Security and security controls in operating systems

A *quantitative* approach
2023-02-27

Robert Malmgren
rom@romab.com

1 minute presentation

- Consultant in IT and infosec since 20+ years
- Working alot on with critical infrastrucutre protection, process control, SCADA security etc, but also in financial sector, government, etc
- Work covers everything from writing policies, requirement specs and steering documents to development, penetration testing, incident handling and forensics

Outline of talk

- Intro
- Background and basics
- Security problems & vulnerabilities
- Example of operating systems and security
- Trends

Some short notes

- The focus is on general operating system used in general computers - COTS products
  - *Embedded systems, code for micro controllers*, etc often lack most fundamental security features
  - Some experimenal OS's and domain specific solutions have better-than-average security concepts and security controls, e.g. military grade usage

Background and basics

Part 1: protection, security controls

Intro - foundation

- Modern software is normally formed into components, parts and layers in *systems*
- Layers and isolation is a way to provide separation, which can be:
  - Logical/ Virtual: A way to make it appear that execution environment have exclusive access
  - Physical: Different computers, different CPUs/cores, different disks
  - Time based: Separation of execution time/Timeshare
  - Based on security technologies, i.e. cryptographic

Intro - foundation

- Complex systems
  - …run multiple programs at once,
  - …have multiple users,
  - …store huge amounts of data,
  - …is interconnected via networks

Intro - foundation

- This there is to built-in security into the foundation of the systems - the operating system
  - To identify and authorize users of the system
  - To allow for an environment where necessary basic controls are in place
  - To prevent unauthorised access to OS resources

Some concepts and principles

TCB - *Trusted Computing Base*

RBAC - *Role Based Access Control*

MAC - *Mandatory Access Control*

Principle of least privilege

DAC - *Discretionary Access Control*

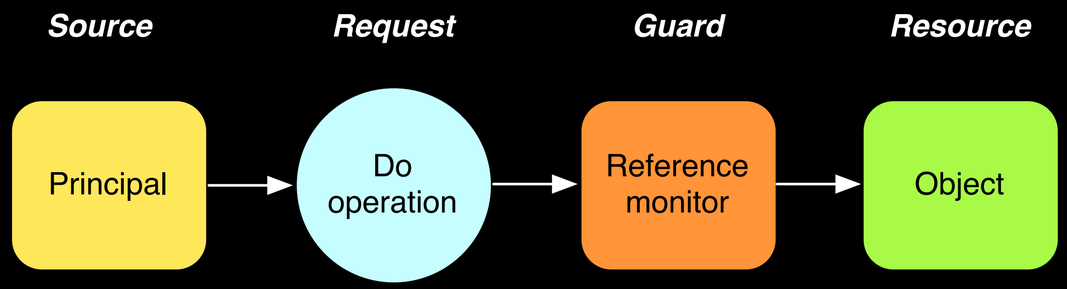
Principle of least surprise

Capabilities and requirements

Need	Description	Example
Protect a system resource	*Prohibit **malicious** or **unintentional** access to*	System tables, direct access to I/O-units, memory protection
Authorization checks for usage of system calls and system resources	*Provide controlled access to system, so that system mainain system integrity and provide continuous security to application and information*	reference monitor
Separation of resources	*Physical, Logical, temporal or cryptographical separation*	separation in running time

Some important concept

- Reference monitor



- Trusted Computing Base, TCB
  - All things in the trusted part of the OS necessary to enforce the *security policy*

[1] Lampson et al: *Authentication in Distributed Systems: Theory and Practice*

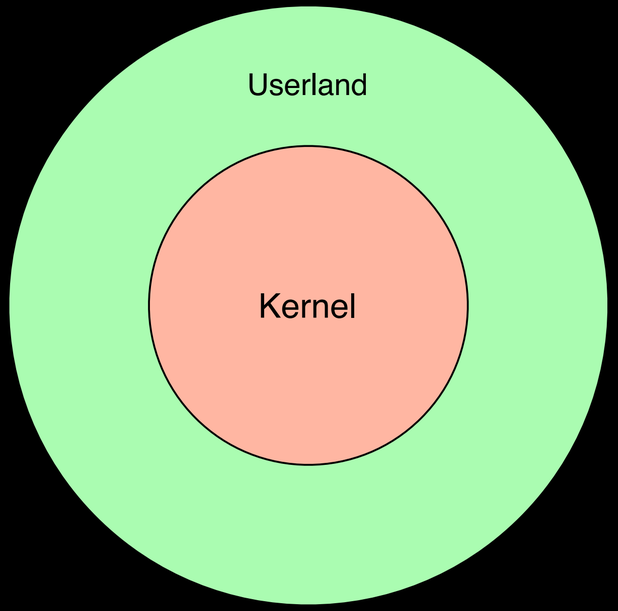
Principles for secure design*

Economy of mechanism	Keep the design as simple and small as possible
Fail-safe defaults	Base access decisions on permission rather than exclusion
Complete mediation	*Every access* to *every object* must be checked for authority
Open design	The design should **not** be secret
Separation of privilege	technique in which a program is divided into parts which are limited to the specific privileges they require in order to perform a specific task
Least privilege	Every program and every user of the system should operate using the least set of privileges necessary to complete the job
Least common mechanism	Minimize the amount of mechanism common to more than one user and depended on by all users
Psychological acceptability	It is essential that the human interface be designed for ease of use, so that users routinely and automatically apply the protection mechanisms correctly

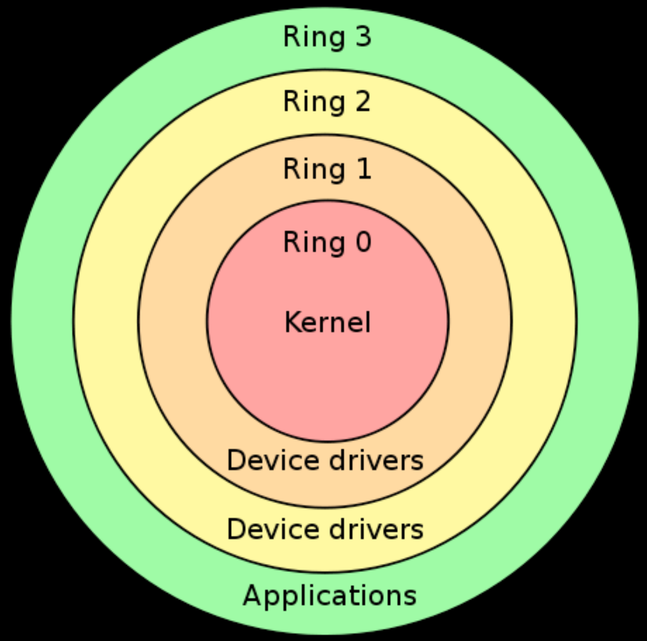
JEROME H. SALTZER et al *The Protection of Information in Computer Systems* http://www.cs.virginia.edu/~evans/cs551/saltzer/

The classical *ring model*

UNIX



x86

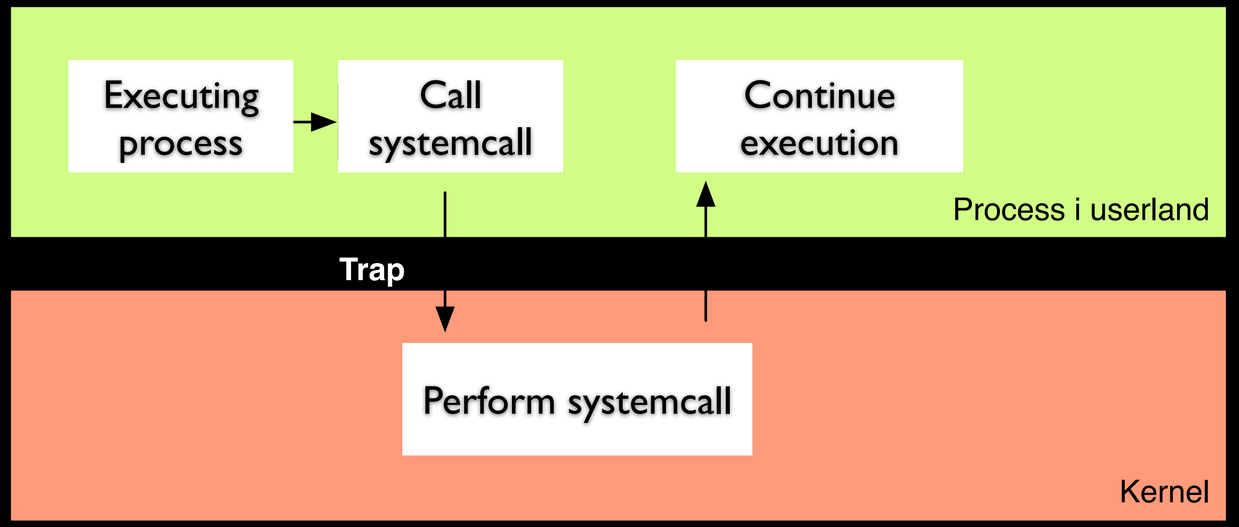


Least privileges

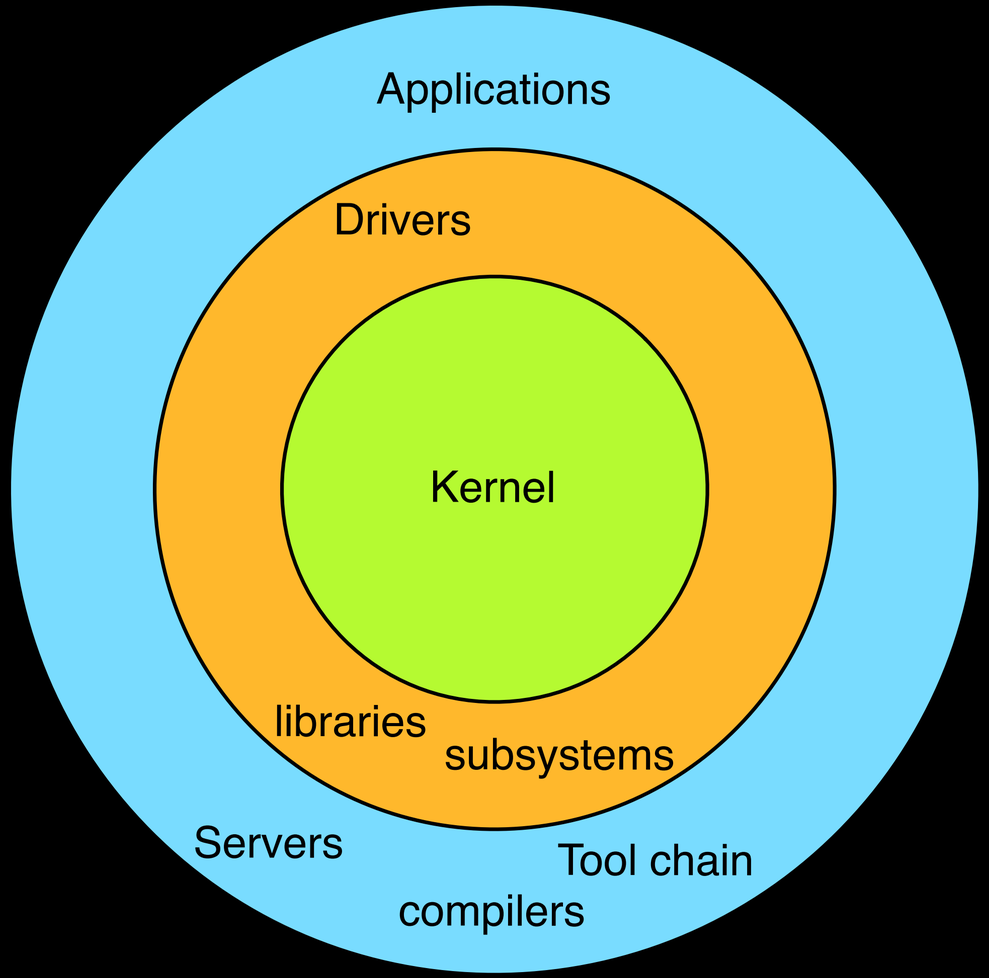
Highes privileges

Källa http://en.wikipedia.org/wiki/File:Priv_rings.svg

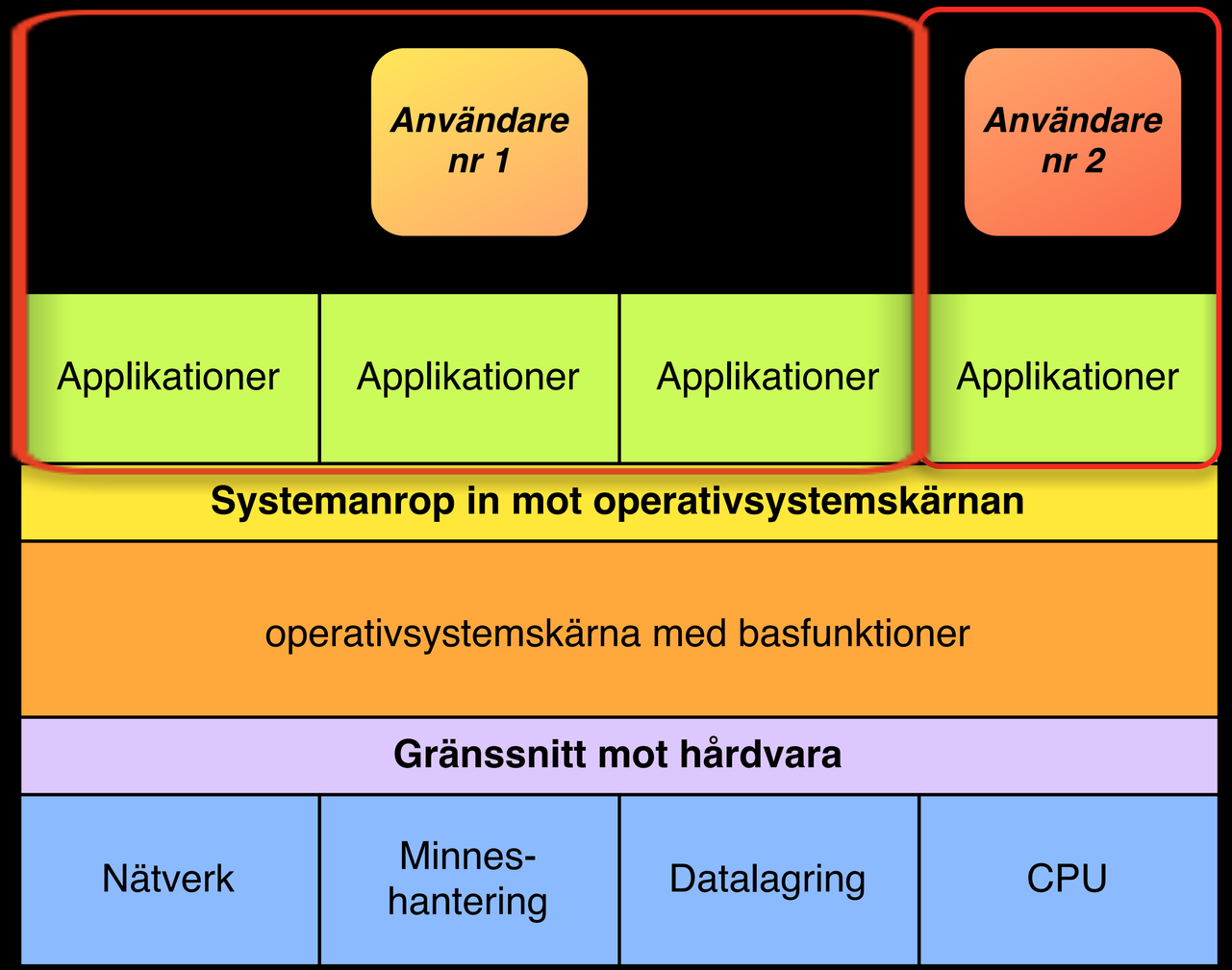
Interaction between application and OS



Overview of operating system (1/2)



Overview of operating system (2/2)



Problem with these pictures and concepts

- Layering violation
  - some software might skip a layer and call an underlaying layer directly and hence bypass controls

- In some scenarios attackers might come an unexpected way
  - Attacking from host operating system against guest operating systems in a virtual machine environment

The classical *ring model, updated*

Other rings

-1	Hypervisor	Allow guest OS ”ring 0”
-2	System Management Mode	APM/ACPI/TPM-support
-3	*Intel* Management Engine / *AMD* Platform Security Processor	Special software running in the Platform Controller Hub (PCH) processor

Källa: http://en.wikipedia.org/wiki/File:Priv_rings.svg

Källa: https://medium.com/swlh/negative-rings-in-intel-architecture-the-security-threats-youve-probably-never-heard-of-d725a4b6f831

Problem with these concepts

- You have a "hidden" processor on your computer
- Its functionality has never been publicly documented
- It appears to have been customized for certain TLA government agencies
- It has unlimited access to the main processor
- It has unlimited access to all memory
- It has unlimited access to all peripherals
- It has its own MAC and IP addresses
- It runs a web server
- It is always running
- You can't turn it off
- You can't disable it
- It has had multiple known exploitable vulnerabilities
- It is the single most privileged known element of an Intel Architecture processor chipset

Memory handling

- RAM memory is a central resource that in a controlled way must be shared and handled *between operating system, applications and other components*

- Modern computer systems have hardware support for memory protection, e.g. **MMU**
  - OS support is required to use the hardware supported memory protection
  - Modern hardware support can enforce several security features related to isolation, non-executable memory areas, etc

File system

- A file system is often a central component in a computer system w.r.t. security and protection
- Besides the actual *file content*, there is *meta data* that is of importance
  - File owner, dates of creation/change/access, access information, security labels, etc
- Manipulation of meta data can in some cases be more serious security breach than the manipulation of the file content itself. Or a combo of both can be misleading and hide the fact that a file has been altered

Local filsystem

File system	Description	Comment
FAT	*No access control*	Classic MS-DOS
NTFS	*Discretional Access Control via* ***ACL***	Advanced possibilities to make controls
UFS	*Discretional Access Control, writing & program execution for owner, group, "others"*	Simple access controls

Network file systems

File system	Description	Comment
NFSv3	*Hostbaserad accesskontroll, uid*	Trivial to circumvent
NFSv4	*Secure RPC, KRB5a, KRB5p, KRB5i*	Require a Kerberos server, KDC a= authentication i=integrity = calculate MAC p= privacy = encrypt packet
SMB/CIFS	*KRB5a*	

Comparing security in Operating systems (1/5)

- When in time was the system developed?
  - What was the state-of-the-art at that time?
  - What trends where currently in fashion?

Comparing security in Operating systems (2/5)

"Given enough eyeballs, all bugs are shallow"
- Linus' Law

- Development methodologies
  - Open Source or Closed Source?
  - What support do one use to ensure that security is *built into the product*?
  - How does one ensure that implementation is a correct representation of the design, that is a correct interpretation of the analysis?

Comparing security in

But really, what good is this comparison?

Write more code = get higher salary?
Manage a 200K-SLOC project is *cooler* than a 5K-SLOC?

More code = more bugs?

More code = more *security checks* and *advanced concepts* like crypto, resillient failure checking built into everything?

But certainly, complexity is considered **bad** and **evil** in the context of security.
And there is often a relation between complexity and size of program

2015	Windows 10	40-60

2020	Linux kernel 5.12	28.8

http://en.wikipedia.org/wiki/Source_lines_of_code

https://informationisbeautiful.net/visualizations/million-lines-of-code/

https://sources.debian.org/stats/

Comparing security in Operating systems (5/5)

- What can one gain by having formal certification of operating systems, subsystems or application
  - Trusted Computer System Evaluation Criteria (TCSEC), Common Criteria (CC, ISO/IEC 15408), etc

- More a theoretical excersice than of any real value?

https://www.commoncriteriaportal.org/

Background and basics

Part 2: bugs and vulnerabilities

Intro - *just the basic facts*

- All software is prone to bugs
- Some bugs will have an impact that can have security implications - data leaks, destruction of data, privilege escalations

Intro - *just the basic facts*

- Some bugs help to circumvent security mechanisms
- Some *security designs* are flawed, or build on *flawed assumptions*

Operating system security

- Security problems in the operating system can affect the integrity of the system itself
  - Someone else can control the system to their own liking - ***pwnd!***
  - Bugs in OS kernel can affect system integrity
- Security problems with the operating system can in turn affect the security in *applications* and *subsystems* (databases, middle ware, etc)

http://en.wikipedia.org/wiki/Pwn

Some concepts and terms

Vulnerability

Exploit

0day exploit

Foreverday exploit

CVE-2021-1234

Stack smashing

Stack overflow

Race conditions

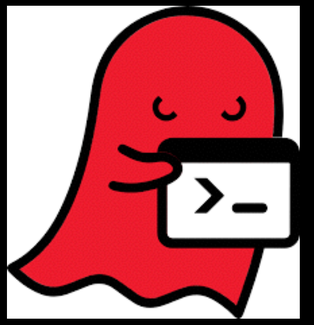
Heap overflow

Intro - *the basics*

- Some bugs are undiscovered for some time, they lay latent
- Once discovered, they can be abused, if it is an security vulnerability, that can be exploited
- A discovered security bug, is sometime called a 0day, until it is mitigated

Intro - *the basics*

- Nowdays bugs and vulnerabilities tend to get names (*heartbleed, ghost, shellshock,* etc) and logos
- also some bugs/vulnerabilities gets ”formal name”, i.e. CVE*, and a scoring CVSS**
  - e.g. CVE-2011-3172



* ”Common Vulnerabilities and Exposures;” https://cve.mitre.org/

** https://www.first.org/cvss/specification-document**

Some concepts and principles

- Attack vector - Different paths to reach an vulnerability. One path might be closed by a vendor patch, but another might still be there, if the root cause is not identified and fixed.
- Reverse engineering - To re-create the original design by observing the final result, in computer science - to re-create some source code by examing a binary.

cvedetails.com/top-50-products.php

CVE Details

The ultimate security vulnerability datasource

(e.g.: CVE-2

Log In Register

Switch to https://
Home

Browse :
Vendors
Products
Vulnerabilities By Date
Vulnerabilities By Type

Reports :
CVSS Score Report
CVSS Score Distribution

Search :
Vendor Search
Product Search
Version Search
Vulnerability Search
By Microsoft References

Top 50 :
Vendors
Vendor Cvss Scores
Products
Product Cvss Scores
Versions

Other :
Microsoft Bulletins
Bugtraq Entries
CWE Definitions
About & Contact
Feedback
CVE Help
FAQ
Articles

External Links :
NVD Website
CWE Web Site

View CVE :
Go
(e.g.: CVE-2009-1234 or 2010-1234 or 20101234)

Top 50 Products By Total Number Of "Distinct" Vulnerabilities

Go to year: 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022

	Product Name	Vendor Name	Product Type	Number of Vulnerabilities
1	Debian Linux	Debian	OS	5816
2	Android	Google	OS	4069
3	Ubuntu Linux	Canonical	OS	3115
4	Mac Os X	Apple	OS	2964
5	Linux Kernel	Linux	OS	2748
6	Fedora	Fedoraproject	OS	2728
7	Windows 10	Microsoft	OS	2590
8	Iphone Os	Apple	OS	2573
9	Windows Server 2016	Microsoft	OS	2334
10	Chrome	Google	Application	2329
11	Windows Server 2008	Microsoft	OS	2154
12	Windows 7	Microsoft	OS	2019
13	Firefox	Mozilla	Application	1993
14	Windows Server 2012	Microsoft	OS	1954
15	Windows 8.1	Microsoft	OS	1841
16	Windows Server 2019	Microsoft	OS	1792
17	Windows Rt 8.1	Microsoft	OS	1682
18	Enterprise Linux Desktop	Redhat	OS	1469
19	Enterprise Linux Server	Redhat	OS	1419
20	Enterprise Linux Workstation	Redhat	OS	1380
21	Opensuse	Opensuse	OS	1314
22	Leap	Opensuse	OS	1305
23	Tvos	Apple	OS	1295
24	Internet Explorer	Microsoft	Application	1171
25	Safari	Apple	Application	1136
26	Mysql	Oracle	Application	1131
27	Enterprise Linux	Redhat	OS	1052

Linux dialects
Windows versions
Apple OS's

Examples of different protection solutions

General example of control principles

Security controls	Description	Example	Where?
Encryption	*Protection against eavesdropping or unauthorized access*	network traffic, file content, disk partitions, memory pages, swap files/ page area	OpenSSL, IPSec, SSH, OS kernel
Electronic signatures	*Protection against changes or unauthorized modifications by third parties,*	network traffic, file content, disk partitions	OpenSSL, IPSec, SSH, OS kernel
Cryptograph -ically strong hash values	*Protection against unauthorized changes, detect errors or changes*	Saved passwords, file content,	Password file, user database, checksums on files

General example of control principles

Security controls	Description	Example	Where?
Random numbers	*Make a resource non-deterministic*	File names, proccess ID,'s port numbers, sesssion keys, session id's, transaction numbers, DNS query ID's, execution time & timing	getrandom() /dev/urandom
Constant numbers	*Make a resource non-deterministic*	execution time, timing of events	Crypto code to prevent side channel attacks

https://www.redhat.com/en/blog/understanding-red-hat-enterprise-linux-random-number-generator-interface?extIdCarryOver=true&sc_cid=701f2000001OH7JAAW

General example of control principles

Security controls	Description	Example
Compiler generated airbag - canary	*Make sure buffer overflows dont get undetected*	ProPolice, VisualStudio /GS
ASLR	*Randomize addresses used by applications. Make sure its hard to write code that knows of addresses. Where did that lib go?*	Android >4.0, iOS > 4.3, Windows >Vista, OpenBSD/NetNSD, Linux >2.6.12, MacOSX >10.5, Solaris >11.1, etc
KASLR	*Randomize addresses used by kernel*	Windows Vista, NetBSD, Linux >3.14, MacOSX 10.8, Android 11, etc

General example of control principles

Security controls	**Description**	**Example**
DEP, NX, W^X	*Make sure memory is not executable*	IE on Windows Vista, Android >2.3, FreeBSD > 5.3, OpenBSD, Linux >2.6.8, MacOSX >10.5, etc
MTE	*Memory Tagging Extension*	Using ARM architecture feature to better protect against memory safety violations

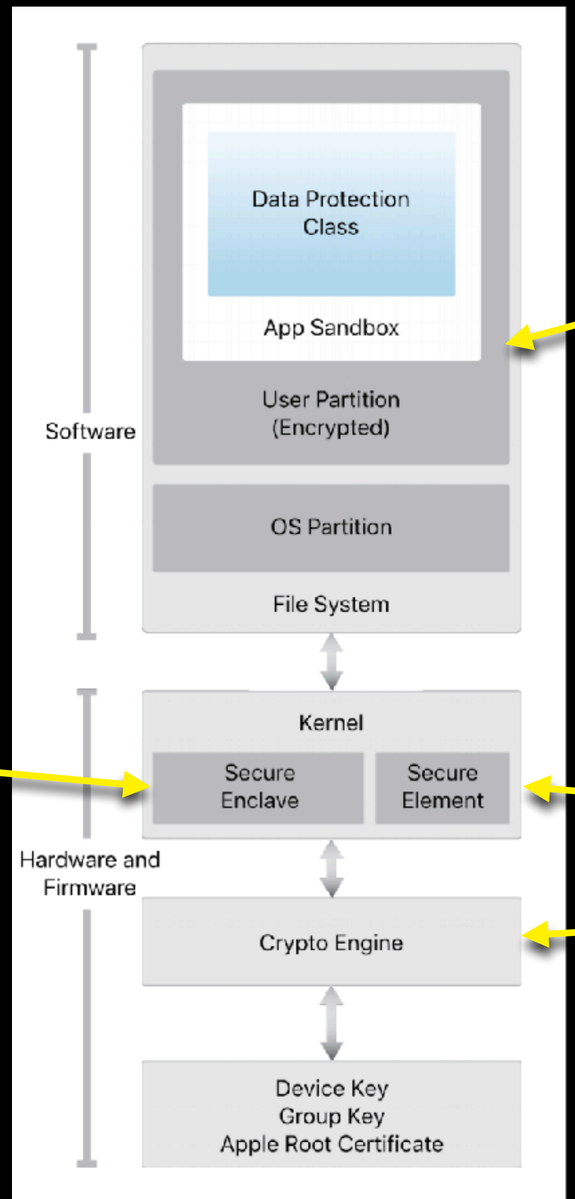
General example of control principles

Security controls	**Description**	**Example**
Secure boot chain / Verified boot	*Make system startup sequence is secure*	Make sure that each step of boot is cryptographically signed to ensure code integrity, e.g. **BIOS vs UEFI**
Secure pairing	*Make sure to connect to peripherals and resources in a secure way*	Using bluetooth to connect to headset,

General example of control principles

Security controls	**Description**	**Example**
Scrubing, zeroing	*Make sure that old data areas are cleaned before usage or returned to system*	memory, file systems, VM system
Logs, audit trails	*Traces, error messages and dumps from systems and applications*	Windows Eventlog, Syslog, audit, BSM

Apple iOS device security



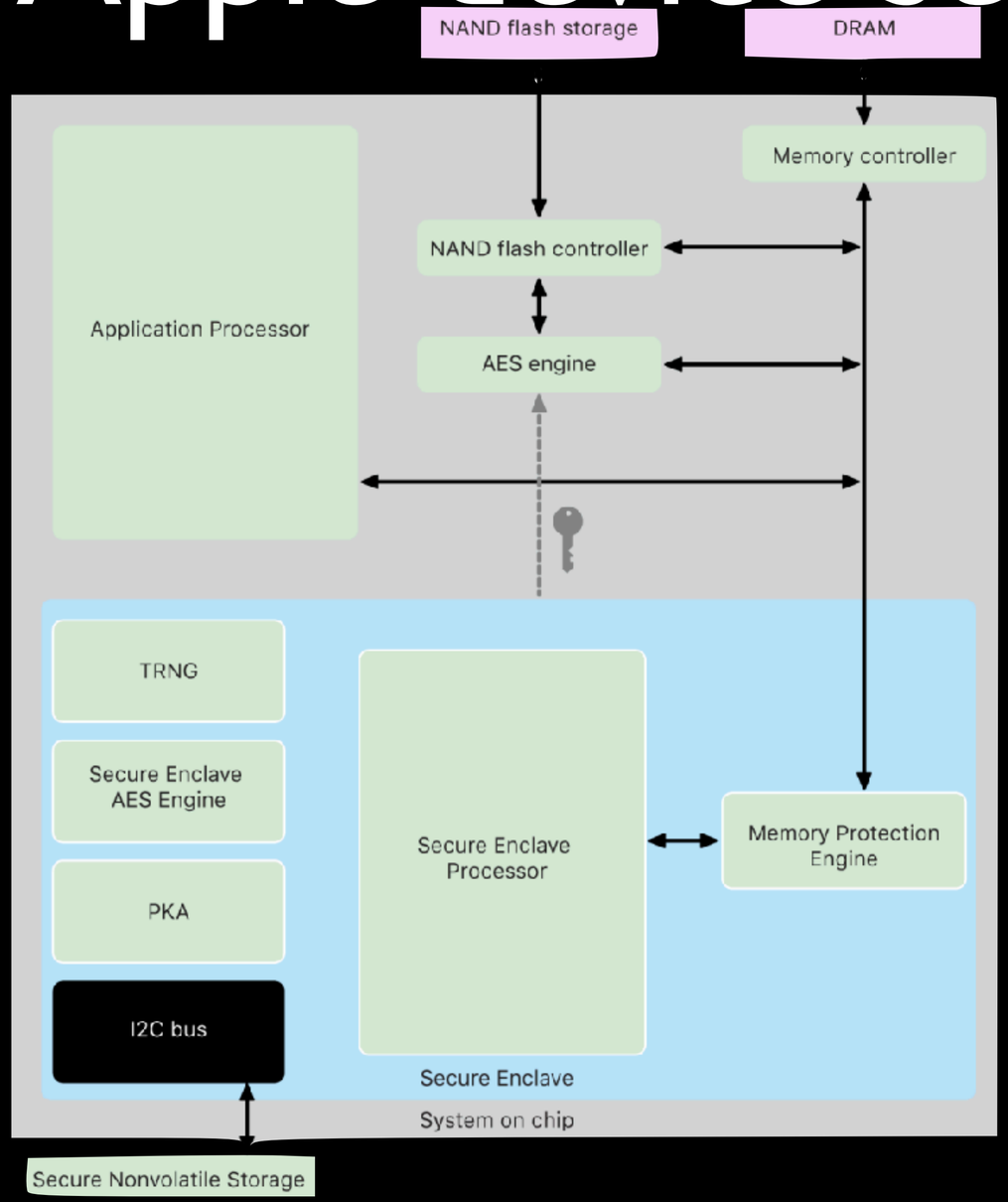
App Sandbox

Secure Enclave

Secure Element

Crypto Engine

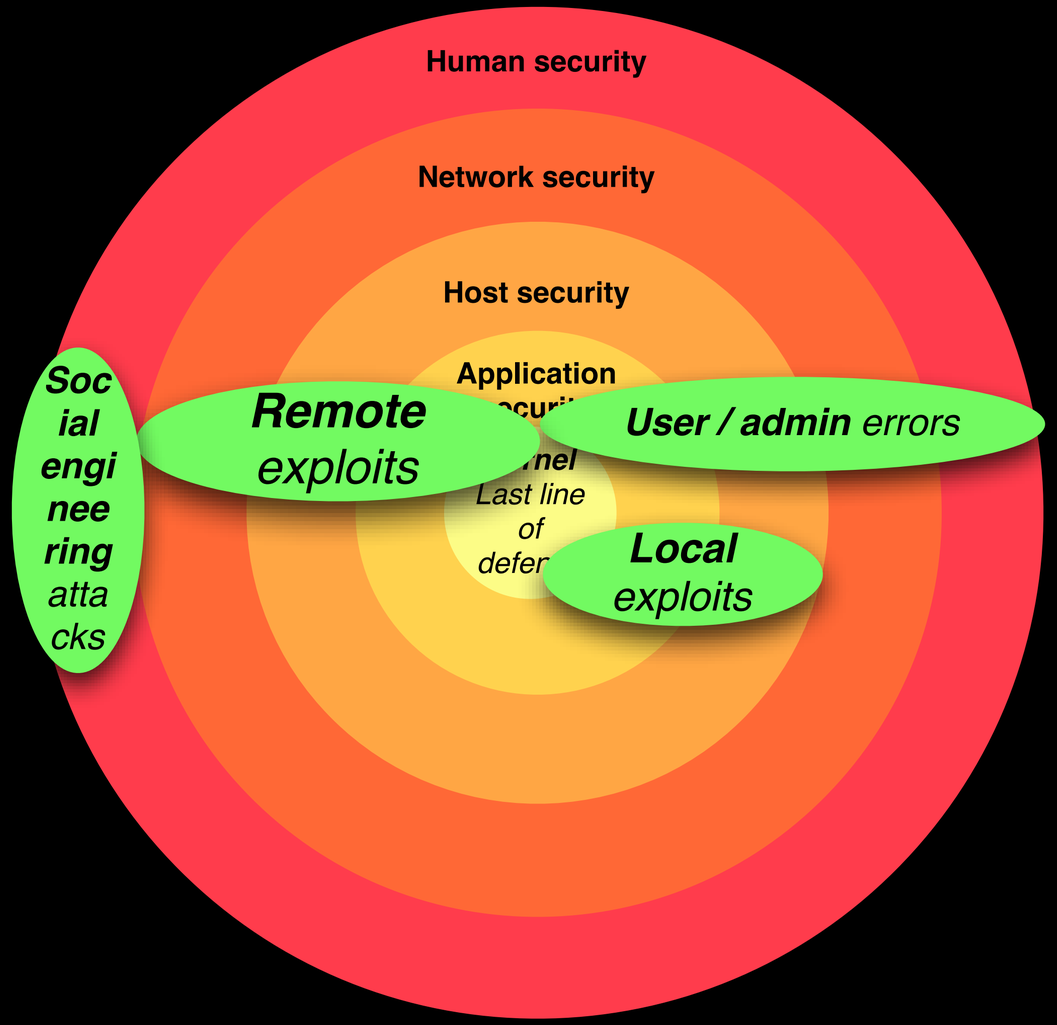
Apple device security



Secure Enclave

Examples of vulnerabilities and attacks

Where do attacks occur?



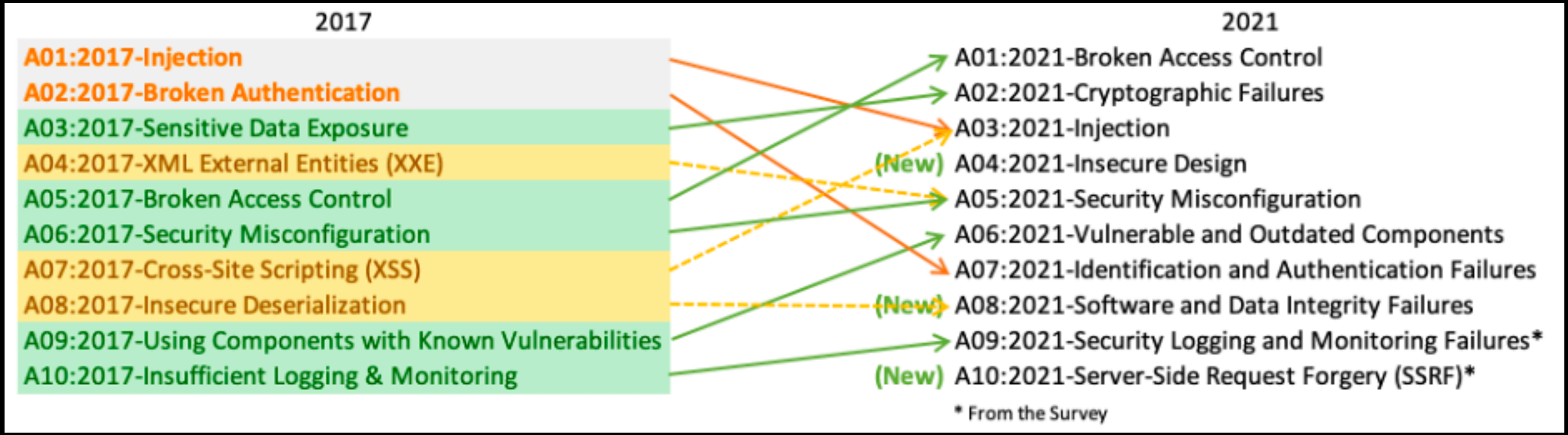
Most common attacks?

2021
A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
A10:2021-Server-Side Request Forgery (SSRF)*

* From the Survey

OWASP top-10 list

Most common attacks?



OWASP top-10 list

General examples of threats and attacks

Sensitive plaintext in RAM

Wrong file permissions

Confidentiality

Crashdumps with credentials or crypto keys

Bypassed security checks

fork bombs

SYN flood

malformed network packets

Availability

*un*intentional filling of disk space

intentional filling of disk space

Manipulated system configuration

System integrity

Manipulated application program binaries

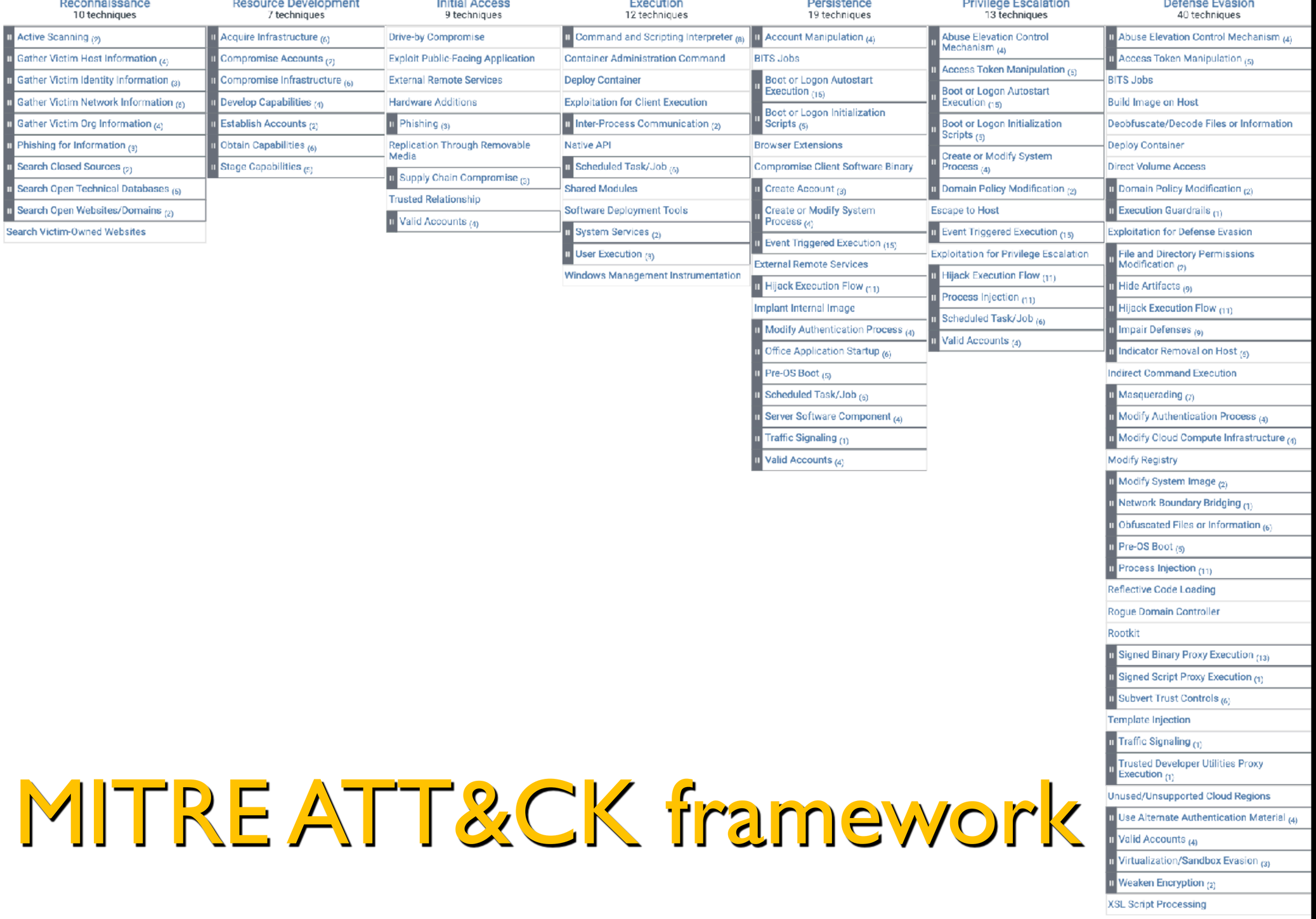
Manipulated system binaries

Manipulated user files

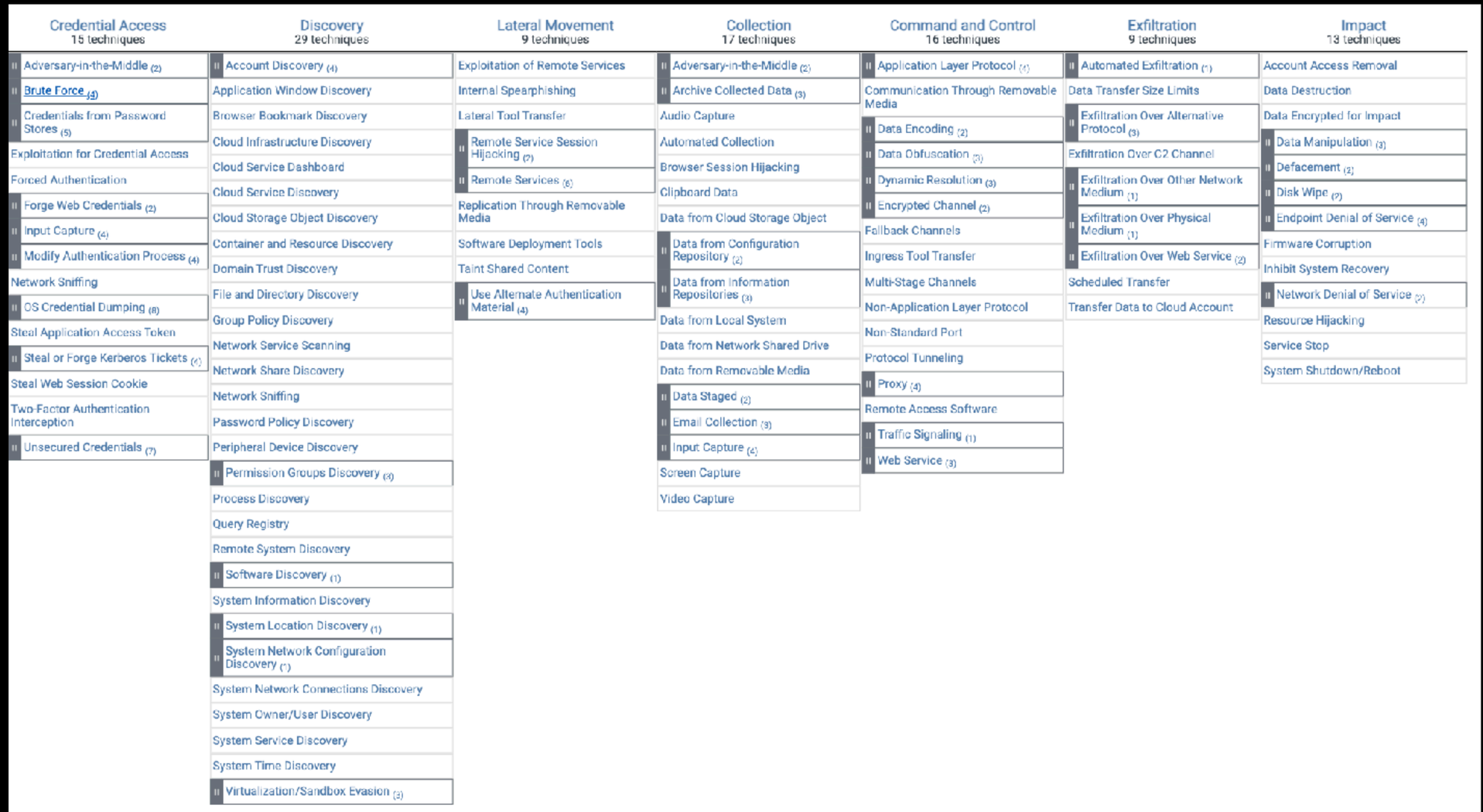
Data integrity

Zapped system logs

Manupulated database content



MITRE ATT&CK framework



MITRE ATT&CK framework

Example of attacks

Attack method	Description	Synonyms and variants
Buffer overflow	Attacks that allow an attacker to *deterministically alter the execution flow of a program by submitting crafted input to an application.* Executable code is written outside the boundaries of a memory buffer originally used for storing data. The executable parts is somehow made to execute, e.g. by manipulate return adress to be used when a function call is finished. Real world examples: OpenBSD IPv6 mbuf's* remote kernel buffer overflow[1], windows kernel pool	Synonyms: memory corruption attack, Buffer overrun, Stack smashing, Variants: Heap smashing, format string bugs,

[1] http://www.coresecurity.com/content/open-bsd-advisorie

* An *mbuf* is a basic unit of memory management in the kernel IPC subsystem

Example of attacks

Attack method	Description	Examples
Backward compability	Attacks that allow an attacker to *use* • *an older version of a service, or* • *an old protocol, or* • *an older mode, or* • *call legacy code* Sometime triggered by downgrade attack, a negotiation to use older variant	Remote Desktop NTLMv1 XML encryption SSLv2, SSLv3, incl POODLE, FREAK Encryption modes Kerberos v4 in v5

https://www.isg.rhul.ac.uk/~kp/BackwardsCompatibilityAttacks.pdf

Attacks and counter measures

?

Hijacking JIT compilers

ROP attacks

Address Space Layout Randomization (ASLR)

No-executable (NX, W^X) stacks

Data Execution Prevention (DEP)

More advanced buffer overflows, defeating canary

Stack canaries

Buffer overflow/memory corruption attacks

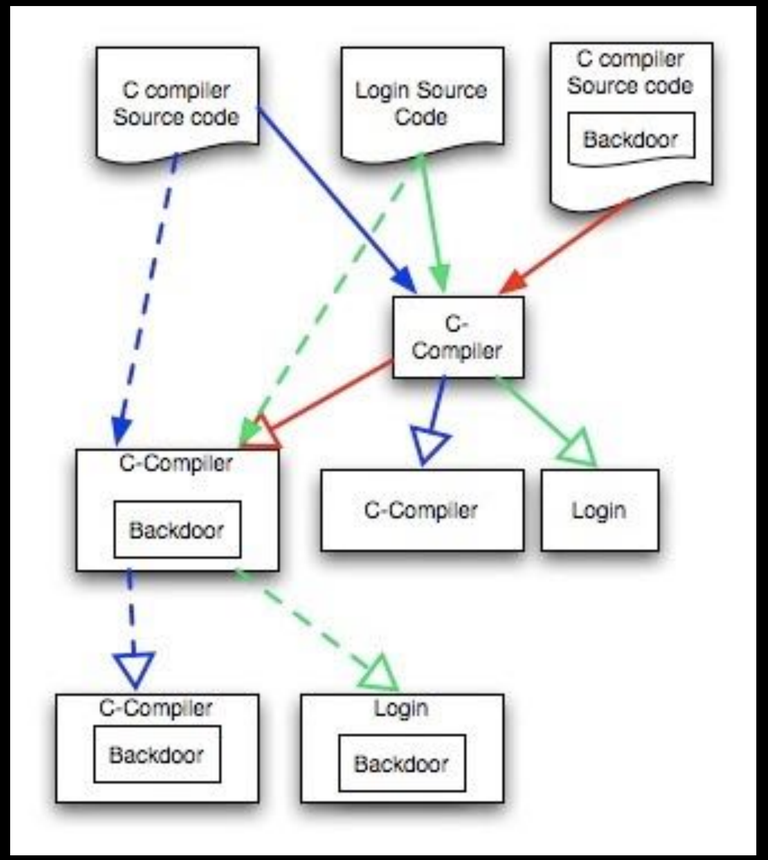
Note - several of these counter measures does not work for protection **within** the kernel

Attacks and counter measures

- *Chaining of attacks - combining a number of exploits to achieve goal*
  - finding and abusing a number of different vulnerabilities might allow an attacker to achieve goals not possible with just one potent exploit
  - *Code execution in gadgets (ROP) + sandbox escape + elevation of privileges + execution of privileged code*

A classic attack

- Ken Thompson's trojanized c compiler
  - Modify the source code to the compiler to recognize if it recompile itself or the login program - insert backdoor in login
  - recompile compiler
  - remove source code changes and recompile the compiler
  - recompile the login program with the modified compiler
- No visible signs for humans or tools to see the backdoor in source code. Calls for binary inspection or decompilation.



Ken Thompson - TURING AWARD LECTURE: *Reflections on Trusting Trust.* http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.5728&rep=rep1&type=pdf

Example of attacks

Remember that there is a number of ways that all OS security controls can be bypassed,
especially if the operating system is not running
- a very good side-channel attack ;-)

Virtualization and isolation

sandboxes, containers, hypervisors, etc

Sandboxing

- Various types of OS supported or application supported ***sandboxing*** is good as a way to get defense-in-depth
- Create ***temporary execution environments*** for certain tasks
  - test of exe files to lure out malicious code execution
  - perform certain tasks that is more prone to attacks
  - perform certain tasks that is more sensitive
- Provide ***isolation***, from other parts of system

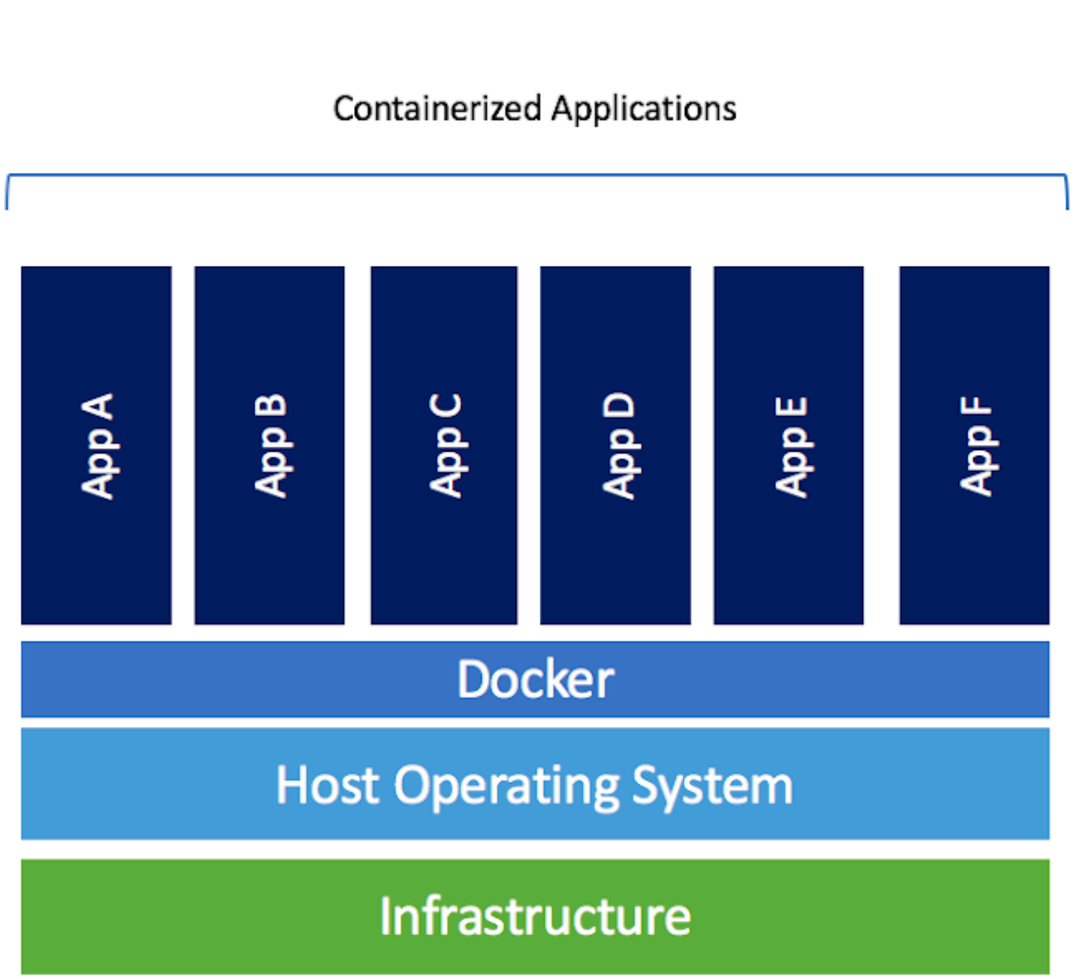
Pro's and con's with virtualization

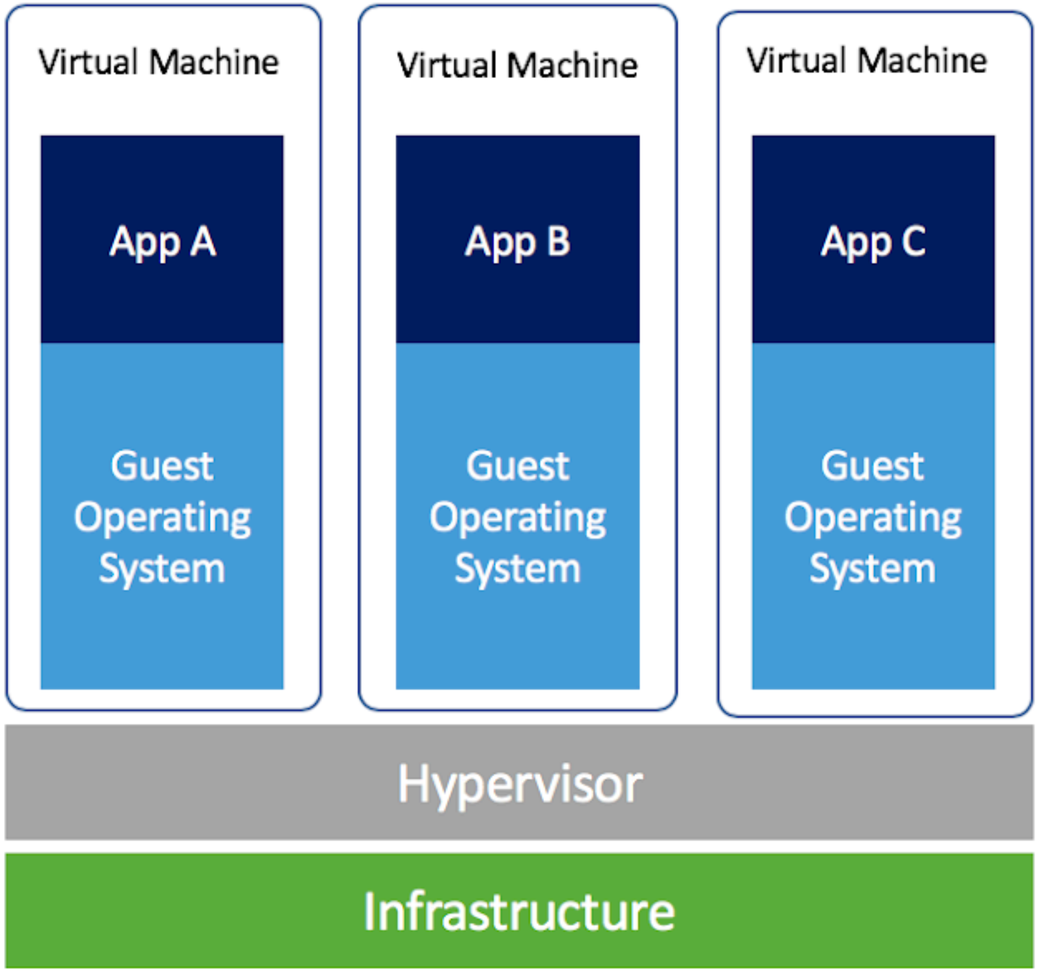
- Some sandbox and isolation technologies are not complete virutalization or separation
  - E.g. share *name space* (processes, file system, etc)
  - Share operating system kernel
  - Share drivers

Isolation, separation and virtualization

Type	Example	Description
chroot	Change root for network service	Classic Unix concept. No virtualization, just isolation
jails	FreeBSD Jails	Userland. Can run FreeBSD and Linux binaries. Integrated into OS.
user mode linux, uml		Userland. More lightweight and thus faster than virtual machines
Containers	Docker	Userland. 3rd party tool on top of OS. Need container engine, not hypervisor. More lightweight and thus faster than virtual machines
Virtual machines	Xen, VMware vSphere, HyperV,	Type 1 hypervisor based. Stronger isolation than container
Virtual machines	VMware Workstation, KVM	Type 2 hypervisor based. Stronger isolation than container
Hardware partitioning	Sun LDOMs, IBM LPAR	Best isolation and separation. Hardware support gives superior performance and security

Overview of virtualization





Pro's and con's with virtualization

- Isolation, and to have hardened and dedicated servers running specific services, are standard ways to minimize attack surface. Virtualization tools can help this
- Its easy to believe that virtualization will automatically make things secure, and that there is no way to jump between guest os', but exploits have shown this not hold true, e.g. cloudburst

http://www.immunityinc.com/documentation/cloudburst-vista.html

Advanced attacks

Hardware attacks, etc

Attack tools

- Reverse Engineering Frameworks, such as Ghidra help debugging, disassemble, reverse engineer binaries
  - Give attackers powerful tools to introspect into applications, drivers, kernels

Example of attacks

- Attacks by attaching malicious hardware to buses and ports
  - Using debug interfaces to snoop & manipulate bus
  - **JTAG** (IEEE standard 1149.1-1990)
  - **SWD** (Serial Wire Debug)
- Firewire and other DMA based methods to access memory of a computer (*evil maid attacks, evil devices*)
- UEFI attacks via Thunderbolt (*thunderstruck attack*)

https://payatu.com/blog/hardware-attack-surface-jtag-swd/

Example of attacks

- Removal of, or direct attachment to, physical memory chips (*cold boot attacks*)

Example of attacks: *cold boot attacks*

F-secure ”The Chilling Reality of Cold Boot Attacks” https://www.youtube.com/watch?v=E6gzVVjW4yY

Example of attacks: *PCILeech*

Ulf Frisk - ”Attacking UEFI Runtime Services and Linux” https://www.youtube.com/watch?v=PiUVRHYTDUg

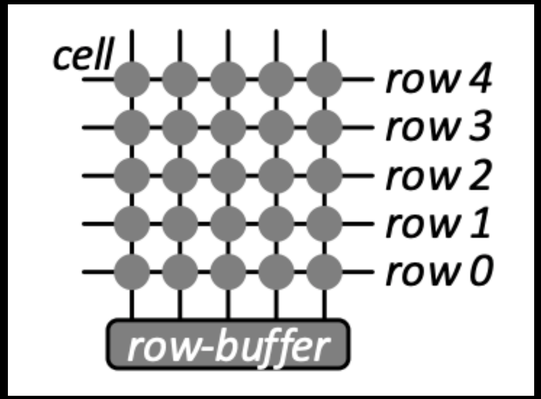
Example of attacks: *HW implants*

(TS//SI//NF) Left: Intercepted packages are opened carefully; Right: A "load station" implants a beacon

https://arstechnica.com/tech-policy/2014/05/photos-of-an-nsa-upgrade-factory-show-cisco-router-getting-implant/

Advanced attacks

- *Rowhammer**
  - *Flipping bits without accessing them*
  - *Method of reading writing memory cells so that memory cells in adjacent rows become changed*
  - *Based on an unintended side effect in dynamic random-access memory (DRAM) that causes memory cells to leak their charges and interact electrically between themselves, possibly altering the contents of nearby memory rows that were not addressed in the original memory access*



"Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors"
— Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, Onur Mutlu, at CMU

Advanced attacks

- *Rowhammer**
  - *This circumvention of the isolation between DRAM memory cells*
    - *Memory leak == information leak*
  - *Have been used to **Gain Kernel Privileges, e.g. DRAMMER attack on Android***
  - *Can be used to attack **Virtual Machines***

* Kim et al ” Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors” https://users.ece.cmu.edu/~yoonguk/papers/kim-isca14.pdf

Advanced attacks

- *Rowhammer*
  - *Have been implemented in JavaScript and runned in a browser*
  - *Modern variants* have been used to* ***defeat ECC memory***

* ”Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks” https://cs.vu.nl/~lcr220/ecc/ecc-rh-paper-eccploit-press-preprint.pdf

Advanced attacks

- *Rowhammer**
  - Initial research published 2014, but variants have been developed later
  - Hardware solutions to protect against it have been circumvented
    - Blacksmith
    - Half-double

https://security.googleblog.com/2021/05/introducing-half-double-new-hammering.html

https://comsec.ethz.ch/wp-content/files/blacksmith_sp22.pdf

Advanced attacks

- *Meltdown* & Spectre***
  - *Low-level **cache** attacks, allow malicious READ's*
  - *Meltdown breaks isolation between **user land and kernel***
  - *Spectre breaks isolation between **applications in user land***

https://meltdownattack.com/

* Lipp et al "Meltdown: Reading Kernel Memory from User Space" https://meltdownattack.com/meltdown.pdf

** Kocker et al "Spectre Attacks: Exploiting Speculative Execution" https://spectreattack.com/spectre.pdf

Advanced attacks

- *Meltdown & Spectre*
  - *work on personal computers, mobile devices, and in the cloud*
  - *Works on Windows, Linux, Android, etc*
  - *Works on containers: docker, LXC, OpenVZ etc*

Advanced attacks

- *Spectre class vulnerabilities will remain unfixed because otherwise CPU designers will have to disable speculative execution which will entail a massive performance loss*

Advanced attacks

- *Meltdown & Spectre*
  - *All modern CPUs are vulnerable (x86, AMD, ARM) in various degrees*

	Method	Attack	Spectre-PHT	Spectre-BTB	Spectre-RSB	Spectre-STL
Intel	same-address-space	in-place	● [52, 50]	★	● [62]	● [32]
		out-of-place	★	● [18]	● [62, 54]	○
	cross-address-space	in-place	★	● [52, 18]	● [62, 54]	○
		out-of-place	★	● [52]	● [54]	○
ARM	same-address-space	in-place	● [52, 50]	★	● [6]	● [6]
		out-of-place	★	☆	● [6]	○
	cross-address-space	in-place	★	● [6, 52]	☆	○
		out-of-place	★	☆	☆	○
AMD	same-address-space	in-place	● [52]	★	★	● [32]
		out-of-place	★	☆	★	○
	cross-address-space	in-place	★	● [52]	★	○
		out-of-place	★	☆	★	○

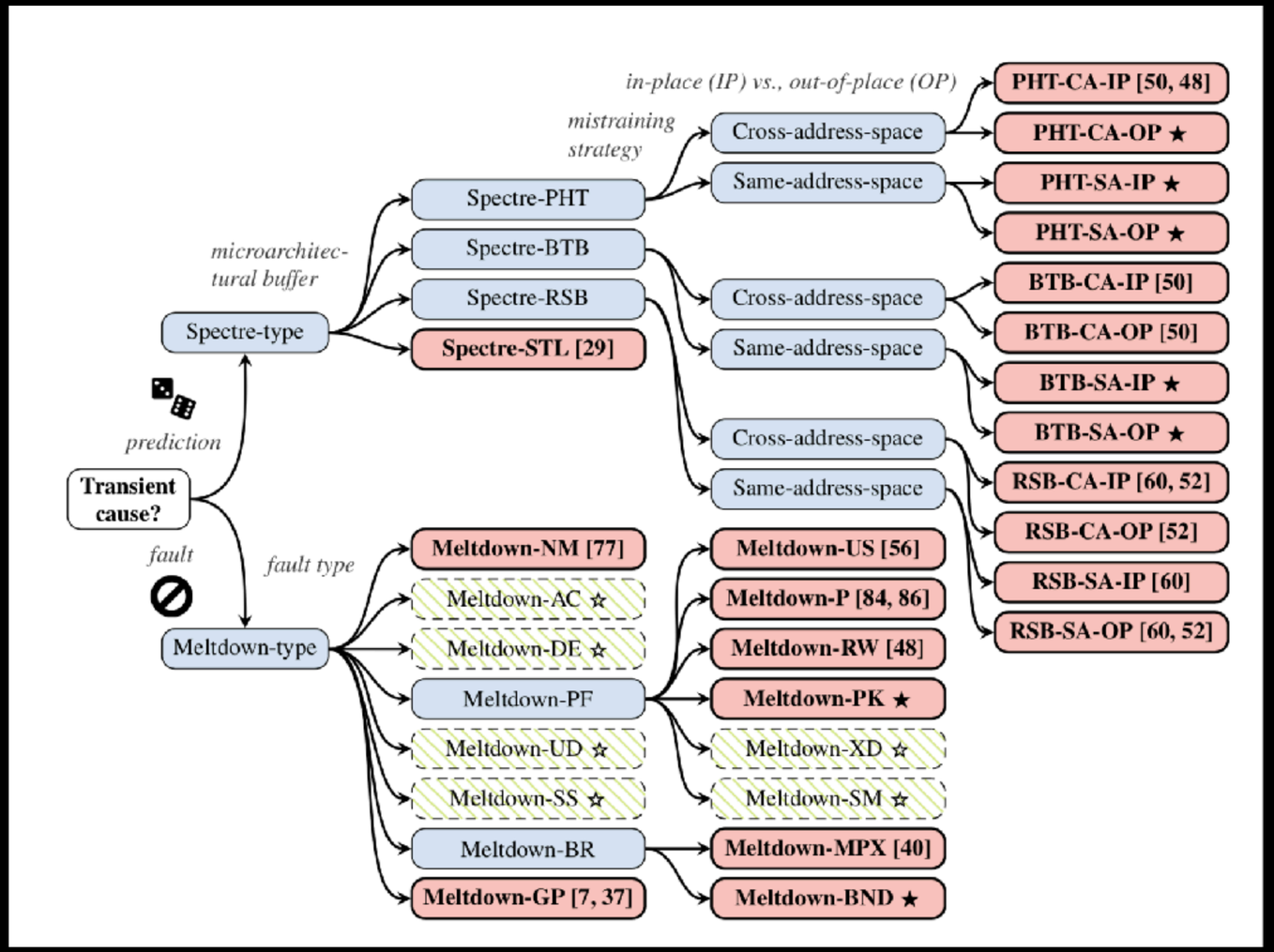
Symbols indicate whether an attack is possible and known (●), not possible and known (○), possible and previously unknown or not shown (★), or tested and did not work and previously unknown or not shown (☆). All tests performed with no defenses enabled.

Vendor \ Attack	Meltdown-US [59]	Meltdown-P [90, 93]	Meltdown-GP [10, 40]	Meltdown-NM [83]	Meltdown-RW [50]	Meltdown-PK	Meltdown-BR	Meltdown-DE	Meltdown-AC	Meltdown-UD	Meltdown-SS	Meltdown-XD	Meltdown-SM
Intel	●	●	●	●	●	★	★	☆	☆	☆	☆	☆	☆
ARM	●	○	●	_	●	_	_	☆	☆	☆	_	☆	☆
AMD	○	○	○	○	○	_	★	☆	☆	☆	☆	☆	☆

Symbols indicate whether at least one CPU model is vulnerable (filled) vs. no CPU is known to be vulnerable (empty). Glossary: reproduced (● vs. ○), first showed in this paper (★ vs. ☆), not applicable (_). All tests performed without defenses enabled.

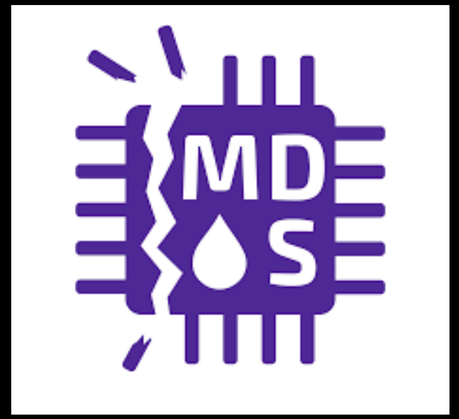
* Canello et al "A Systematic Evaluation of Transient Execution Attacks and Defenses" https://arxiv.org/pdf/1811.05441.pdf

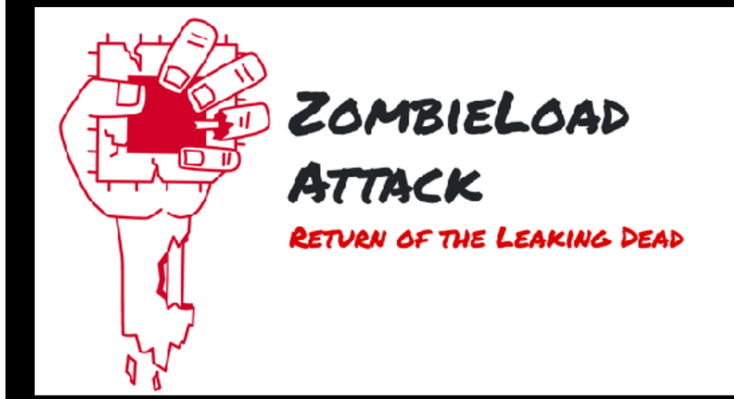
Advanced attacks



* Canello et al ”A Systematic Evaluation of Transient Execution Attacks and Defenses” https://arxiv.org/pdf/1811.05441.pdf

Advanced attacks





https://en.wikipedia.org/wiki/Transient_execution_CPU_vulnerability

Advanced attacks

- *Attacks against Intel Management Engine*
  - *Proprietary and non-documented*
  - *Own OS (Minix!)*
  - *Reverse engineered and analysed by attackers*
  - *Found multiple vulnerabilities in Skylake & Kabylake architecture*

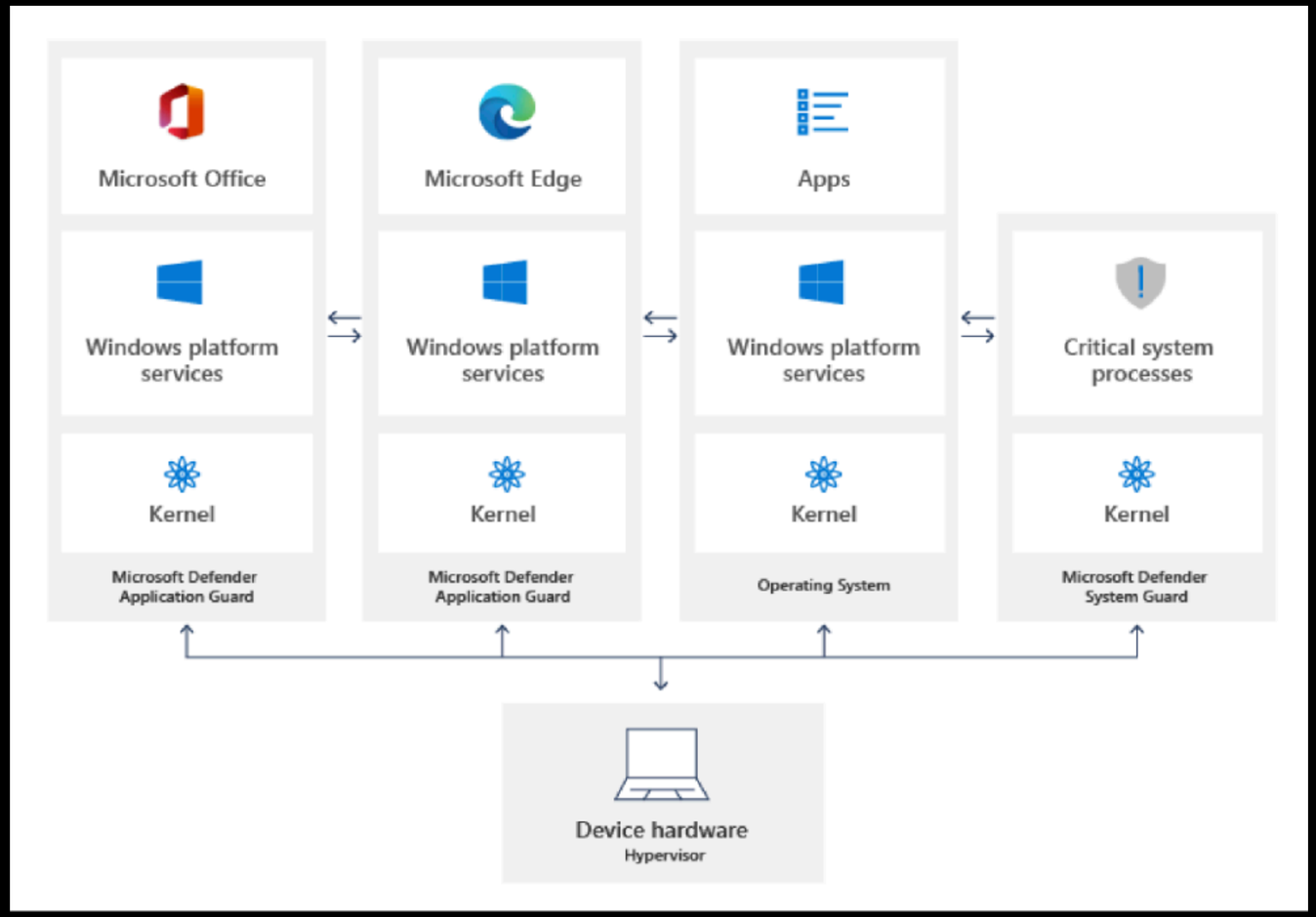
Examples of modern security controls

Windows Defender security features in Win 10, Win 11

Windows

- Application Guard, **WDAG**
  - App & browser control
  - Isolation browsing

- Windows Device Guard, now called Windows Defender Application Control



https://docs.microsoft.com/en-us/deployedge/microsoft-edge-security-windows-defender-application-guard

https://docs.microsoft.com/en-us/windows/security/threat-protection/microsoft-defender-application-guard/md-app-guard-overview

Windows

- Windows Device Guard, And Applocker, now called Windows Defender Application Control
  - Attributes of the *codesigning* certificate(s) used to sign an app and its binaries
  - Attributes of the app's binaries that come from the signed metadata for the files, such as Original Filename and version, or the hash of the file
  - The path from which the app or file is launched

https://docs.microsoft.com/en-us/deployedge/microsoft-edge-security-windows-defender-application-guard

https://docs.microsoft.com/en-us/windows/security/threat-protection/microsoft-defender-application-guard/md-app-guard-overview

Windows

- *Core isolation* with *Memory integrity*, aka Hypervisor-protected Code Integrity (**HVCI**)
  - make it difficult for malicious programs to use low-level drivers to hijack your computer

https://support.microsoft.com/en-us/windows/core-isolation-e30ed737-17d8-42f3-a2a9-87521df09b78

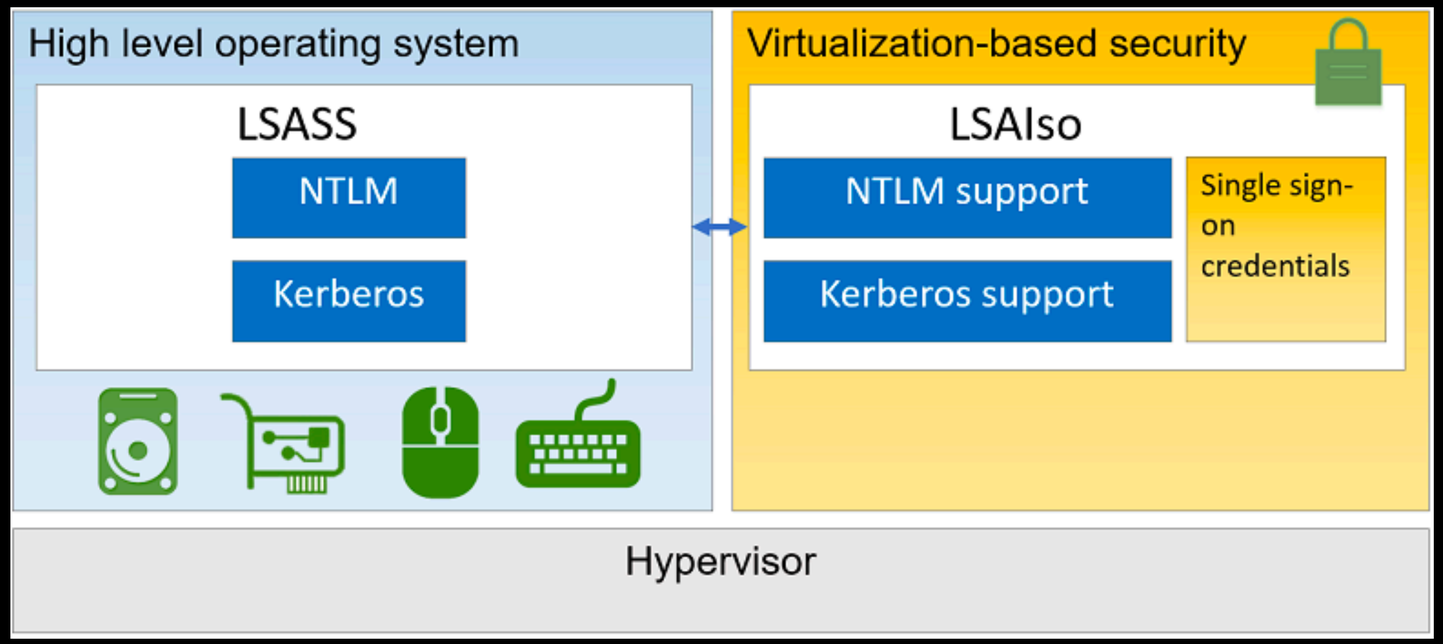
Windows

- Windows Defender Exploit Guard, **WDEG**
  - **Attack Surface Reduction (ASR)**: A set of controls that enterprises can enable to prevent malware from getting on the machine by blocking Office-, script-, and email-based threats
  - **Network protection**: Protects the endpoint against web-based threats by blocking any outbound process on the device to untrusted hosts/IP through Windows Defender SmartScreen
  - **Controlled folder access**: Protects sensitive data from ransomware by blocking untrusted processes from accessing your protected folders
  - **Exploit protection**: A set of exploit mitigations (replacing EMET) that can be easily configured to protect your system and applications

https://support.microsoft.com/en-us/windows/core-isolation-e30ed737-17d8-42f3-a2a9-87521df09b78

Windows

- Windows Credential Guard
  - To protect *Local Security Authority Server Service* (LSASS) by moving it into *LSAIso*
- Build on top of
  - Virtualization Based Security (VBS)
  - Secure boot
  - Trusted Platform Module (TPM)
  - UEFI lock

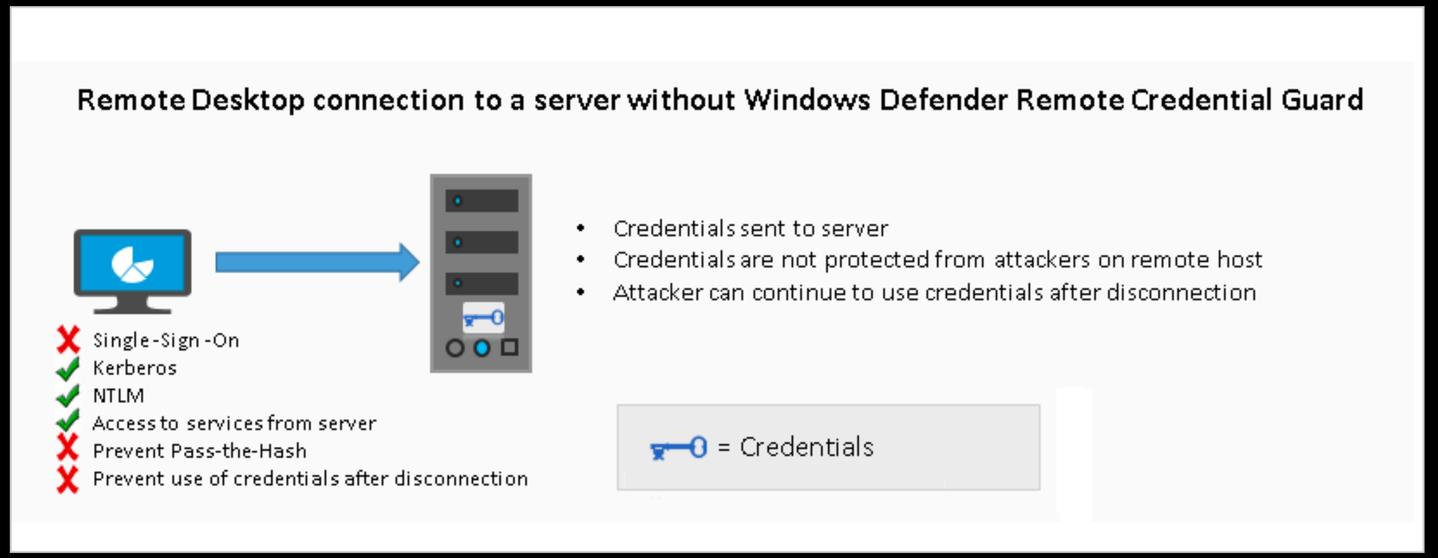


https://docs.microsoft.com/en-us/windows/security/identity-protection/credential-guard/credential-guard-how-it-works

https://en.wikipedia.org/wiki/Local_Security_Authority_Subsystem_Service

Windows

- Windows Remote Credential Guard
- To protect *against theft of credentials sent to server side*
  - *Others that have admin access to the server*
- *Especially important on jump hosts*



https://docs.microsoft.com/en-us/windows/security/identity-protection/credential-guard/credential-guard-how-it-works

https://en.wikipedia.org/wiki/Local_Security_Authority_Subsystem_Service

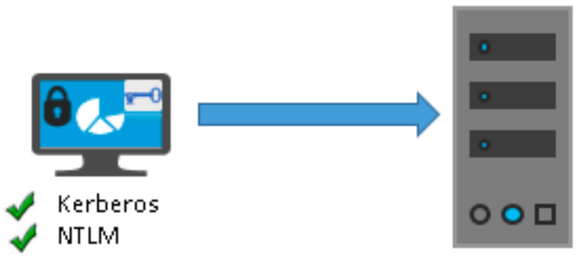
Windows

Windows Defender Remote Credential Guard

- ✓ Kerberos
- ✗ NTLM
- ✓ Access to services from server
- ✓ Prevent Pass-the-Hash
- ✓ Prevent use of credentials after disconnection

- Credentials protected by Windows Defender Remote Credential Guard
- Connect to other systems using SSO
- Host must support Windows Defender Remote Credential Guard

Restricted Admin Mode



- ✓ Kerberos
- ✓ NTLM
- ✗ Access to services from server
- ✓ Prevent Pass-the-Hash
- ✓ Prevent use of credentials after disconnection

- Credentials used are remote server local admin credentials
- Connect to other systems using the host's identity
- Host must support Restricted Admin mode
- Highest protection level
- Requires user account administrator rights

= Credential protection

= Credentials

https://docs.microsoft.com/en-us/windows/security/identity-protection/remote-credential-guard

MacOSX

- GateKeeper
  - Checks code signing
- XProtect
  - Malware protection

Tools mentioned during the class

- Ghidra - Reverse Enginering Framework
- IDA pro - Disassembler
- Hexray - Decompiler
- Ollydbg, windbg - Other disassemblers
- Bindiff - Advanced tool from zynamics to compare binaries, with call graphs etc. Not same as built-in windows tool with same name.

Referenses used during the class

- https://www.commoncriteriaportal.org/
- https://www.cs.virginia.edu/~av6ds/papers/isca2021a.pdf
- https://www.cvedetails.com/top-50-products.php
- https://owasp.org/www-project-top-ten/

Referenses used during the class

- http://en.wikipedia.org/wiki/Source_lines_of_code
- https://sources.debian.org/stats/
- https://informationisbeautiful.net/visualizations/million-lines-of-code/
-

Referenses used during the class

- https://docs.microsoft.com/en-us/windows/security/identity-protection/credential-guard/credential-guard-how-it-works
- https://docs.microsoft.com/en-us/deployedge/microsoft-edge-security-windows-defender-application-guard
- https://docs.microsoft.com/en-us/windows/security/threat-protection/microsoft-defender-application-guard/md-app-guard-overview
- https://docs.microsoft.com/en-us/windows/security/identity-protection/remote-credential-guard

Referenses used during the class

- https://support.microsoft.com/en-us/windows/core-isolation-e30ed737-17d8-42f3-a2a9-87521df09b78
- https://en.wikipedia.org/wiki/Local_Security_Authority_Subsystem_Service
- http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.5728&rep=rep1&type=pdf
-