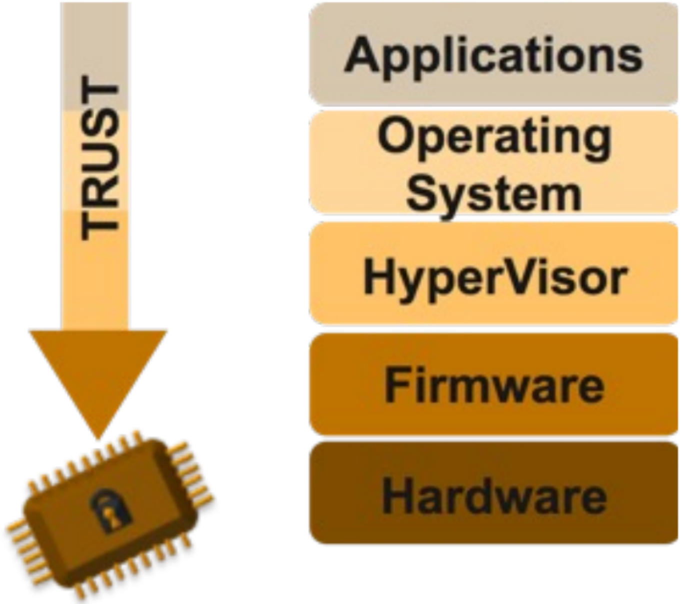
Computer and Network Security

Lecture 15: Hardware Security

COMP-5370/6370
Fall 2024

Trusted Computing Base

The **Trusted Computing Base (TCB)** is the collection of all components within a system critical to providing security properties.



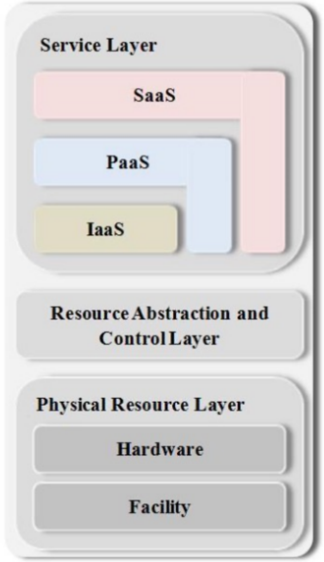
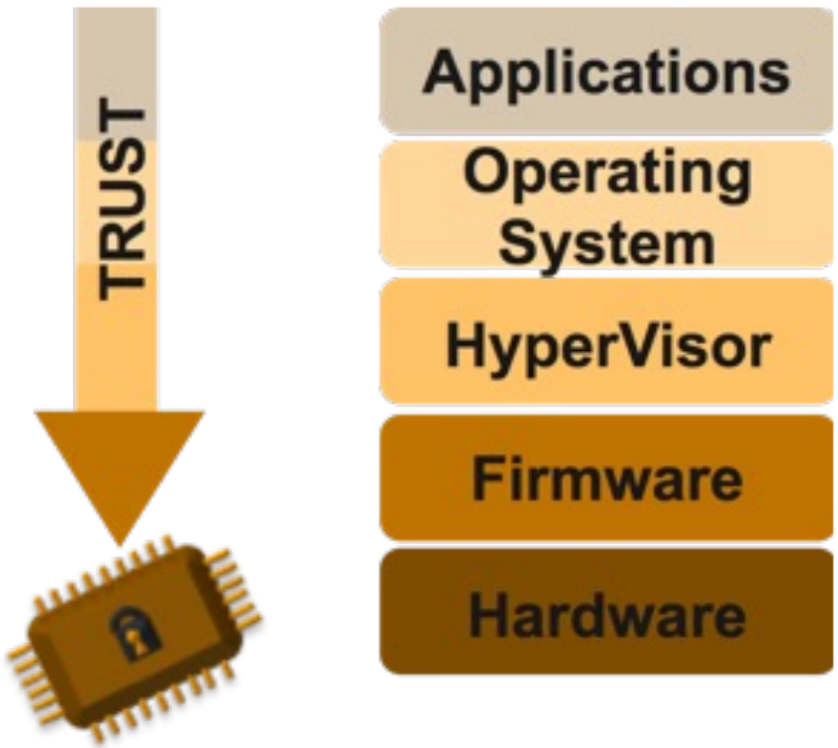


Figure 15: Cloud Provider - Service Orchestration

Local TCB



- Everything needed to run the application safely
- Each layer relies on the layers below it to behave correctly

Let's Connect to a Server

Let's Connect to a Server

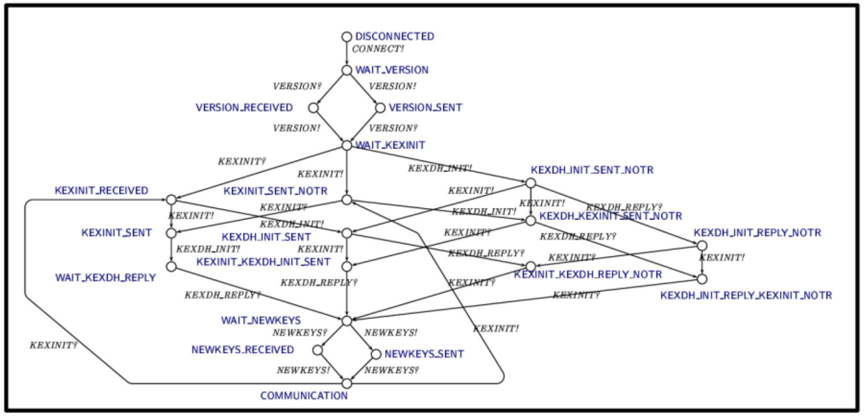
```
$>
$> ssh username@host
```

- Bash Environment

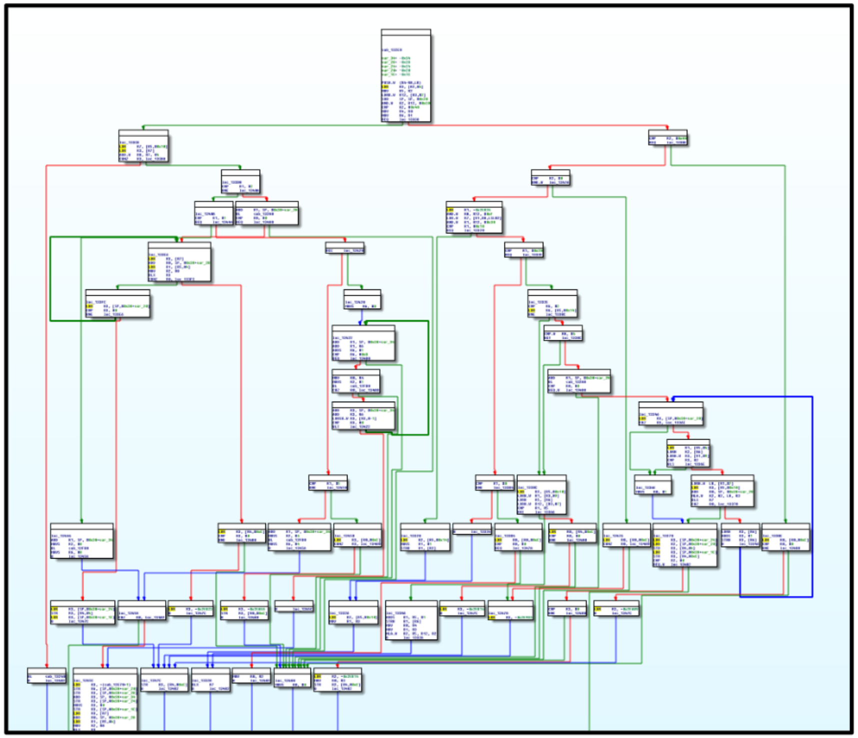
```
$> which ssh
/tmp/malwarehooks/ssh
```

Let's Connect to a Server

- Bash Environment
- SSH protocol design

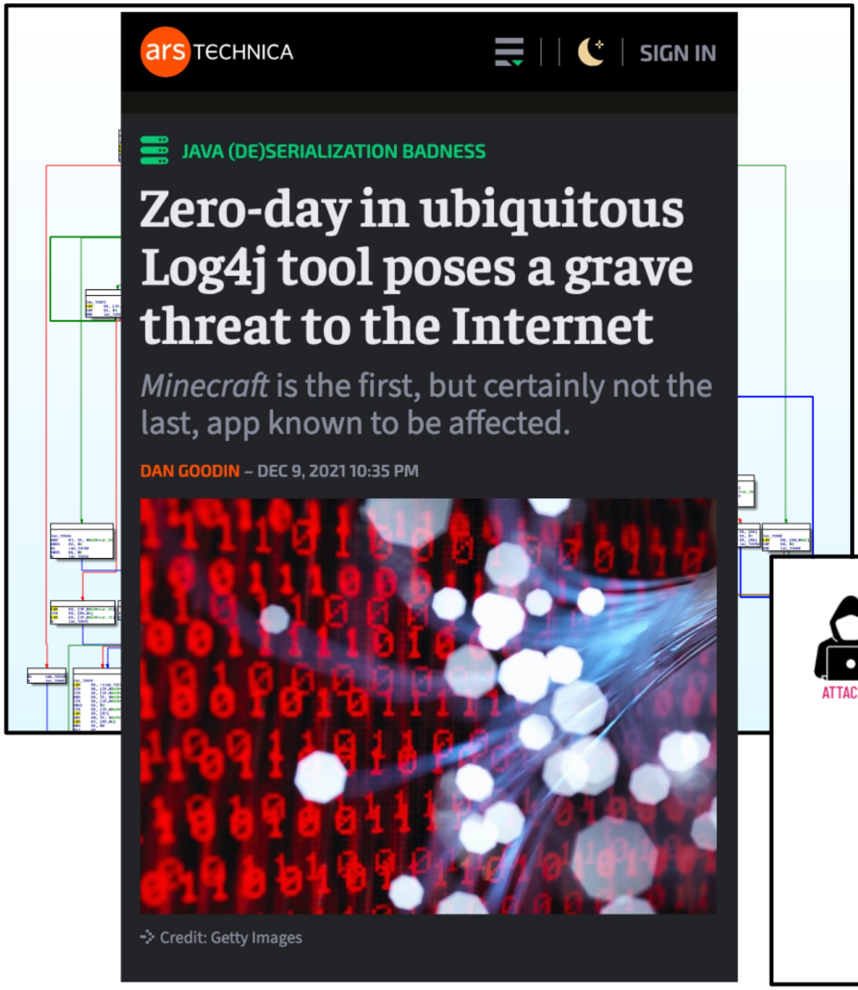


Let's Connect to a Server



- Bash Environment
- SSH protocol design
- SSH app architecture

Let’s Connect to a Server



ars TECHNICA | SIGN IN

JAVA (DE)SERIALIZATION BADNESS

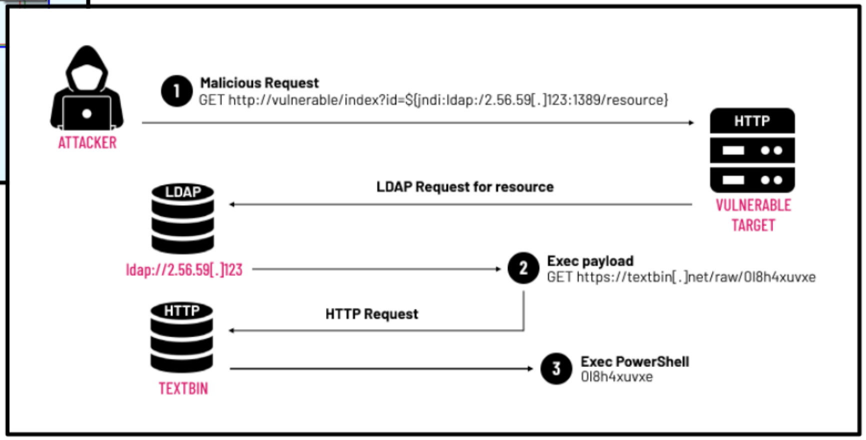
Zero-day in ubiquitous Log4j tool poses a grave threat to the Internet

Minecraft is the first, but certainly not the last, app known to be affected.

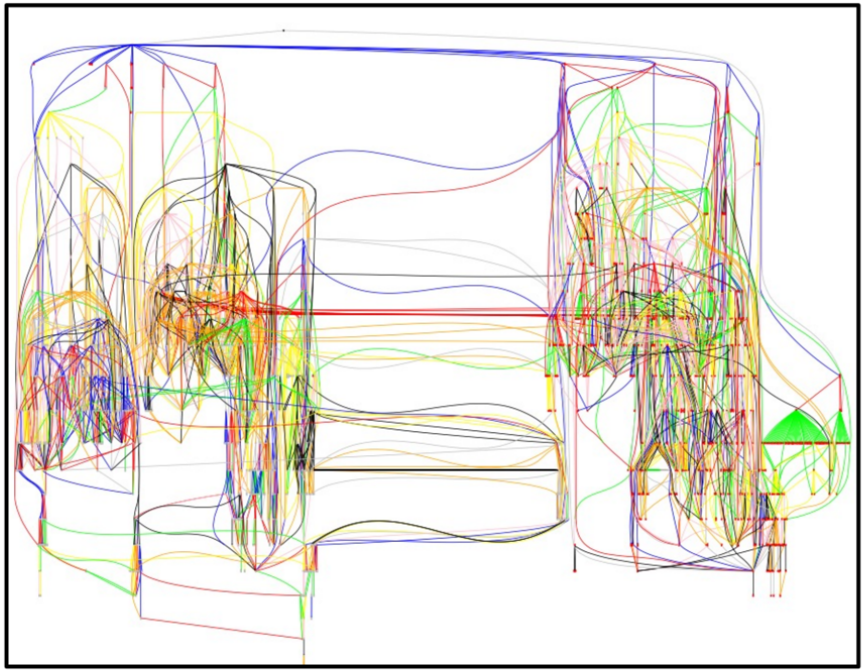
DAN GOODIN – DEC 9, 2021 10:35 PM

Credit: Getty Images

- Bash Environment
- SSH protocol design
- SSH app architecture



Let's Connect to a Server



- Bash Environment
- SSH protocol design
- SSH app architecture
- SSH app logic

Let's Connect to a Server

$$
\begin{aligned}
SEC_k &: \mathsf{Hon}(\hat{C}, \hat{K}) \supset (\mathsf{GoodKeyAgainst}(X, k) \vee \hat{X} \in \{\hat{C}, \hat{K}\}) \\
SEC_{akey} &: \mathsf{Hon}(\hat{C}, \hat{K}, \hat{T}) \supset (\mathsf{GoodKeyAgainst}(X, AKey) \vee \hat{X} \in \{\hat{C}, \hat{K}, \hat{T}\}) \\
SEC_{skey} &: \mathsf{Hon}(\hat{C}, \hat{K}, \hat{T}, \hat{S}) \supset (\mathsf{GoodKeyAgainst}(X, SKey) \vee \hat{X} \in \{\hat{C}, \hat{K}, \hat{T}, \hat{S}\}) \\
AUTH_{kas} &: \exists \eta.\ \mathsf{Send}((\hat{K}, \eta), Cert_K.SIG[sk_K](\text{``}DHKey\text{''}.gy.\tilde{n_1}).E_{sym}[k^{t \to k}_{T,K}](AKey.\hat{C}). \\
&\quad E_{sym}[k](AKey.n_1.\hat{T})) \\
AUTH_{tgs} &: \exists \eta.\ \mathsf{Send}((\hat{T}, \eta), \hat{C}.E_{sym}[k^{s \to t}_{S,T}](SKey.\hat{C}).E_{sym}[AKey](SKey.n_2.\hat{S}))
\end{aligned}
$$

$$
\begin{aligned}
SEC^{client}_k &: [\mathbf{Client}]_C\ SEC_k & SEC^{kas}_k &: [\mathbf{KAS}]_K\ SEC_k
\end{aligned}
$$

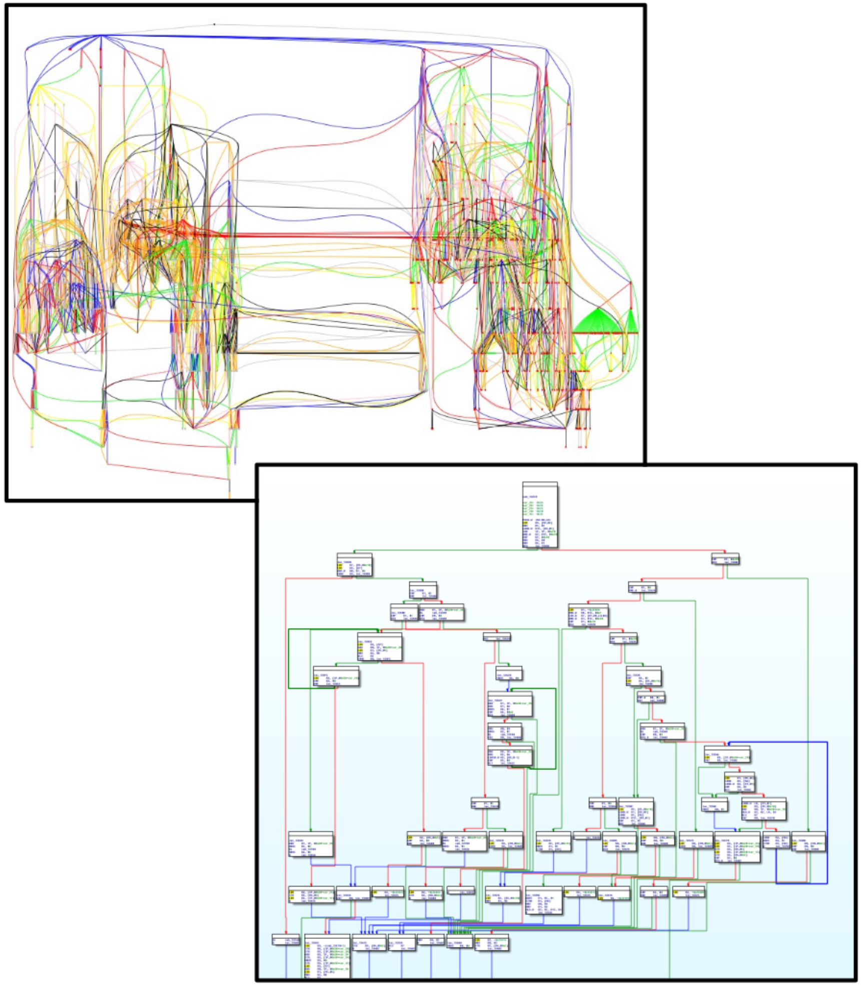
$$
\begin{aligned}
SEC^{client}_{akey} &: [\mathbf{Client}]_C\ SEC_{akey} & AUTH^{client}_{kas} &: [\mathbf{Client}]_C\ \mathsf{Hon}(\hat{C}, \hat{K}) \supset AUTH_{kas} \\
SEC^{kas}_{akey} &: [\mathbf{KAS}]_K\ SEC_{akey} & AUTH^{tgs}_{kas} &: [\mathbf{TGS}]_T\ \mathsf{Hon}(\hat{T}, \hat{K}) \\
SEC^{tgs}_{akey} &: [\mathbf{TGS}]_T\ SEC_{akey} & &\quad \supset \exists n_1, k, gy, \tilde{n_1}.\ AUTH_{kas} \\
& & AUTH^{client}_{tgs} &: [\mathbf{Client}]_C\ \mathsf{Hon}(\hat{C}, \hat{K}, \hat{T}) \supset AUTH_{tgs} \\
SEC^{client}_{skey} &: [\mathbf{Client}]_C\ SEC_{skey} & AUTH^{server}_{tgs} &: [\mathbf{Server}]_S\ \mathsf{Hon}(\hat{S}, \hat{T}) \\
SEC^{tgs}_{skey} &: [\mathbf{TGS}]_T\ SEC_{skey} & &\quad \supset \exists n_2, AKey.\ AUTH_{tgs}
\end{aligned}
$$

Table 1. DHINIT Security Properties

$$
\begin{aligned}
C \longrightarrow K(I) &: Cert_C.SIG[sk_C](\text{``}Auth\text{''}.HASH(\hat{C}.\hat{T}.n_1).\tilde{n_1}.gx).\hat{C}.\hat{T}.n_1 \\
I \longrightarrow K &: Cert_I.SIG[sk_I](\text{``}Auth\text{''}.HASH(\hat{I}.\hat{T}.n_1).\tilde{n_1}.gx).\hat{I}.\hat{T}.n_1 \\
K \longrightarrow I \longrightarrow C &: Cert_K.SIG[sk_K](\text{``}DHKey\text{''}.gy.\tilde{n_1}). \\
&\quad E_{sym}[k^{t \to k}_{T,K}](AKey.\hat{I}).E_{sym}[k](AKey.n_1.\hat{T})
\end{aligned}
$$

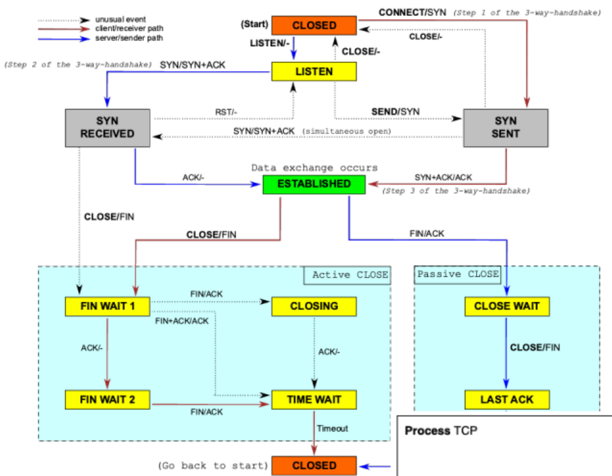
- Bash Environment
- SSH protocol design
- SSH app architecture
- SSH app logic
- Crypto math

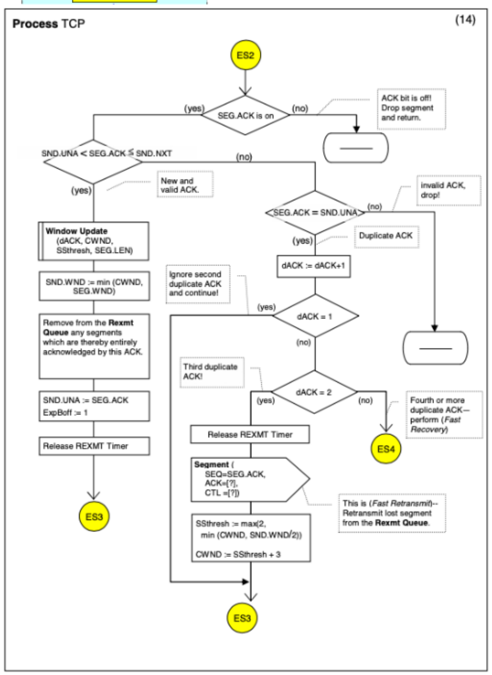
Let's Connect to a Server



- Bash Environment
- SSH protocol design
- SSH app architecture
- SSH app logic
- Crypto math
- Crypto implementation

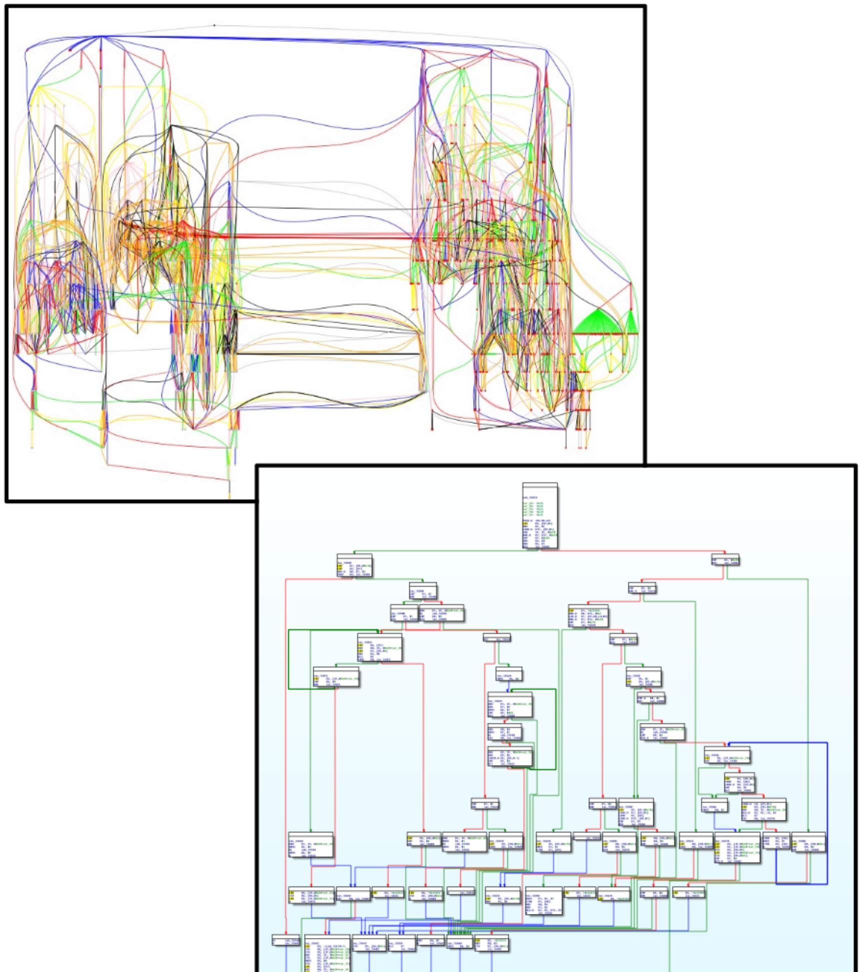
Let’s Connect to a Server





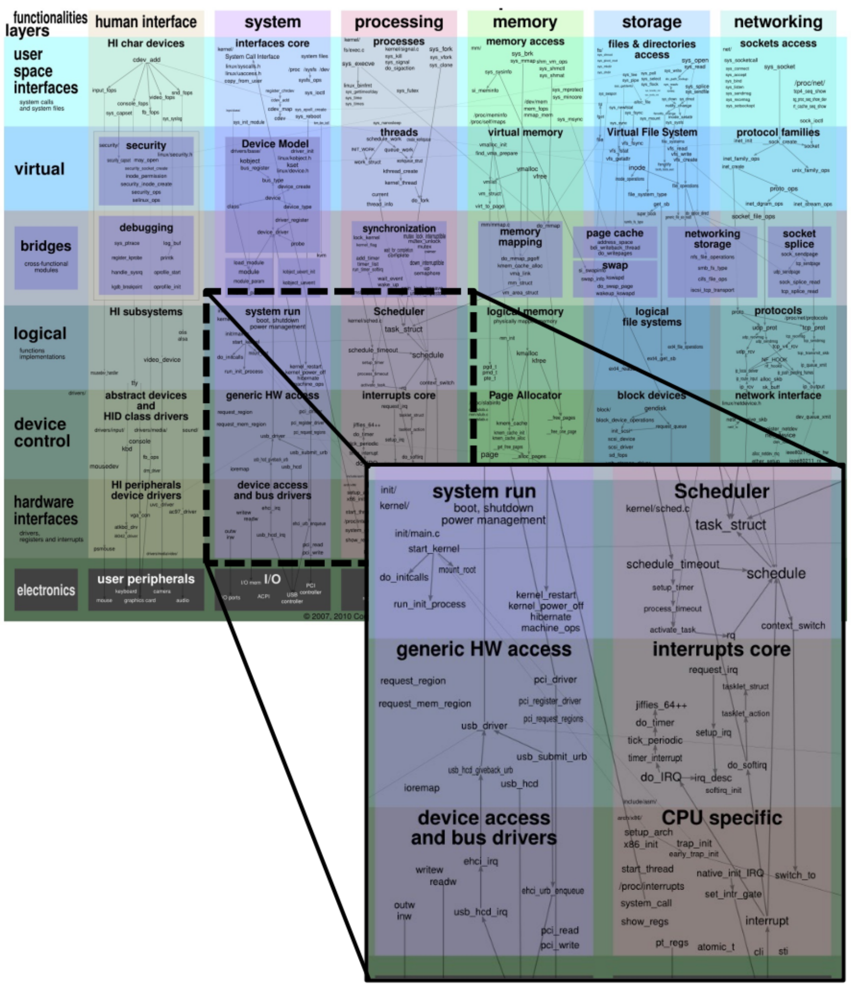
- Bash Environment
- SSH protocol design
- SSH app architecture
- SSH app logic
- Crypto math
- Crypto implementation
- Channel design

Let's Connect to a Server



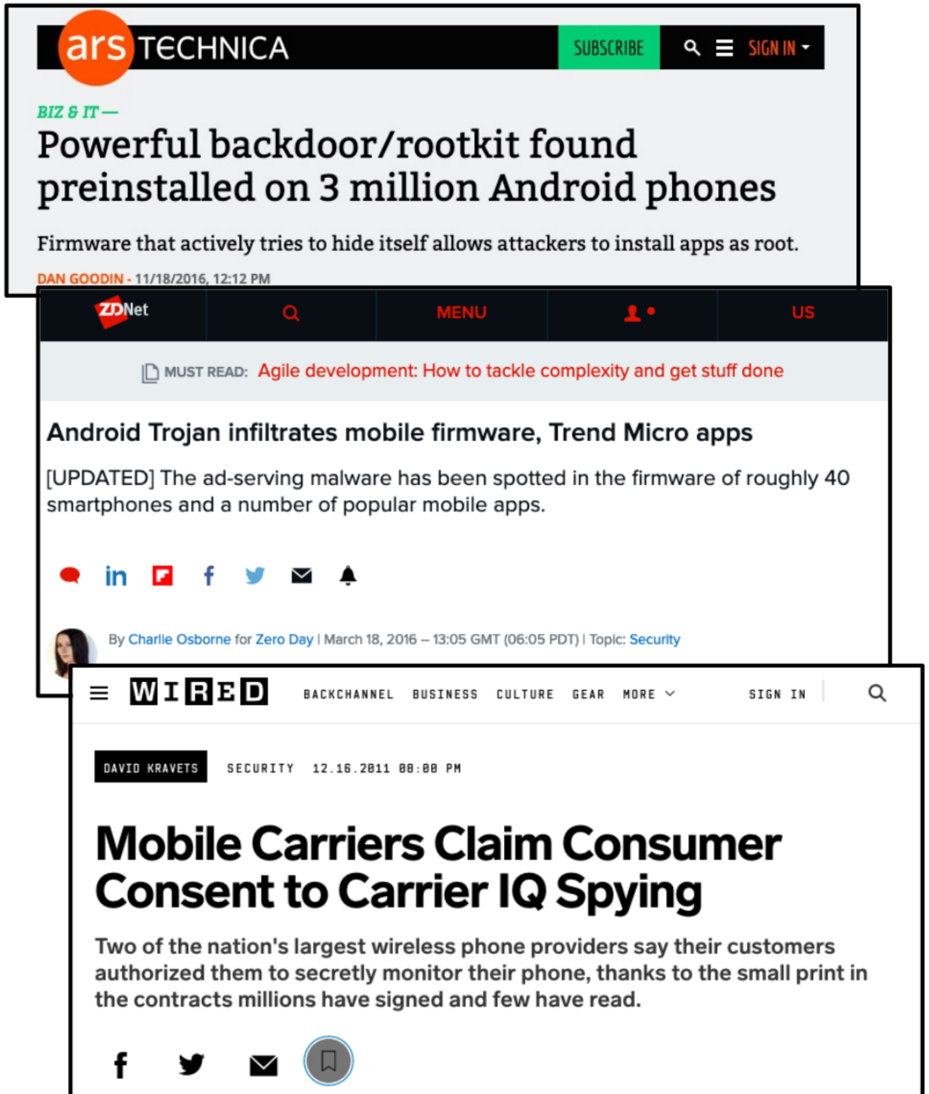
- Bash Environment
- SSH protocol design
- SSH app architecture
- SSH app logic
- Crypto math
- Crypto implementation
- Channel design
- Channel implementation

Let’s Connect to a Server



- Bash Environment
- SSH protocol design
- SSH app architecture
- SSH app logic
- Crypto math
- Crypto implementation
- Channel design
- Channel implementation
- OS implementation

Android is Notorious



ars TECHNICA SUBSCRIBE SIGN IN

BIZ & IT —

Powerful backdoor/rootkit found preinstalled on 3 million Android phones

Firmware that actively tries to hide itself allows attackers to install apps as root.

DAN GOODIN - 11/18/2016, 12:12 PM

ZDNet MENU US

MUST READ: Agile development: How to tackle complexity and get stuff done

Android Trojan infiltrates mobile firmware, Trend Micro apps

[UPDATED] The ad-serving malware has been spotted in the firmware of roughly 40 smartphones and a number of popular mobile apps.

By Charlie Osborne for Zero Day | March 18, 2016 -- 13:05 GMT (06:05 PDT) | Topic: Security

WIRED BACKCHANNEL BUSINESS CULTURE GEAR MORE SIGN IN

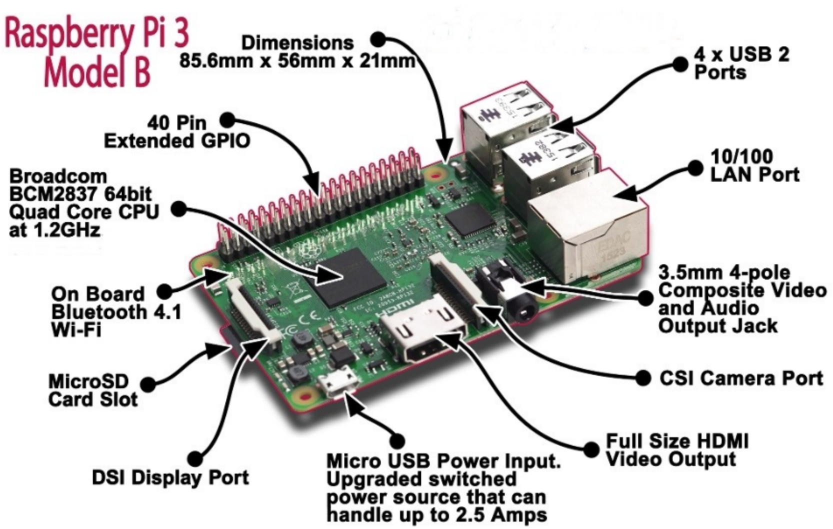
DAVID KRAVETS SECURITY 12.16.2011 08:00 PM

Mobile Carriers Claim Consumer Consent to Carrier IQ Spying

Two of the nation's largest wireless phone providers say their customers authorized them to secretly monitor their phone, thanks to the small print in the contracts millions have signed and few have read.

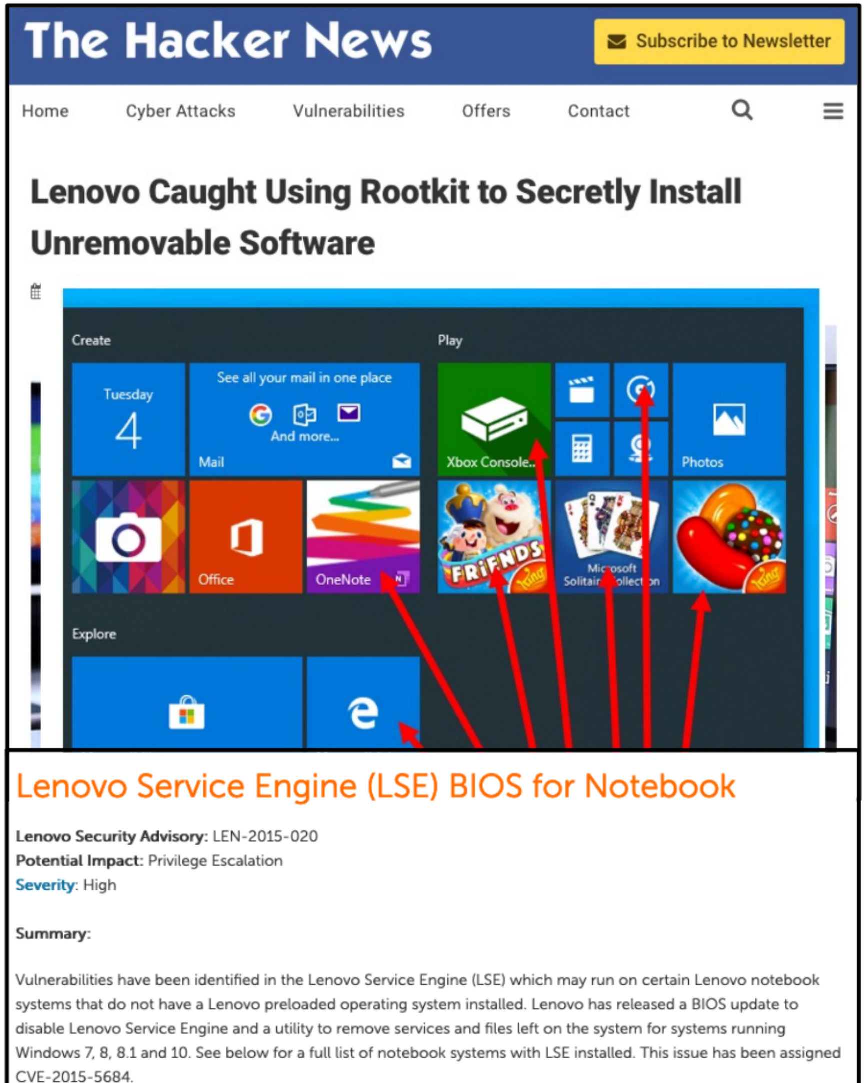
- "System" apps are 3P apps that are treated as part of the OS
- Pre-installed apps that can't be uninstalled
  - Carrier IQ was first major one (2011)
  - Far from the only

Let's Connect to a Server



- Bash Environment
- SSH protocol design
- SSH app architecture
- SSH app logic
- Crypto math
- Crypto implementation
- Channel design
- Channel implementation
- OS implementation
- Firmware for HW

Lenovo Firmware Rootkit



Lenovo Service Engine (LSE) BIOS for Notebook

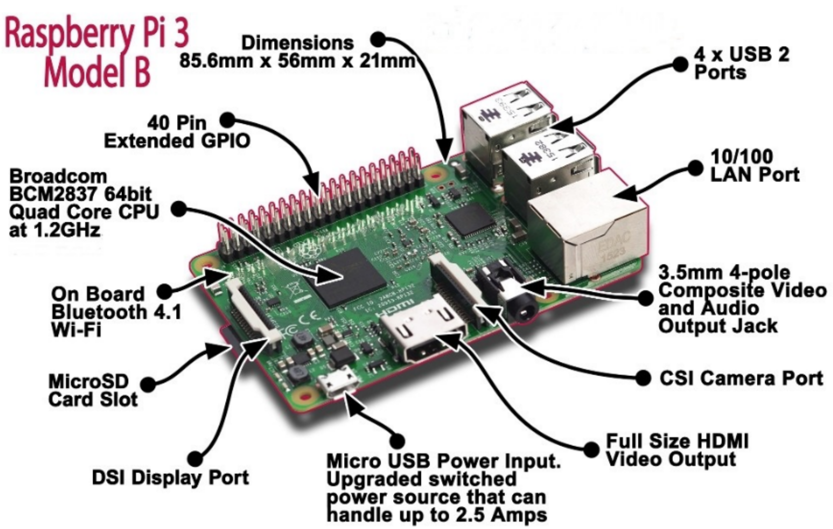
Lenovo Security Advisory: LEN-2015-020
Potential Impact: Privilege Escalation
Severity: High

Summary:

Vulnerabilities have been identified in the Lenovo Service Engine (LSE) which may run on certain Lenovo notebook systems that do not have a Lenovo preloaded operating system installed. Lenovo has released a BIOS update to disable Lenovo Service Engine and a utility to remove services and files left on the system for systems running Windows 7, 8, 8.1 and 10. See below for a full list of notebook systems with LSE installed. This issue has been assigned CVE-2015-5684.

- BIOS replaces Win7-10 OS files during boot
- Sent device-specific info to corp. on 1st boot
- Automatically installed applications in OS
  - Auto-updates for drivers
  - Bloat-ware
- Had exploitable vulnerabilities

Let's Connect to a Server



- Bash Environment
- SSH protocol design
- SSH app architecture
- SSH app logic
- Crypto math
- Crypto implementation
- Channel design
- Channel implementation
- OS implementation
- Firmware for HW
- HW components

Rowhammer

Project Zero

News and updates from the Project Zero team at Google

Monday, March 9, 2015

Exploiting the DRAM rowhammer bug to gain kernel privileges

Posted by Mark Seaborn, sandbox builder and breaker, with contributions by Thomas Dullien, reverse engineer

[This guest post continues Project Zero's practice of promoting excellence in security research on the Project Zero blog]

ars TECHNICA SUBSCRIBE SIGN IN

RAMBLEED —

Researchers use Rowhammer bit flips to steal 2048-bit crypto key

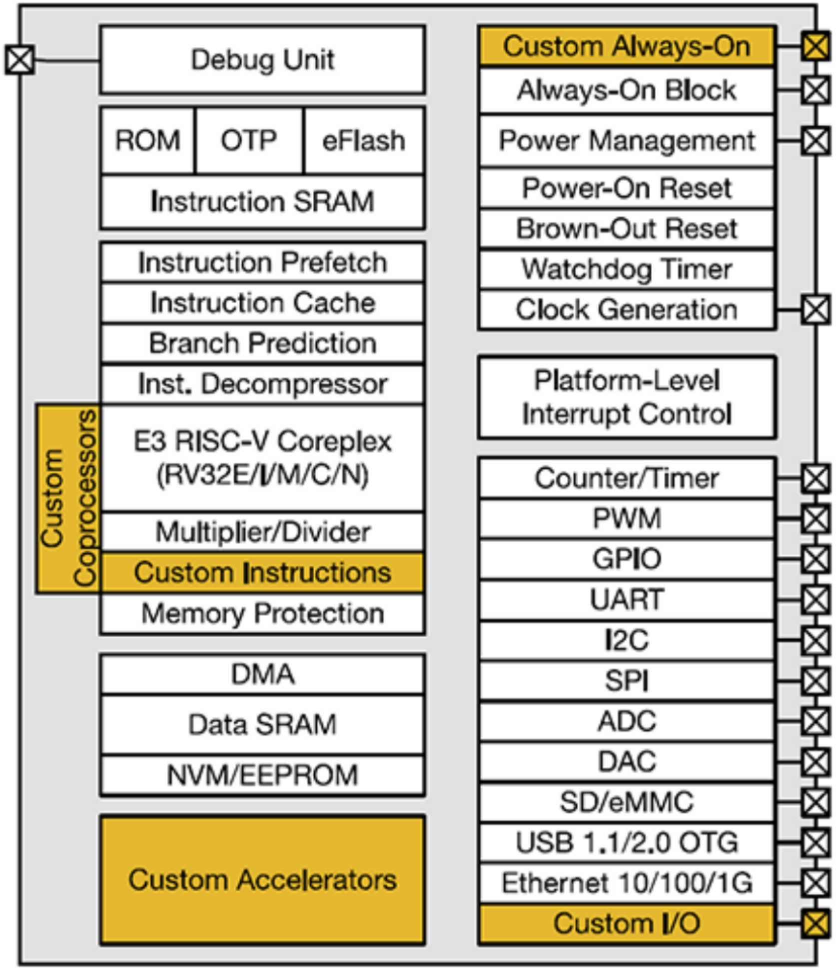
RAMBleed side-channel attack works even when DRAM is protected by error-correcting code.

DAN GOODIN - 6/11/2019, 12:00 PM

- Allows arbitrary process to flip bits in physical memory
- Predictable but not 100% perfect

Think of all the things you can do with 1 bit.

Let’s Connect to a Server



- Bash Environment
- SSH protocol design
- SSH app architecture
- SSH app logic
- Crypto math
- Crypto implementation
- Channel design
- Channel implementation
- OS implementation
- Firmware for HW
- HW components
- SoC design/impl.
- Chip design/impl.

CPU Management

The Intel Management Engine is Really Rather Scary

This closed source non-auditable subsystem can:

- Access all areas of your computer's memory, without the CPU's knowledge.
- Access every peripheral attached to your computer.
- Set up a TCP/IP server on your network interface that can send and receive

WIRED — BACKCHANNEL BUSINESS CULTURE GEAR MORE SIGN IN

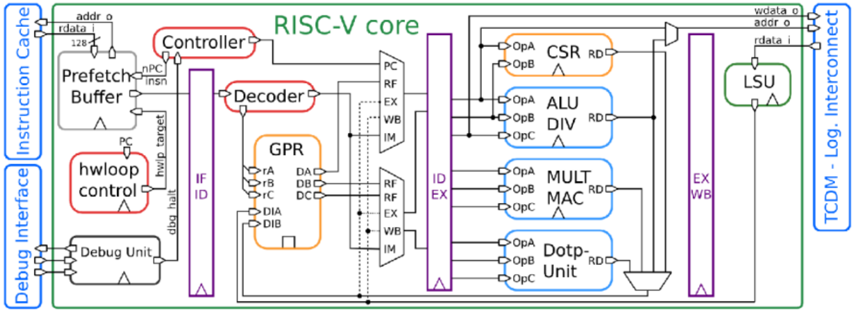
LILY HAY NEWMAN SECURITY 11.20.2017 11:10 PM

Intel Chip Flaws Leave Millions of Devices Exposed

Security experts have warned of Intel's Management Engine for years. A new set of confirmed vulnerabilities that impact PCs, servers, and IoT devices shows they may have been right.

- Completely separate co-processor built into Intel CPUs
- AMD Platform Security Processor (AMD PSP)
  - All chips since 2013
- Intel Management Engine (Intel ME)
  - All chips since 2008

Let's Connect to a Server



- Bash Environment
- SSH protocol design
- SSH app architecture
- SSH app logic
- Crypto math
- Crypto implementation
- Channel design
- Channel implementation
- OS implementation
- Firmware for HW
- SoC design/impl.
- CPU design/arch

CPU Bugs

OSDev.org

Page | Discussion | Read | View source | View

Navigation
- Main Page
- Forums
- FAQ
- OS Projects
- Random page

About
- This site
- Joining
- Editing help
- Recent changes

Toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link

CPU Bugs

Computers are made by humans, and thus inherently prone to errors. This page describes known bugs for various models and brands.

Contents [hide]

CPU Bug Examples

Pentium FDIV Bug

$$c = 4195835 / 3145727$$

appeared to be an instance of the worst-case error. Coe did his analysis without actually using a Pentium—he doesn't own one. To verify his prediction, he had to bundle his year-and-a-half-old daughter into his car, drive to a local computer store, and borrow a demonstration machine.

The true value of Coe's ratio is

$$c = 1.33382044\ldots.$$

But computed on a Pentium, it is

$$c = 1.33373906\ldots.$$

Cyrix Coma Bug

```
unsigned char c[4] = {0x36, 0x78, 0x38, 0x36};
int main()
{
  asm (
  "          movl    $c, %ebx\n"
  "again:    xchgl   (%ebx), %eax\n"
  "          movl    %eax, %edx\n"
  "          jmp     again\n"
  );
}
```

HALT AND CATCH FIRE (HCF):

An early computer command that sent the machine into a race condition, forcing all instructions to compete for superiority at once.

Control of the computer could not be regained._

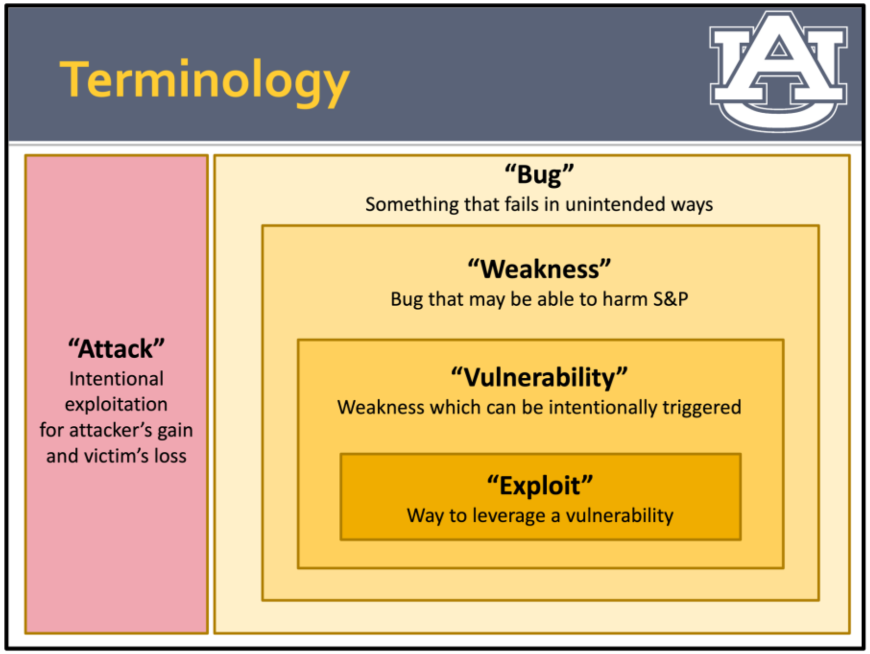
CPU Microcode

Microcode is firmware that acts as a translator for CPUs and turns *opcodes* into *micro-operations* which are executed.

- RISC won the *RISC vs. CISC War*
  - RISC-style is much more efficient
- Allows manufacturers to patch w/o recall
  - Just install a microcode patch
- **Completely unknown to almost everyone**

Software Facts of Life

- Software has bugs
- Some bugs are weaknesses
- Some weaknesses are vulnerabilities
- Some vulnerabilities can be exploited
- Someone has an interest in exploiting others for gain
- Malware is a different breed of software



Microcode Vulnerabilities

Side Channel Vulnerabilities:
Microarchitectural Data Sampling and
Transactional Asynchronous Abort

ars TECHNICA SUBSCRIBE SIGN IN

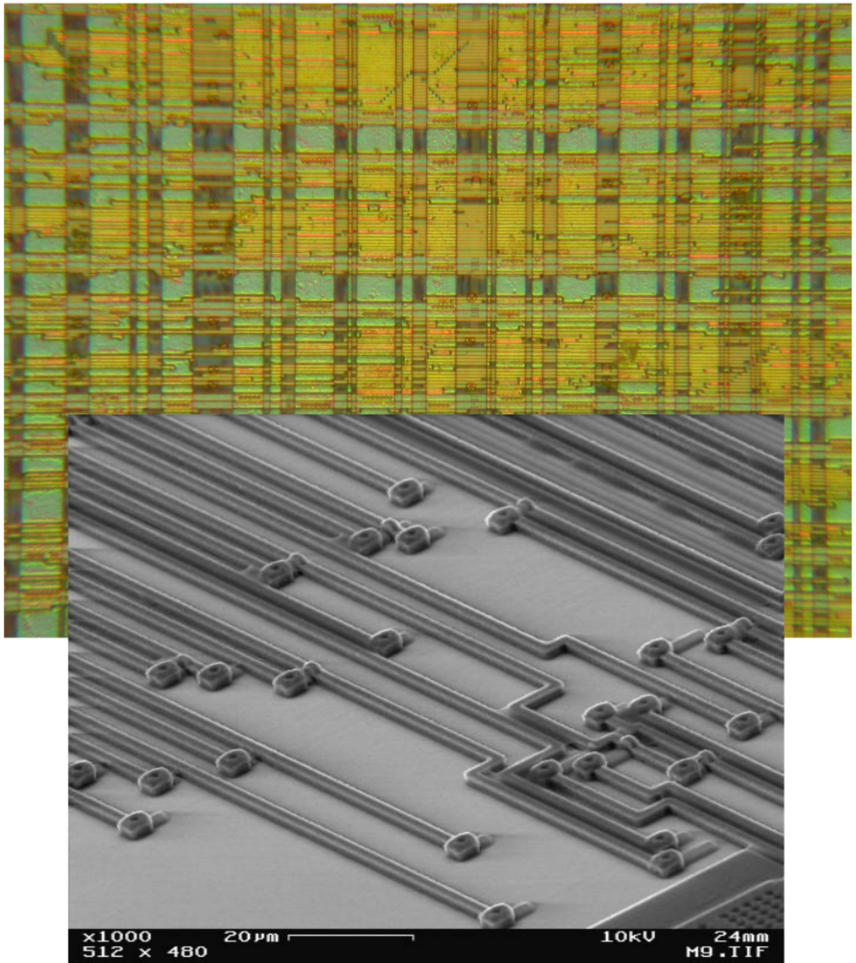
SPECULATIVE EXECUTION STRIKES AGAIN —

Intel SGX is vulnerable to an unfixable flaw that can steal crypto keys and more

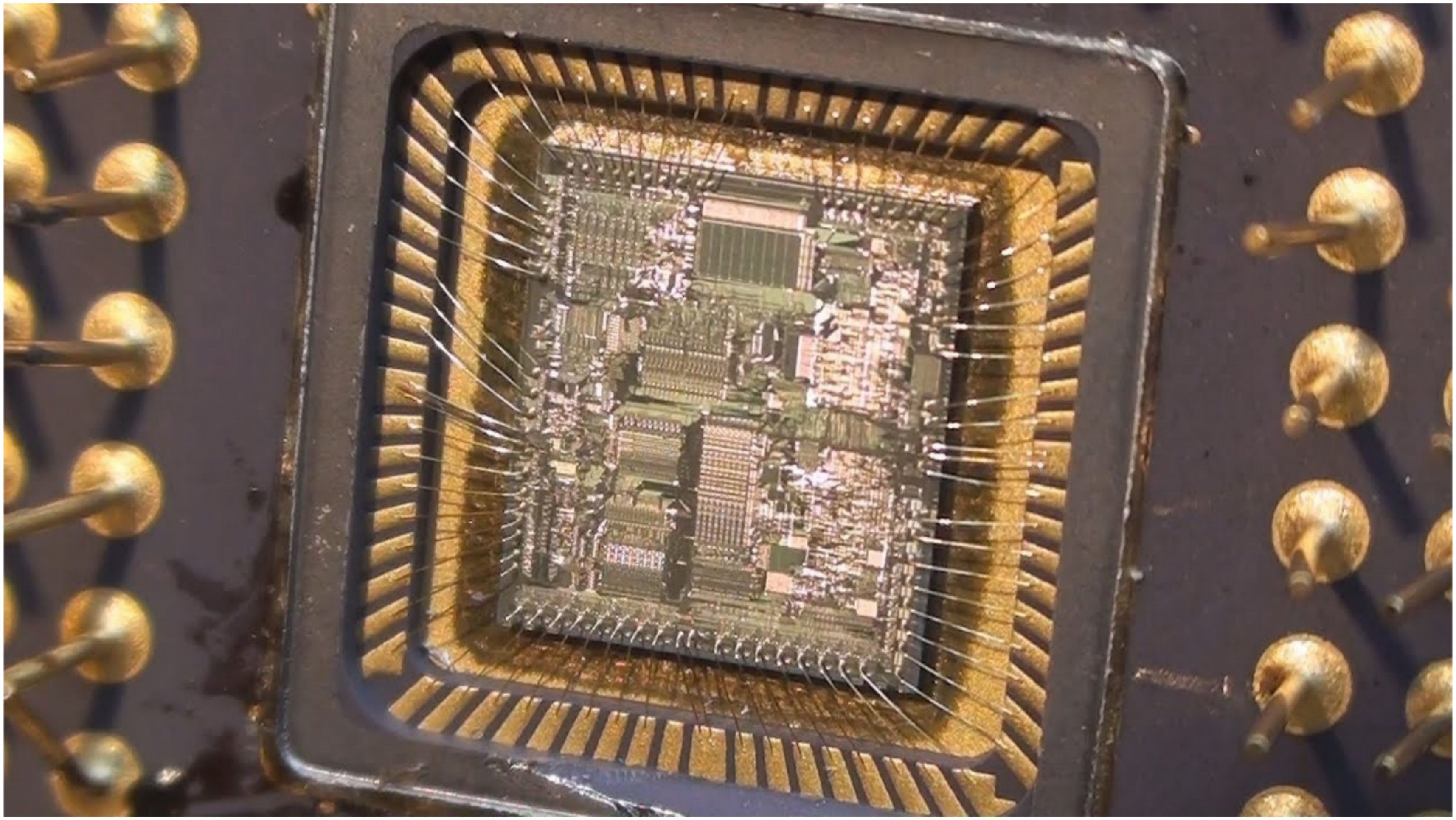
Just when you thought it was secure again, Intel's digital vault falls to a new attack.

DAN GOODIN - 3/10/2020, 5:40 PM

Let's Connect to a Server



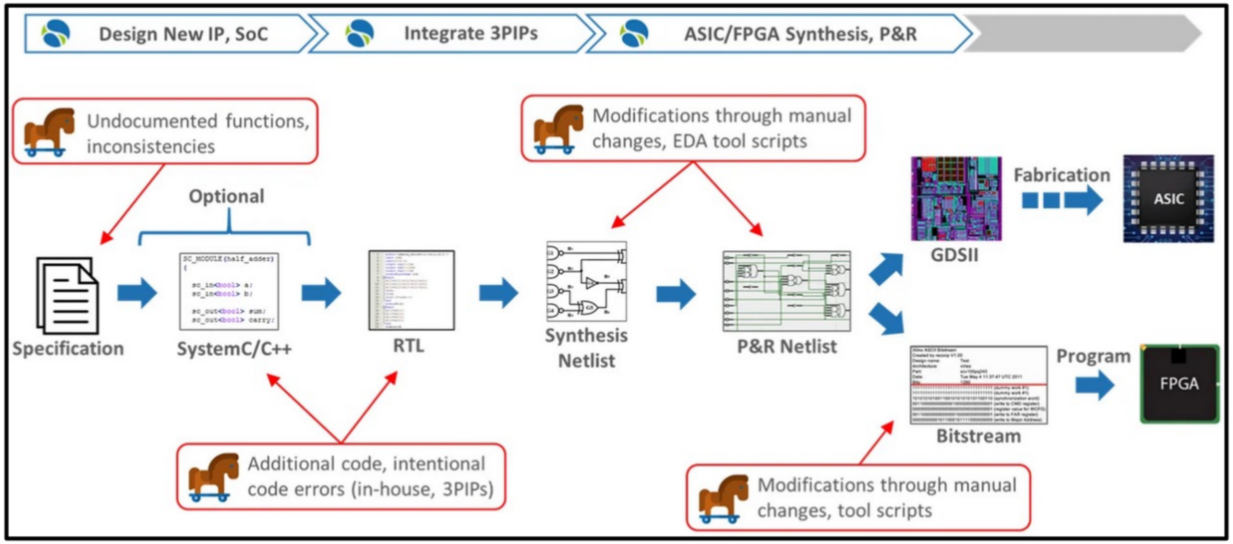
- Bash Environment
- SSH protocol design
- SSH app architecture
- SSH app logic
- Crypto math
- Crypto implementation
- Channel design
- Channel implementation
- OS implementation
- HW components
- SoC design/impl.
- CPU design/arch
- Sub-component impl.



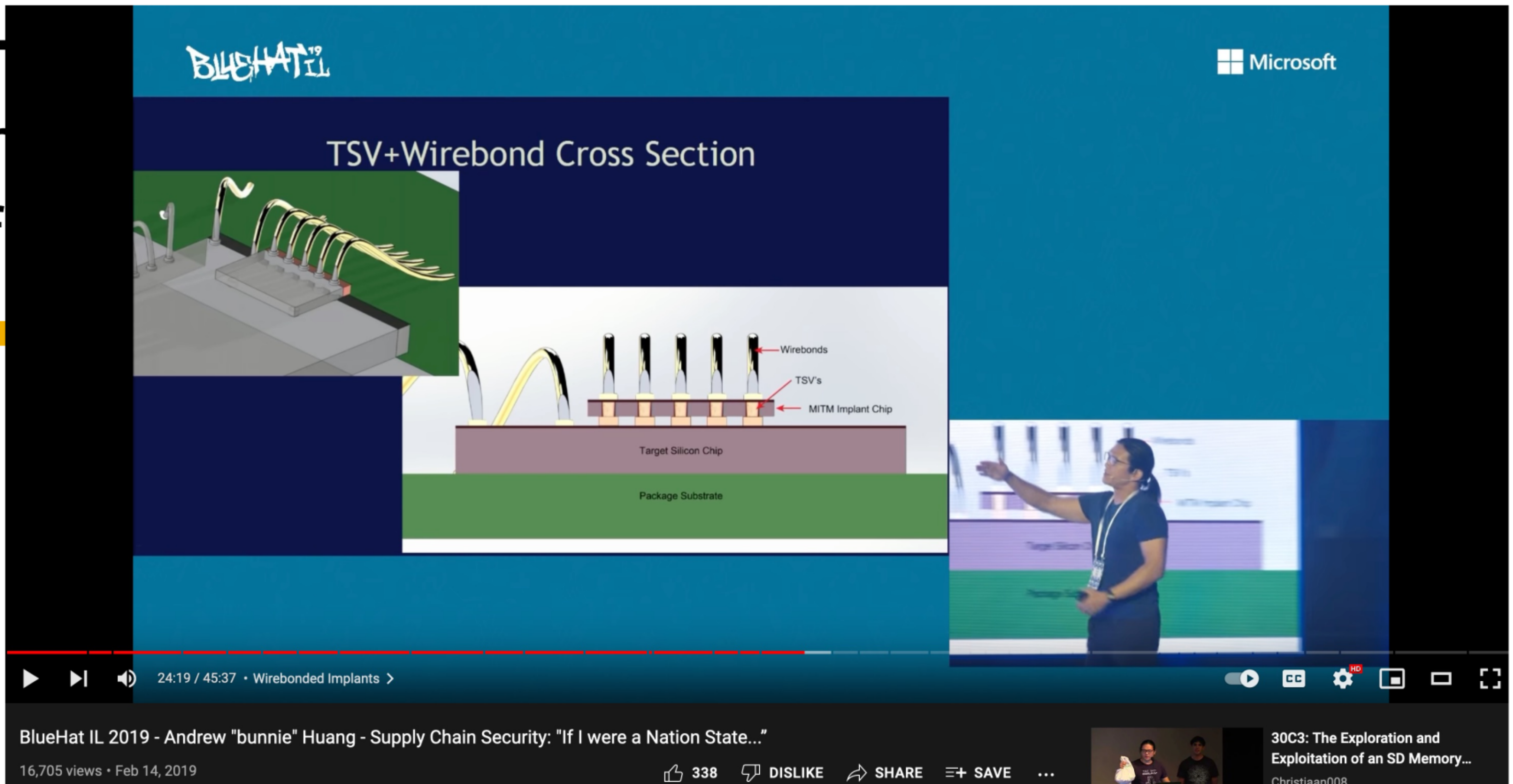
Hardware Trojans

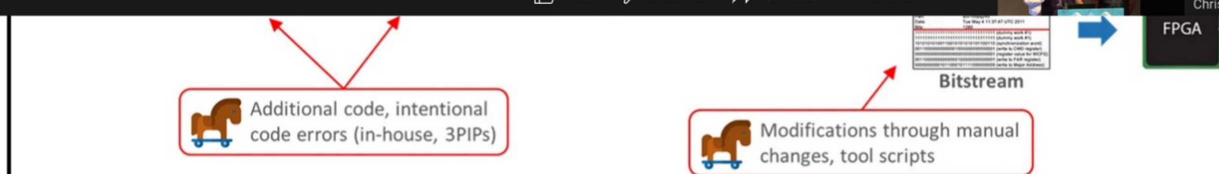
Hardware trojans are just like trojan horse malware except implemented in hardware from the manufacturer.

- Known capability but never seen (AFAIK)

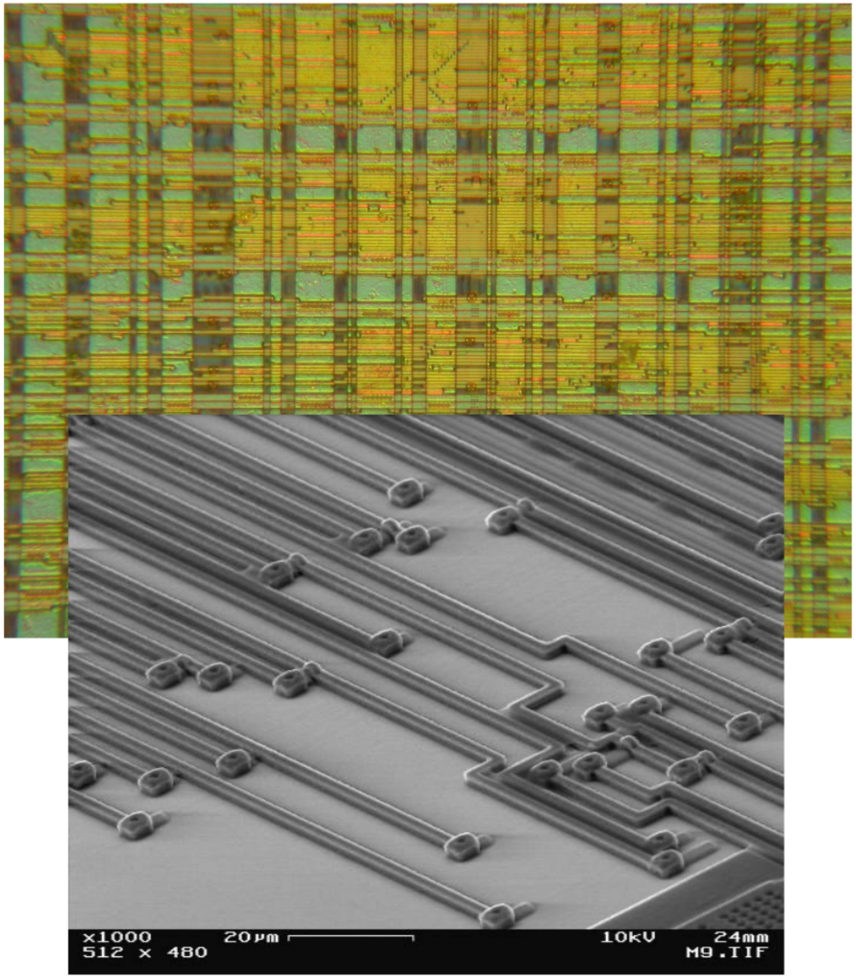


Hardware Trojans



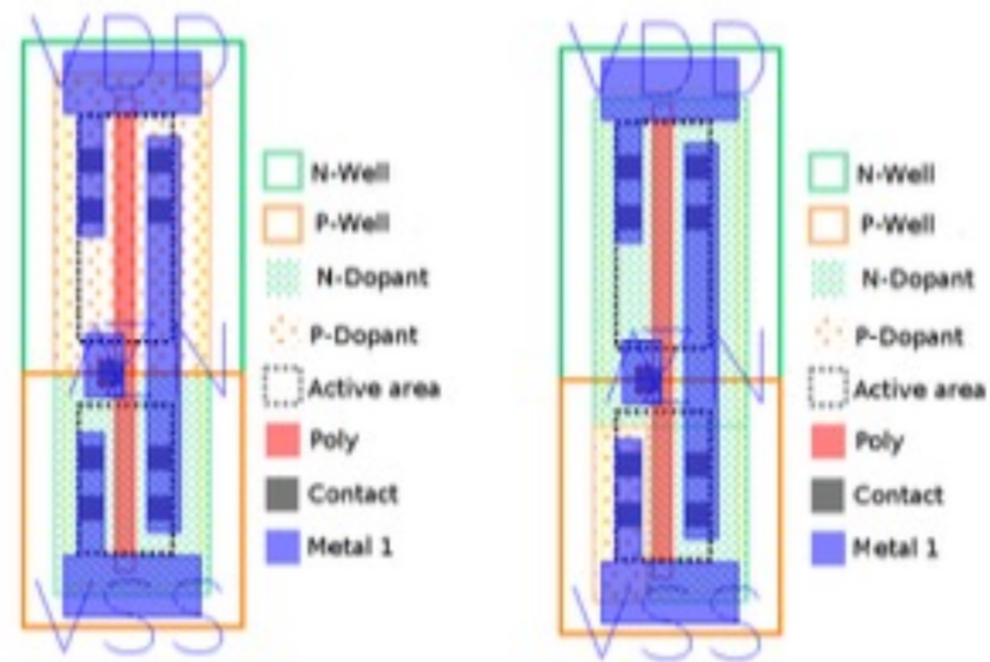


Let’s Connect to a Server



- Bash Environment
- SSH protocol design
- SSH app architecture
- SSH app logic
- Crypto math
- Crypto implementation
- Channel design
- Channel implementation
- OS implementation
- HW components
- SoC design/impl.
- CPU design/arch
- Sub-component impl.
- Silicon traces

Stealthy Dopant Trojans



(a) Original (b) Trojan

Fig. 1. Figure of an unmodified inverter gate (a) and of a Trojan inverter gate with a constant output of V_{DD} (b).

Stealthy Dopant Trojans

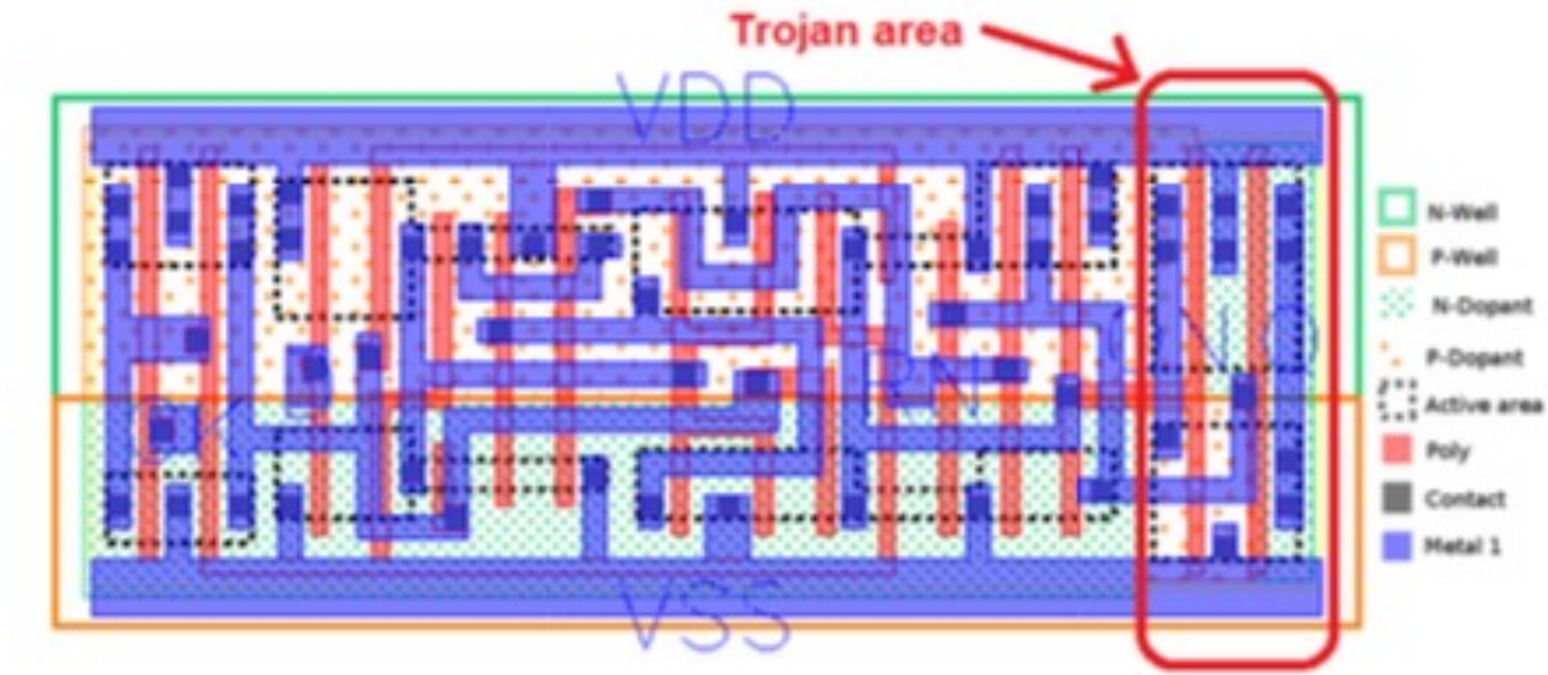
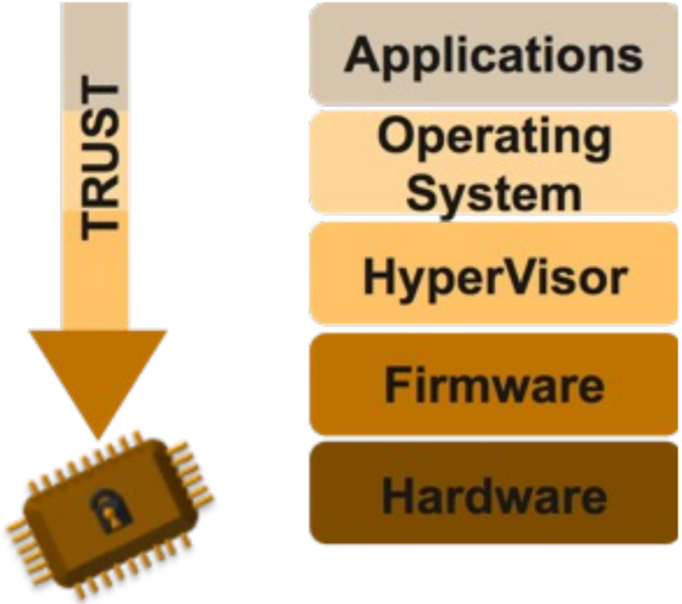


Fig. 2. Layout of the Trojan DFFR_X1 gate. The gate is only modified in the highlighted area by changing the dopant mask. The resulting Trojan gate has an output of $Q = V_{DD}$ and $QN = GND$.

Let's Connect to a Server

Trusted Computing Base

The **Trusted Computing Base (TCB)** is the collection of all components within a system critical to providing security properties.



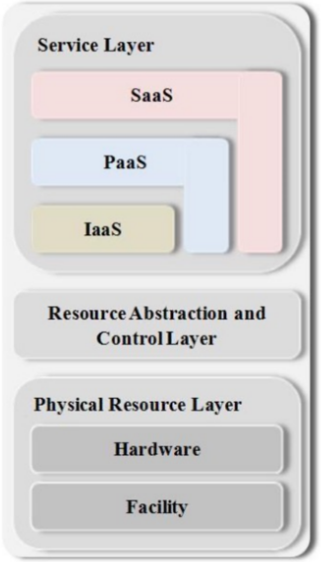


Figure 15: Cloud Provider - Service Orchestration

CPU Instructions

```
section     .text
global      _start                              ;must be declared for linker (ld)

_start:                                         ;tell linker entry point

    mov     edx,len                             ;message length
    mov     ecx,msg                             ;message to write
    mov     ebx,1                               ;file descriptor (stdout)
    mov     eax,4                               ;system call number (sys_write)
    int     0x80                                ;call kernel

    mov     eax,1                               ;system call number (sys_exit)
    int     0x80                                ;call kernel

section     .data

msg     db  'Hello, world!',0xa                 ;our dear string
len     equ $ - msg                             ;length of our dear string
```

- Low-level commands passed to CPU
- Carried out via physical gates
  - AND, OR, NOT, etc
  - Latches, Counters, Adders, etc

Security CPU Instructions

AES-NI

- Perform specific AES operation in HW
  - Encrypt/Decrypt round
  - Generate round key

Instruction	Description[2]
AESENC	Perform one round of an AES encryption flow
AESENCLAST	Perform the last round of an AES encryption flow
AESDEC	Perform one round of an AES decryption flow
AESDECLAST	Perform the last round of an AES decryption flow
AESKEYGENASSIST	Assist in AES round key generation
AESIMC	Assist in AES Inverse Mix Columns
PCLMULQDQ	Carryless multiply (CLMUL)[3]

Security CPU Instructions

AES-NI

- Perform specific AES operation in HW
  - Encrypt/Decrypt round
  - Generate round key

Instruction	Description[2]
AESENC	Perform one round of an AES encryption flow
AESENCLAST	Perform the last round of an AES encryption flow
AESDEC	Perform one round of an AES decryption flow
AESDECLAST	Perform the last round of an AES decryption flow
AESKEYGENASSIST	Assist in AES round key generation
AESIMC	Assist in AES Inverse Mix Columns
PCLMULQDQ	Carryless multiply (CLMUL)[3]

RDRAND

- Read random value from HW and store in given register

RDRAND—Read Random Number

Opcode*/ Instruction
NFx 0F C7 /6 RDRAND r16
NFx 0F C7 /6 RDRAND r32
NFx REX.W + 0F C7 /6 RDRAND r64

RDRAND Leakage

Special Register Buffer Data Sampling (SRBDS) Hardware Vulnerability in Intel CPUs (CVE-2020-0543, aka Crosstalk)

It was discovered that special register buffer data of certain Intel CPUs may be exposed to a malicious process executing on the same CPU. Particular processor operations (e.g RDRAND, RDSEED) use data from outside the physical processor core - this can be done via an internal microarchitectural operation called a special register read. This uses part of a shared staging buffer which may not be completely zero'd on subsequent uses by other threads. As such, a local attacker may be able to infer stale values which were previously returned from special register reads to other processors (as their contents may still be present in other parts of the shared staging buffer). Some special register reads return sensitive information (such as RDRAND, RDSEED and SGX EGETKEY) and so an attacker executing code on the same CPU may be able to infer these values for another thread / process executing on the same CPU.

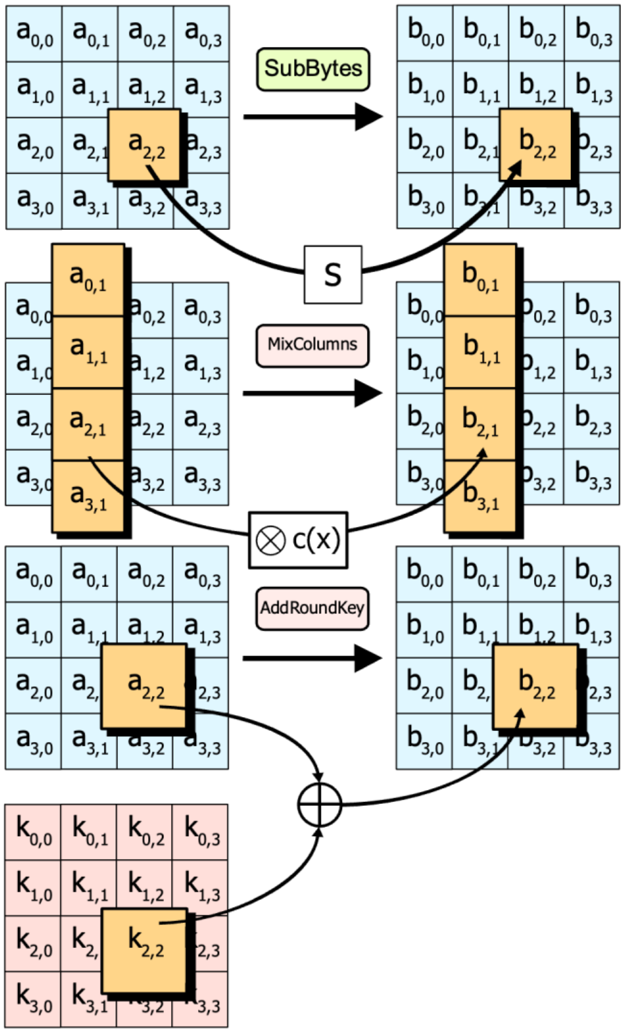
This attack relies on the same techniques used to exploit previous microarchitectural speculative execution vulnerabilities such as MDS and TSX Asynchronous Abort, and affects some client and Intel® Xeon® E3 processors; it does not affect other Intel Xeon or Intel Atom® processors.

Mitigations for this issue are provided by CPU microcode updates via the `intel-microcode` package. This mitigation consists of ensuring data in the shared staging buffer is overwritten by the RDRAND, RDSEED and EGETKEY instructions and serialising execution of RDRAND etc instructions across multiple logical processors. As a result, this will have an effect on the performance of these instructions. In conjunction, the microcode updates also supports an opt-out mechanism so that performance can be restored if desired - to support this, the Linux kernel supports a kernel command-line option `srbds=off` to allow system administrators to make use of the opt-out mechanism. For details on the boot command line option and how to check system status, please see the Linux Kernel Special Register Buffer Data Sampling Admin Guide.

Oversimplified Descriptions

- **Security CPU Instructions**
  - Trusted actions in standard hardware

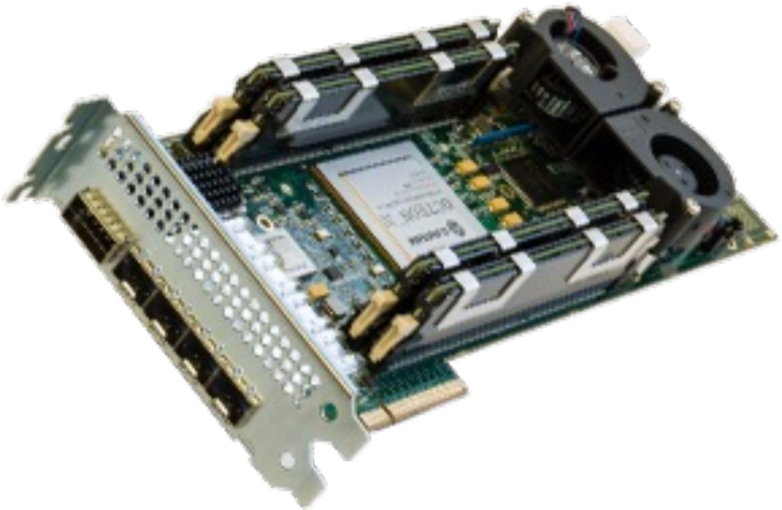
Cryptographic Accelerator



- Operations are often well-defined and repetitive
  - 14-rounds for AES256
  - Trial-and-error for bitcoin mining
  - Standardized protocols
- ASIC allows optimized pipelines for specific behavior

Cryptographic Accelerator

A **Cryptographic Accelerator** is an add-on component that allows software to leverage custom ASICs for improved performance.



Usage: Crypto Accelerator

Primitive-Level Variant

- Offload actions
- Software provides:
  - Action-specific input
    - CT/PT/data/sig+data
  - Instance-specific secret
- Accelerator provides
  - Primitive algorithms
  - Action-specific output

Protocol-Level Variant

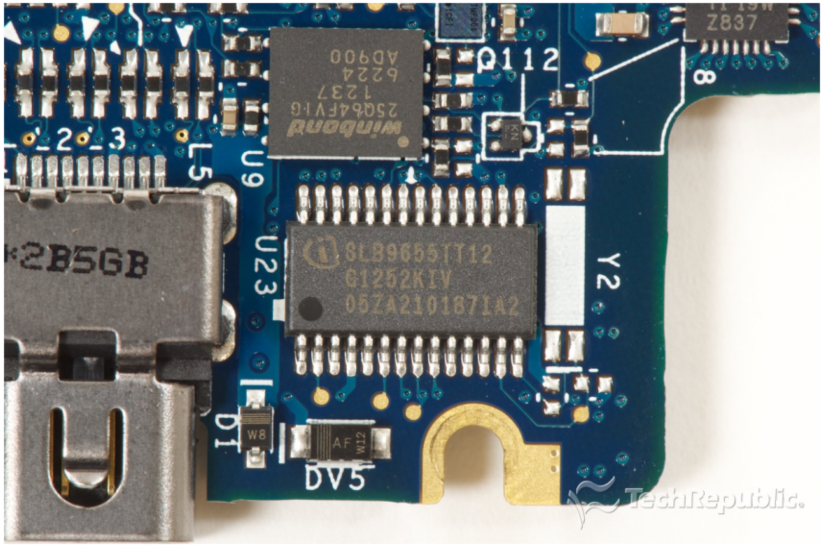
- Offload layers
- Software provides:
  - Configuration
  - Long-term secrets
- Accelerator provides:
  - Protocols negotiation
  - Primitive algorithms
  - Short-term secrets
  - Plaintext messages

Oversimplified Descriptions

- **Security CPU Instructions**
  - Trusted actions in standard hardware
- **Crypto Accelerator**
  - Fast, trusted actions in add-on hardware

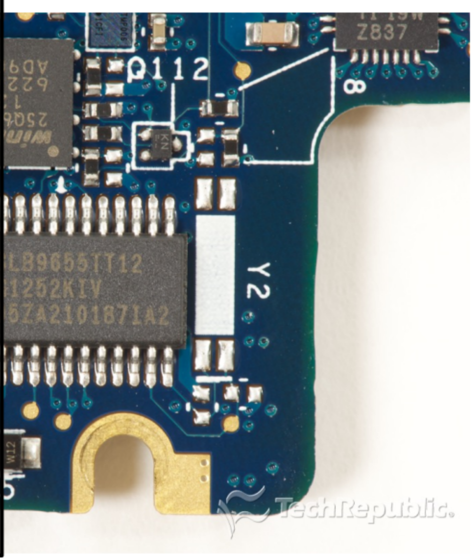
Trusted Platform Module

A **Trusted Platform Module (TPM)** is additional built-in, self-contained ASIC that provides a central “root of trust” for a device.

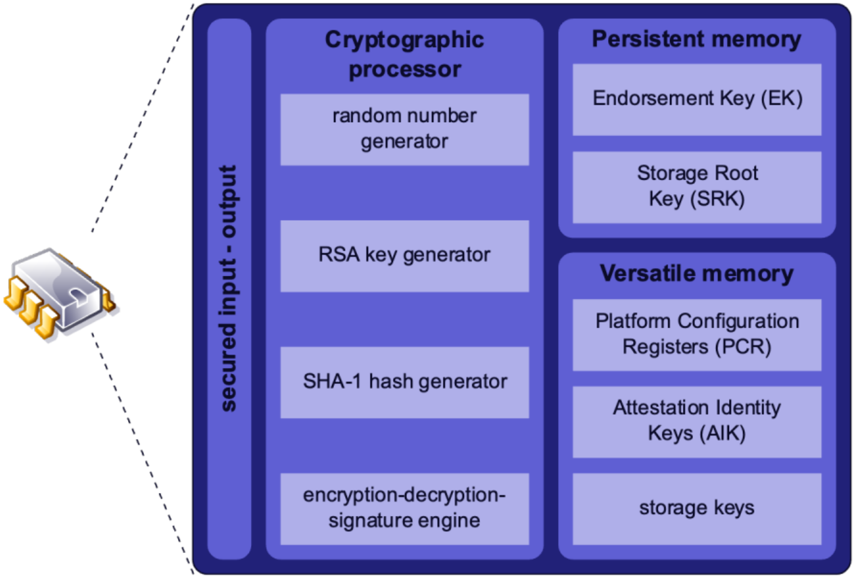


Trusted Platform Module

A **Trusted Platform Module (TPM)** is additional built-in, self-contained ASIC that provides a central "root of trust" for a device.

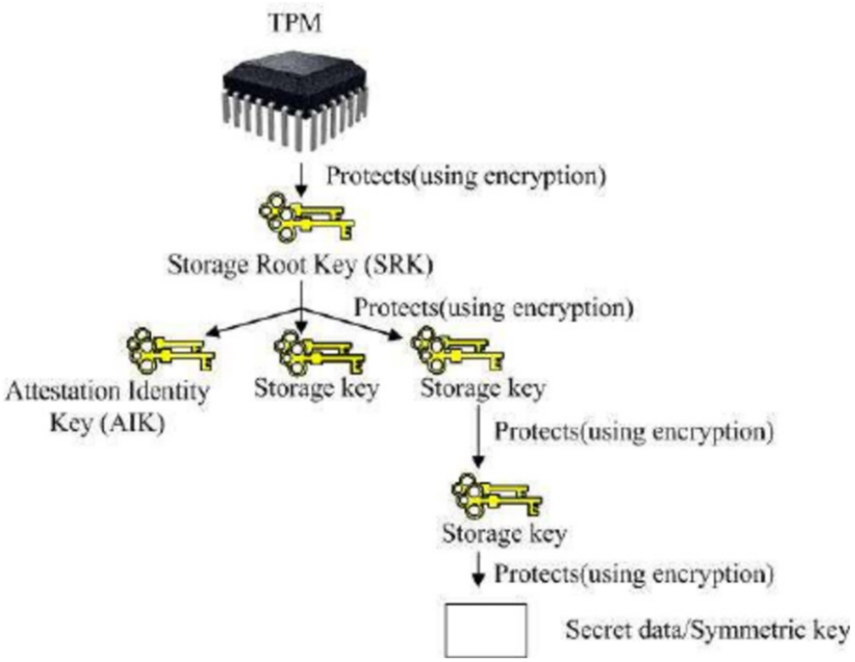


TPM Internals



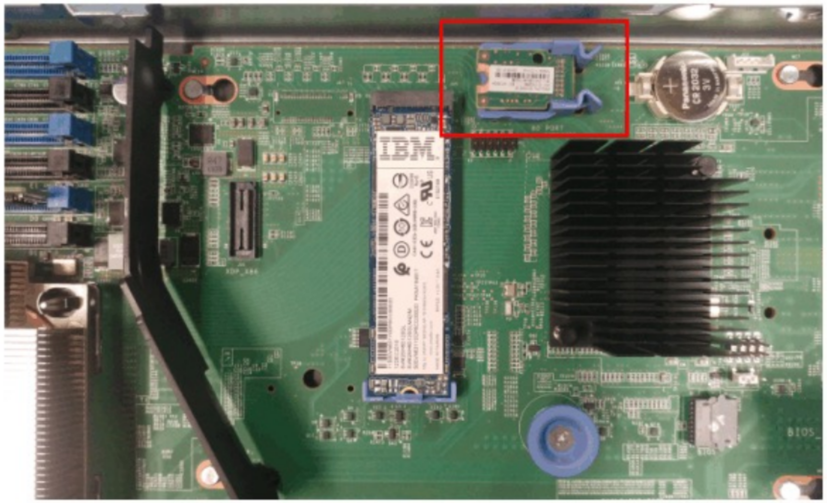
- Includes suite of crypto primitives
  - RNG
  - Algorithm implementations
  - Secure storage
- Arbitrary control logic
  - Timers, persistent counters, etc

TPM Keys



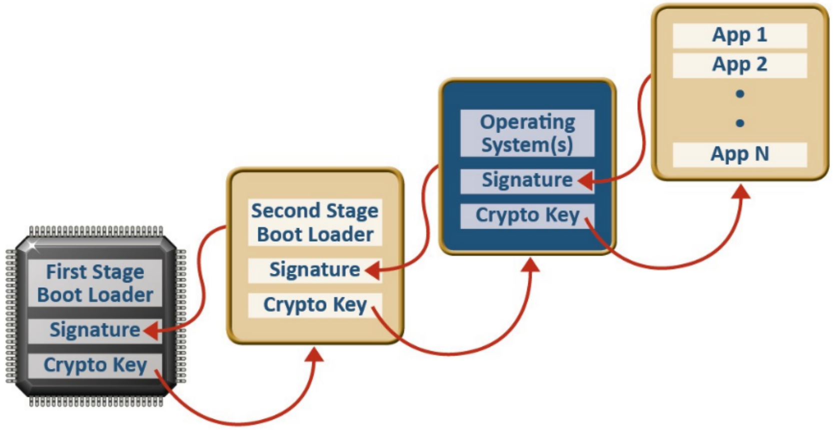
- Secrets are either generated on-board or injected
- Derive many secrets from single root secret via KDF

Usage: Building Block



- Most commonly a component on motherboard
- Software treats as black-box operations
  - Hardened, well-defined interface for use

Example: Secure Boot



- TPM validates firmware signature before booting
- If invalid, refuse to launch bootloader
- Used as foundational trust for validating higher-level software

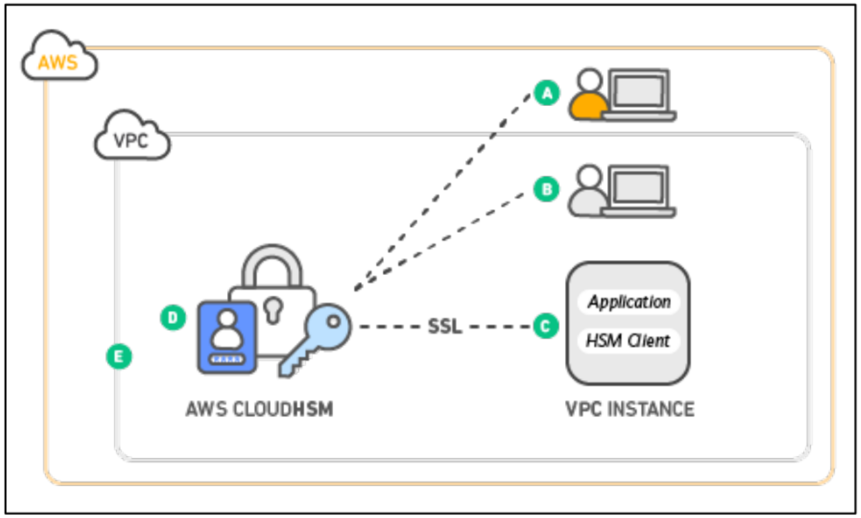
Oversimplified Descriptions

- **Security CPU Instructions**
  - Trusted actions in standard hardware
- **Crypto Accelerator**
  - Fast, trusted actions in add-on hardware
- **Trusted Platform Module**
  - Trusted actions in built-in hardware w/ keys

Hardware Security Module

A **Hardware Security Module (HSM)** is a special-purpose add-on component that securely stores cryptographic keys and performs cryptographic operations.

Hardware Security Module



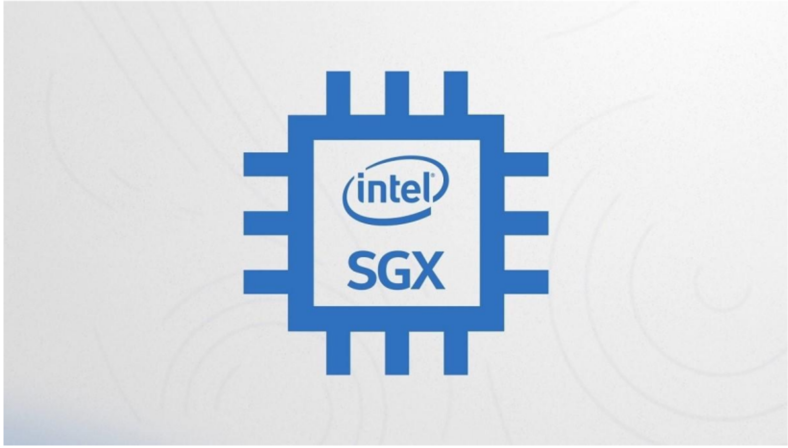
- High-performance operations
- Restricted logic
  - Most commonly used for signing operations
- Commonly available “in the cloud” for use with AWS/GCP/…

Oversimplified Descriptions

- **Security CPU Instructions**
  - Trusted actions in standard hardware
- **Crypto Accelerator**
  - Fast, trusted actions in add-on hardware
- **Trusted Platform Module**
  - Trusted actions in built-in hardware w/ keys
- **Hardware Security Module**
  - Fast, trusted actions in add-on hardware w/ keys

Trusted Execution Environment

A **Trusted Execution Environment (TEE)** is a general computation environment that provides additional security properties such as access to keys, memory encryption, etc.

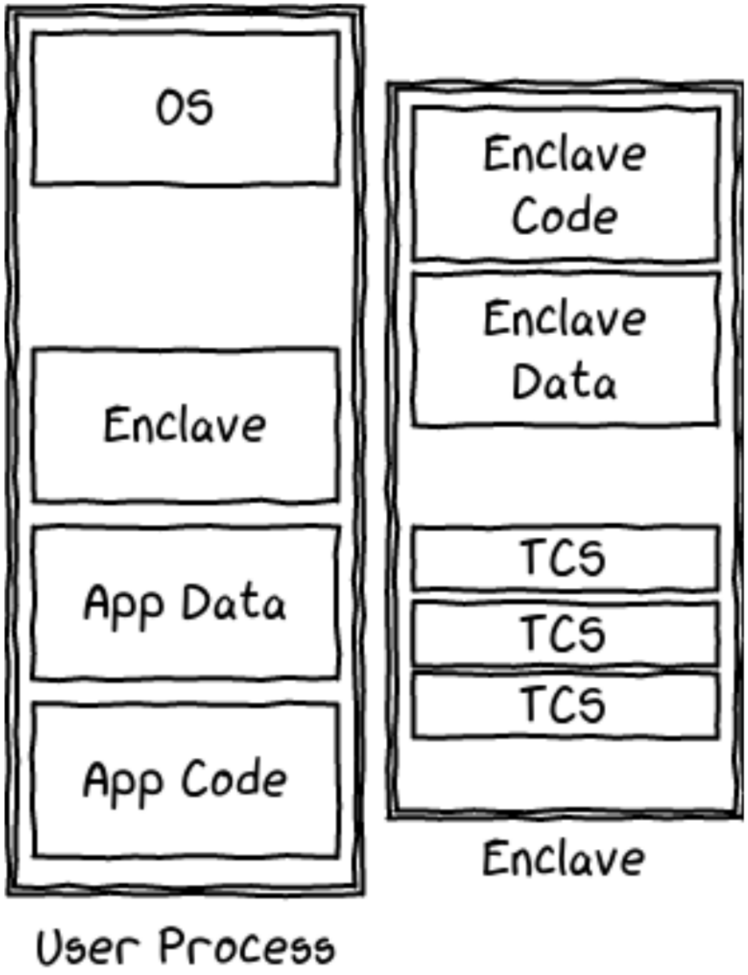


Enclave Logic

```c
/*
 * Return a SHA256 hash of the requested key. KEYS SHOULD NEVER BE
 * SENT OUTSIDE THE ENCLAVE IN PLAIN TEXT. This function let's us
 * get proof of possession of the key without exposing it to untrusted
 * memory.
 */

sgx_status_t enclave_ra_get_key_hash(sgx_status_t *get_keys_ret,
	sgx_ra_context_t ctx, sgx_ra_key_type_t type, sgx_sha256_hash_t *hash)
{
	sgx_status_t sha_ret;
	sgx_ra_key_128_t k;

	// First get the requested key which is one of:
	//  * SGX_RA_KEY_MK
	//  * SGX_RA_KEY_SK
	// per sgx_ra_get_keys().

	*get_keys_ret= sgx_ra_get_keys(ctx, type, &k);
	if ( *get_keys_ret != SGX_SUCCESS ) return *get_keys_ret;

	/* Now generate a SHA hash */

	sha_ret= sgx_sha256_msg((const uint8_t *) &k, sizeof(k),
		(sgx_sha256_hash_t *) hash); // Sigh.

	/* Let's be thorough */

	memset(k, 0, sizeof(k));

	return sha_ret;
}
```

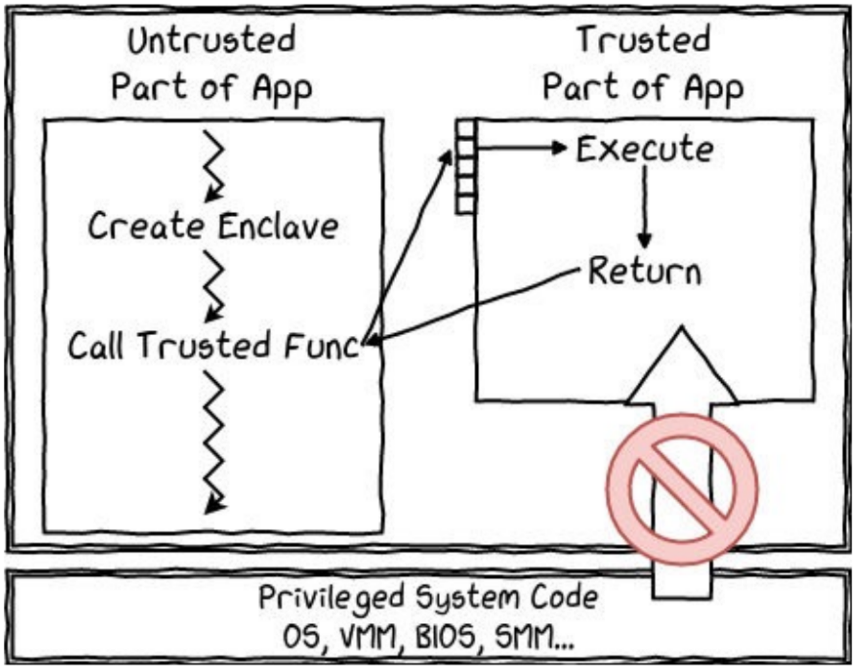
- Allow arbitrary logic from developers
- Extremely small TEE standard library
- Statically compile against TEE standard library & dependencies

TEE Application



- Enclave logic is compiled and signed with enclave key
- Enclave blob compiled into application binary
- Application binary behaves as normal

TEE Operation



- App loads enclave blob via OS driver
- App can make function calls into the enclave
- Enclave executes outside the influence of OS, apps, etc

TEEs in the Real-World

ars TECHNICA SUBSCRIBE SIGN IN

RAIDING FORT KNOX —

SGX, Intel's supposedly impregnable data fortress, has been breached yet again

ÆPIC Leak spills users' most sensitive secrets in seconds from SGX enclaves.

DAN GOODIN - 8/9/2022, 12:01 PM

Enlarge

BLEEPINGCOMPUTER

Home › News › Security › New Intel chips won't play Blu-ray disks due to SGX deprecation

New Intel chips won't play Blu-ray disks due to SGX deprecation

By Bill Toulas

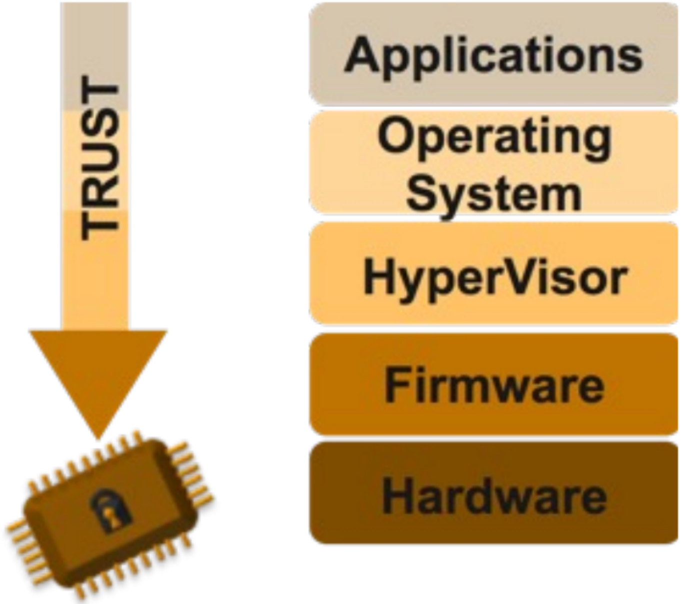
January 14, 2022 11:46 AM 12

Oversimplified Descriptions

- **Security CPU Instructions**
  - Trusted actions in standard hardware
- **Crypto Accelerator**
  - Fast, trusted actions in add-on hardware
- **Trusted Platform Module**
  - Trusted actions in built-in hardware w/ keys
- **Hardware Security Module**
  - Fast, trusted actions in add-on hardware w/ keys
- **Trusted Execution Environment**
  - Fast(ish), trusted logic in common hardware w/ keys

Trusted Computing Base

The **Trusted Computing Base (TCB)** is the collection of all components within a system critical to providing security properties.



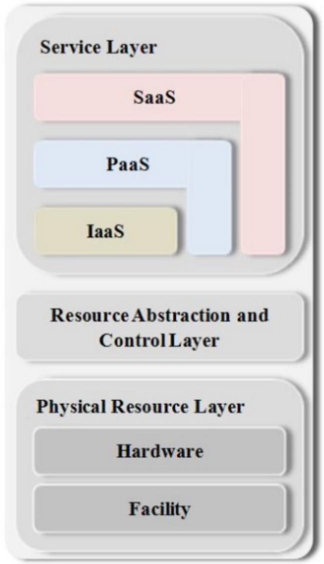
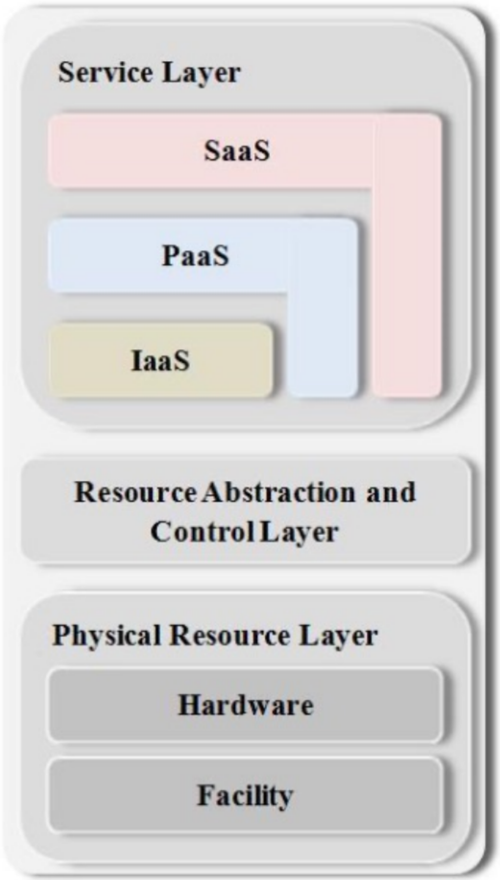


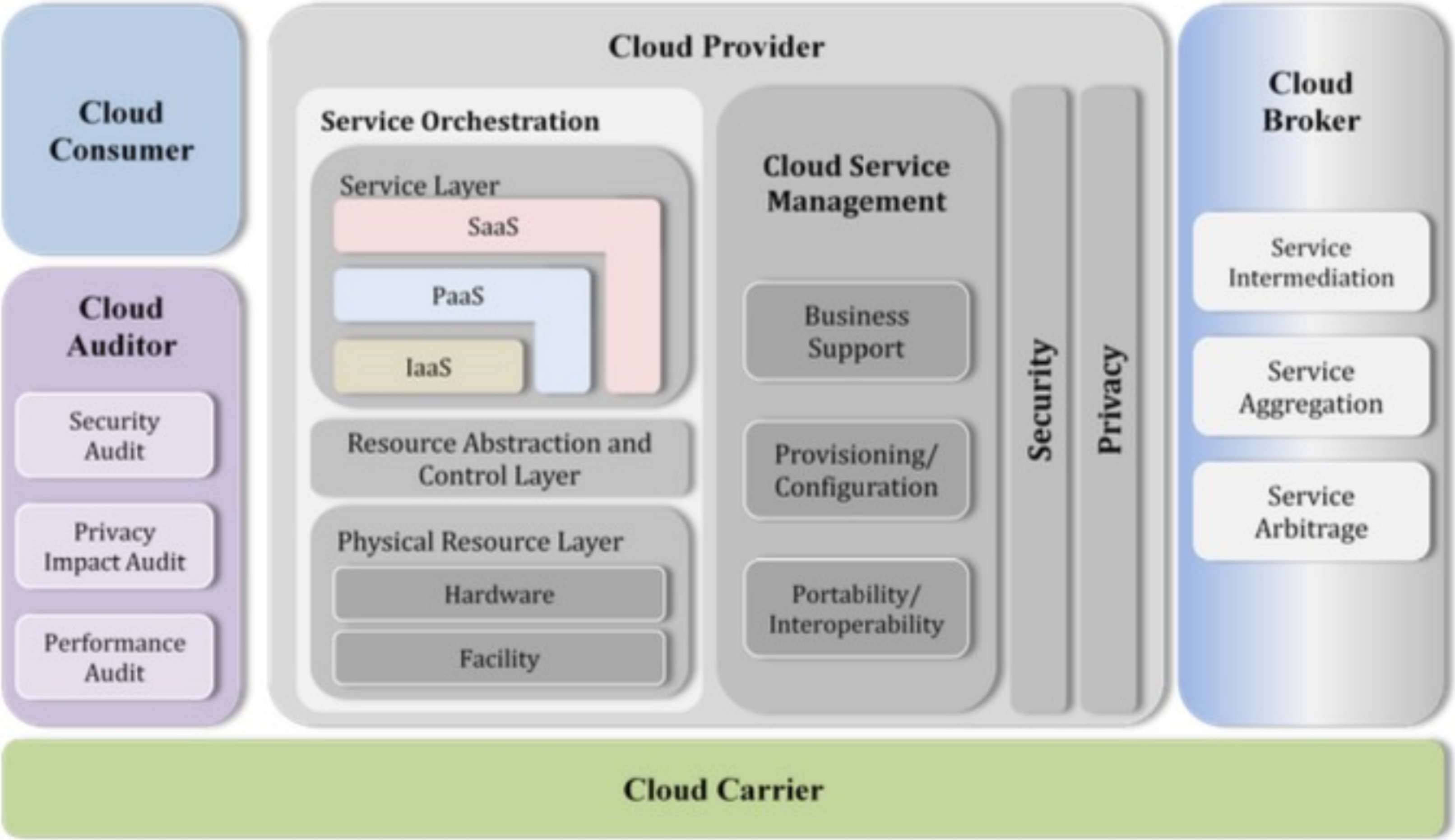
Figure 15: Cloud Provider - Service Orchestration

Cloud Provider TCB



- SaaS: Software
  - Office 365
- PaaS: Platform
  - Elastic Container Service
- IaaS: Infrastructure
  - EC2 Instances
- <many more layers of internal services>
- **All on top of Local TCB**

Cloud Computing Architecture TCB



Computer and Network Security

Lecture 15:
Hardware Security/Attacks

COMP-5370/6370
Fall 2024

Evil Maid Attacks

An **Evil Maid Attack** is where an attacker gains temporary physical access to the target's device to exploit or prepare for exploitation.

- Image storage for future forensics
- Completely replace and proxy interaction
- Install firmware-level malware
- Install malicious HW components