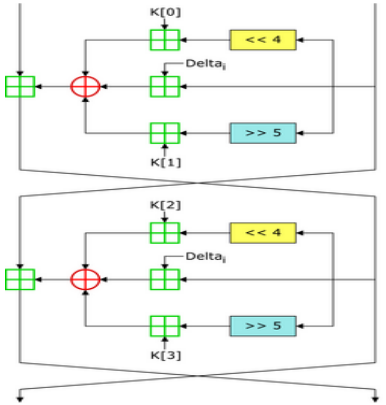


Xbox v1.1 Fixes

- Supposed Fixes
  - Changed the decryption key in secret ROM.
  - RC4 → TEA
  - Accessing ROM while off → crash (no fall downs).
  - remove pins in the low pin count (LPC) bus.
  - makes modchips harder (but not much) to install
- TEA
  - bad hash function
  - allows mods to xcodes that still match hash.
  - Hackers patch a jump in 2bl to enter flash.
- The A20 I/O pin grounds address line 20, making address 1MB == 0.
  - remaps secret ROM addresses into flash memory.
  - secret ROM is still on, so it can be dumped by a slow bus.

TEA

- Operates on 2 uint32_t's 128-bit key
- Key-schedule identical for each cycle
- Magic constant
  - used to combat symmetry across rounds
  - = 2^32 / golden-ratio
- Each key equivalent to 3 others
  - Poor as hash function
  - Breakable with 2^23 chosen plaintexts



V1.1 Exploits

V1.1 Exploits

- Savegames are saved and restored from USB drives.

V1.1 Exploits

- Savegames are saved and restored from USB drives.
- modify savegames → buffer overflows in savegame loaders.
  - buffer overflows only compromise user-level programs.
  - all games execute in kernel mode on the xbox.
  - then overwrite flash,
  - but need to solder a bridge make flash writable
  - or just run savegame every time you need unsigned code.

Potpourri

Potpourri

- Note
  - Linux hackers wanted solution that would not allow illegal games
  - …but could not find one.

Potpourri

- Note
  - Linux hackers wanted solution that would not allow illegal games
  - …but could not find one.
- Dashboard program is signed and contains hashes of all data.
  - EXCEPT two font files
  - So, corrupt font file.
  - Start xbox, dashboard starts,
  - crashes with corrupt font file and
  - runs exploit which accepts xbox and xbox Linux-signed code.

Potpourri

- Note
  - Linux hackers wanted solution that would not allow illegal games
  - …but could not find one.
- Dashboard program is signed and contains hashes of all data.
  - EXCEPT two font files
  - So, corrupt font file.
  - Start xbox, dashboard starts,
  - crashes with corrupt font file and
  - runs exploit which accepts xbox and xbox Linux-signed code.
- Crazy back and forth between MS and hackers

Potpourri

- Note
  - Linux hackers wanted solution that would not allow illegal games
  - …but could not find one.
- Dashboard program is signed and contains hashes of all data.
  - EXCEPT two font files
  - So, corrupt font file.
  - Start xbox, dashboard starts,
  - crashes with corrupt font file and
  - runs exploit which accepts xbox and xbox Linux-signed code.
- Crazy back and forth between MS and hackers
- ***ENOUGH ALREADY!!!!***

Ostensible Lessons

Design

- Security v Money
- Security v Performance
- Combined Weaknesses
- “Harder for Hackers” meaningless
- single security → different purposes
- security by obscurity
- quick fixes

Implementation

- Read data sheets
- Read literature
- Get professionals
- Be complete
- Inspect final artifacts
- End to end re-evaluation after changes

Policy

- Source code management
- Many people scrutinize
- Talk to Enemies

Ostensible Lessons

Design

- Security v Money
- Security v Performance
- Combined Weaknesses
- “Harder for Hackers” meaningless
- single security → different purposes
- security by obscurity
- quick fixes

Implementation

- Read data sheets
- Read literature
- Get professionals
- Be complete
- Inspect final artifacts
- End to end re-evaluation after changes

Policy

- Source code management
- Many people scrutinize
- Talk to Enemies

- Defense in depth.
- Don’t run in kernel mode!
- Know your attacker’s motivations.
- Security by obscurity does not work.