Operating Systems Security: concepts in security controls, vulnerabilities and attacks

A *quantitative* approach
2025-02-24

Robert Malmgren
rom@romab.com

1 minute presentation

- Consultant in IT, infosec and cybersecurity since 25+ 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

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

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

Multiuser

Multitasking

Locally & remote

Multiple services and clients

Intro - foundation: *Isolation*

- Modern software is normally formed into components, parts and layers in *systems*
- This will create a software stack
  - Layers in the stack provides abstraction
  - Layers in the stack provides supporting frameworks, functionality and support mechanisms
- Layers in the stack is one form of isolation

Intro - foundation: *Isolation*

- 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 algorithms and crypto mechanisms can also be used to compartmentalise and isolate information.

Intro - foundation: IAM

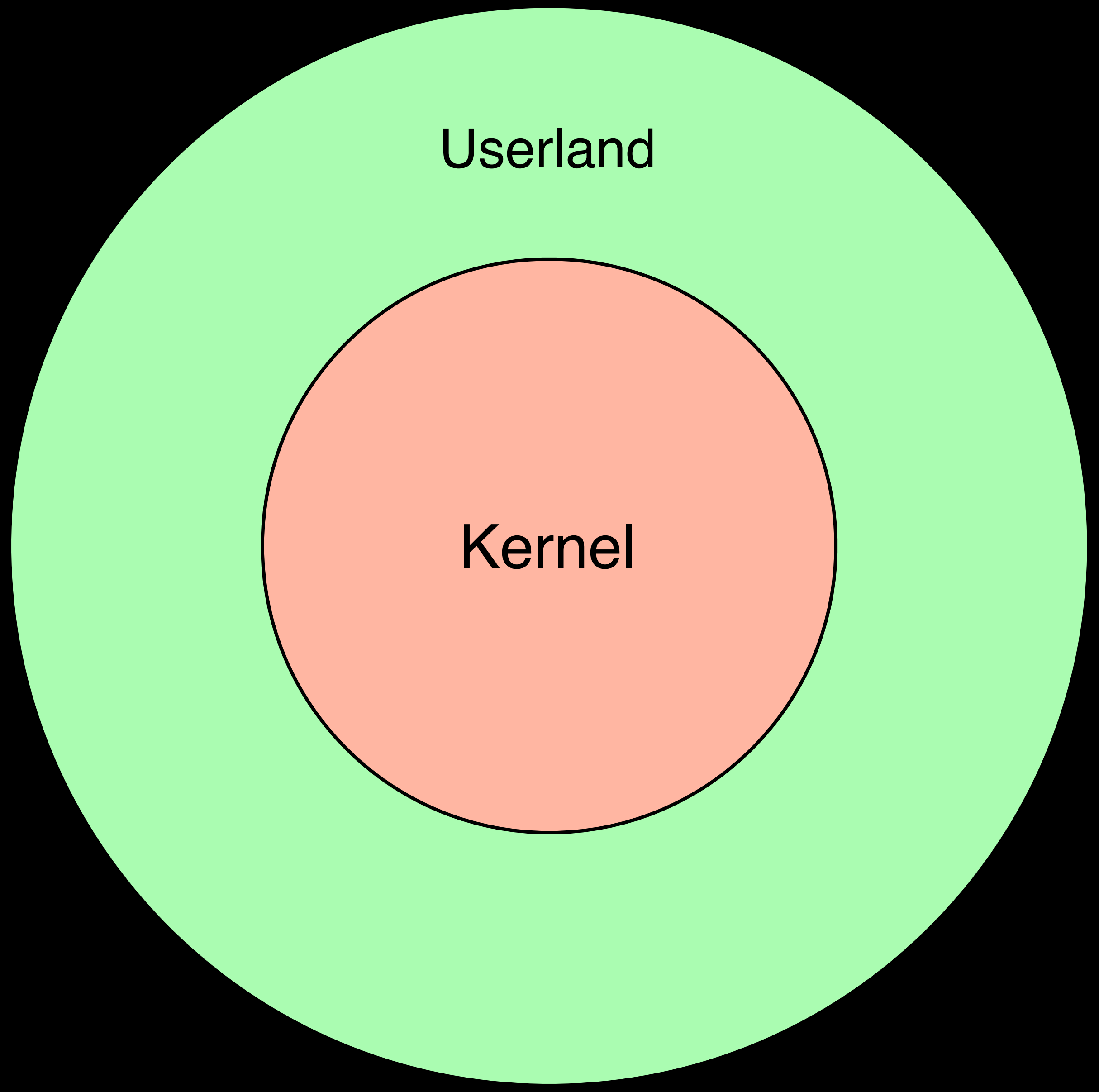
- This there is to built-in security into the foundation of the systems - the operating system
    - To identify and authorize users of the system
    - To allow for an environment where necessary basic controls are in place
    - To prevent unauthorised access to OS resources

Capabilities and requirements

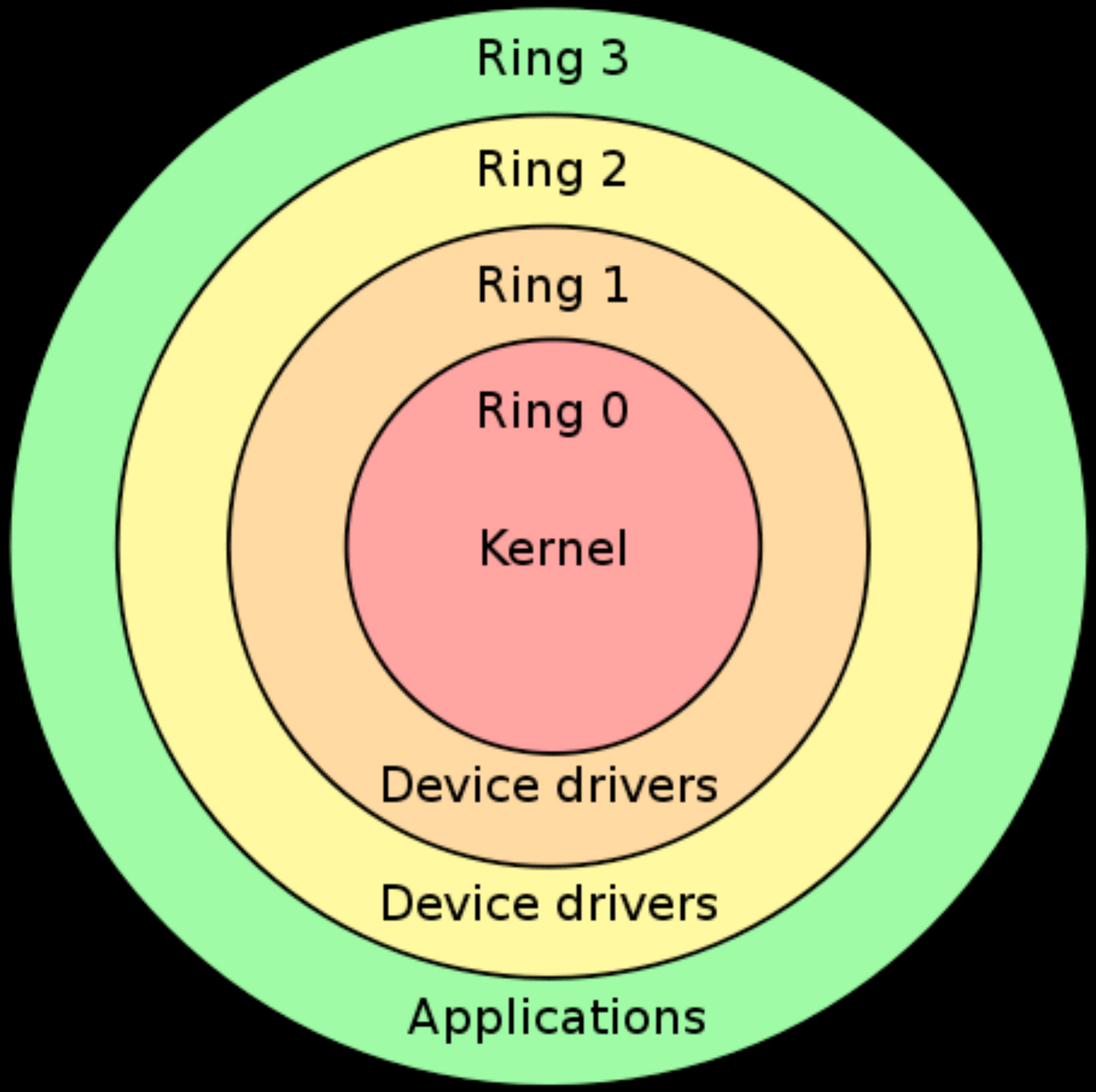
Need	**Description**	**Example**
Protect a system resource	*Prohibit **malicious** or **unintentional** access to system resources*	System tables, direct access to I/O-units, memory protection
Authorization checks for usage of system calls and system resources	*Provide controlled access to system, so 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

The classical *ring model*

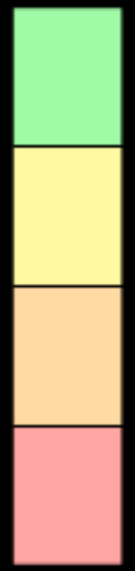
UNIX



x86



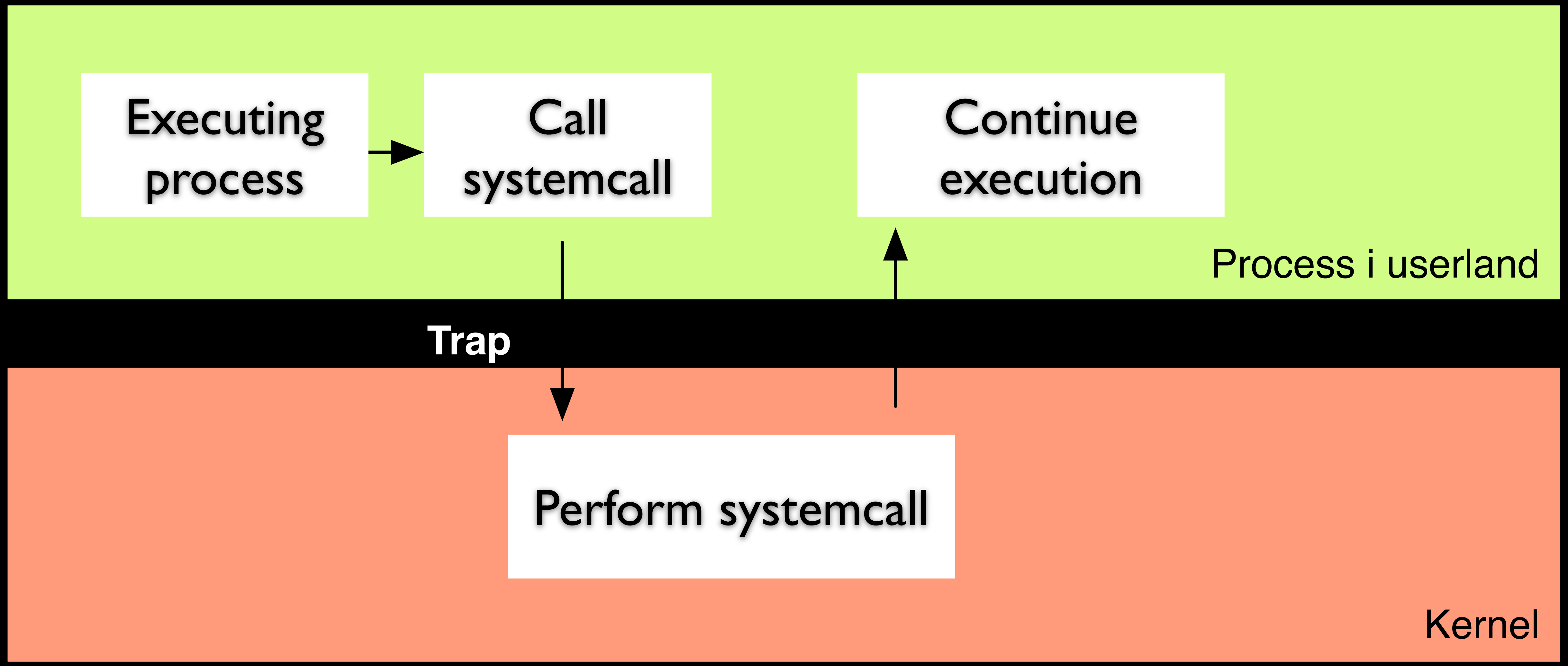
Least privileges



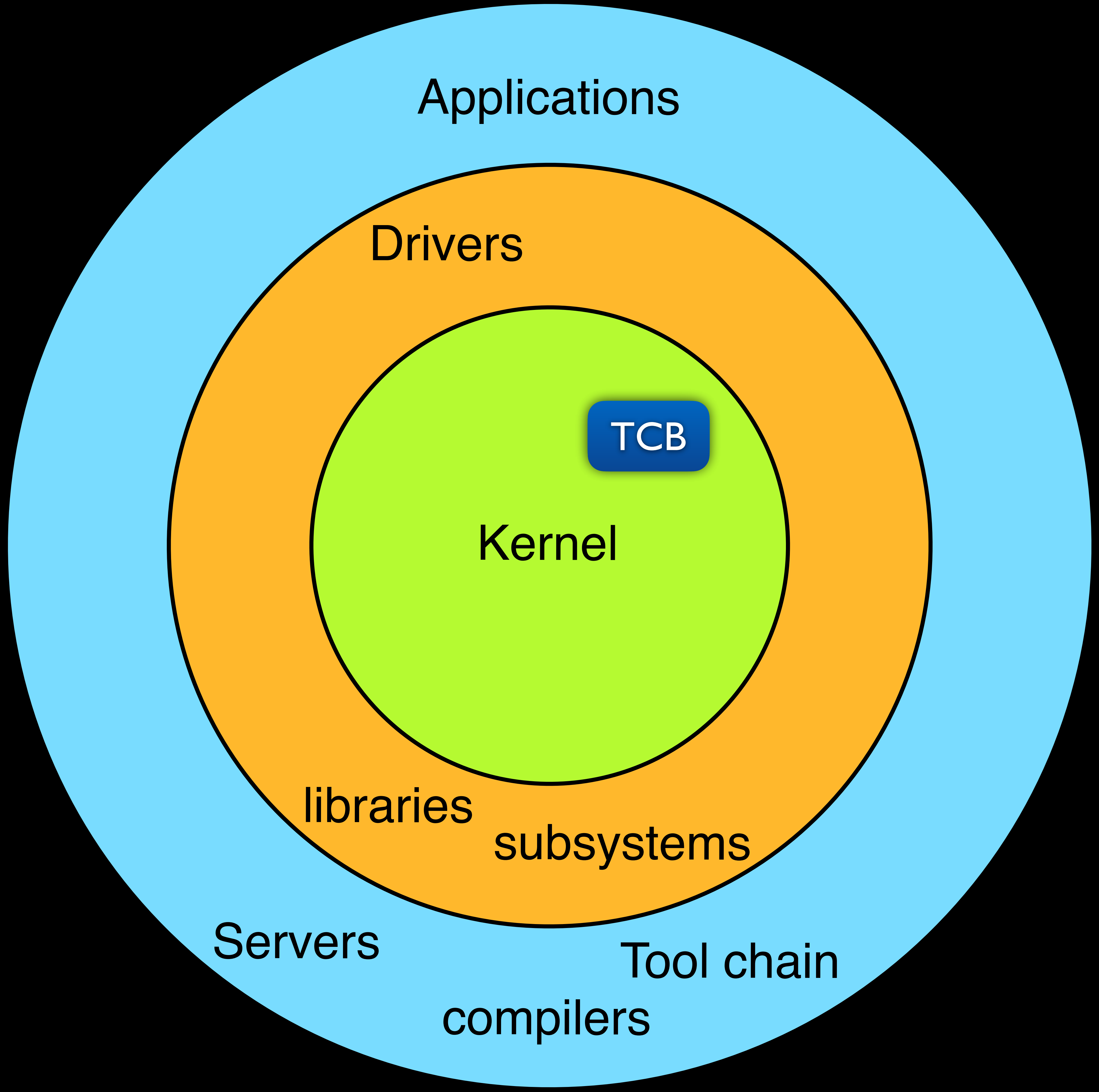
Highes privileges

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

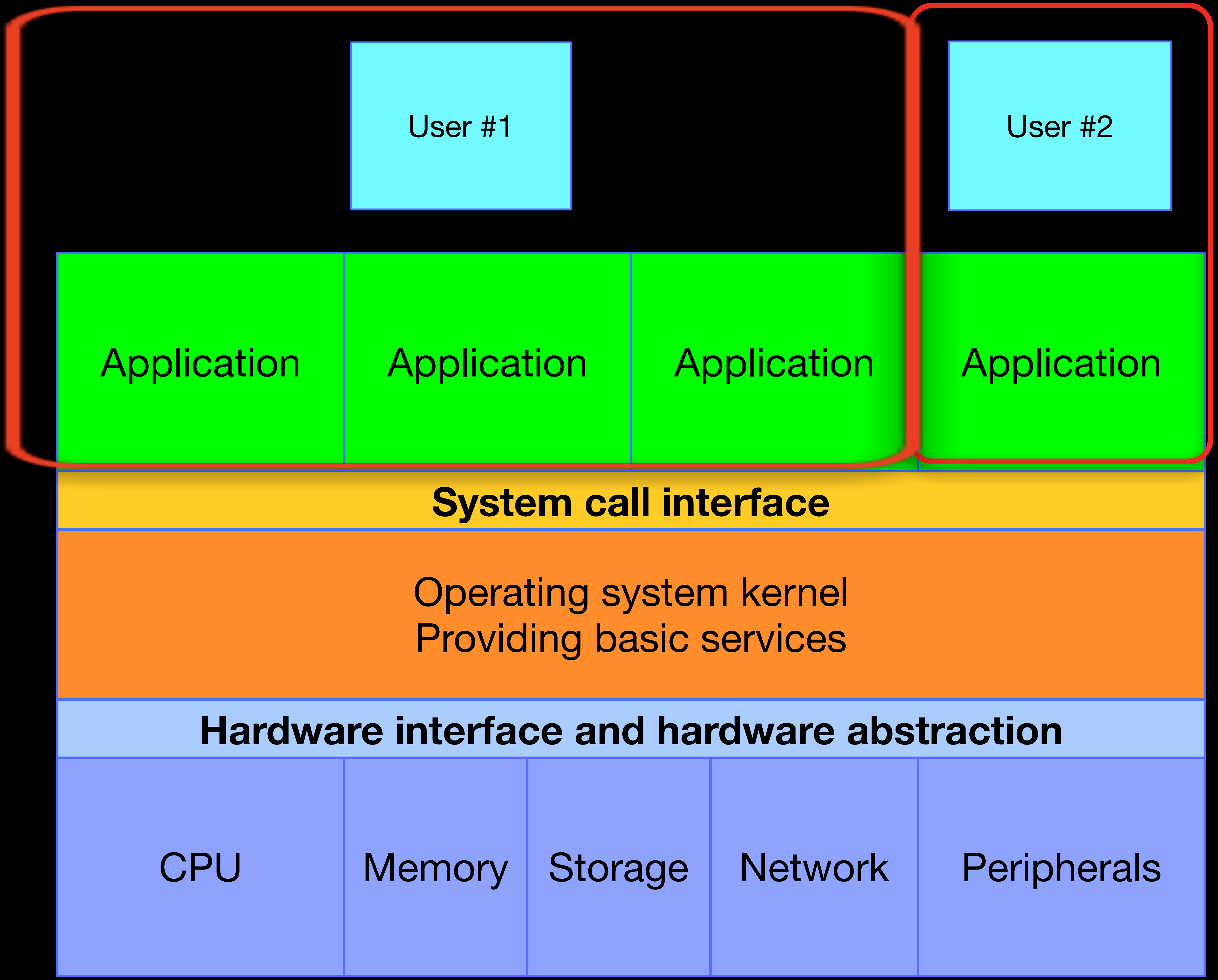
Interaction between application and OS



Overview of operating system (1/2)



Overview of operating system (2/2)



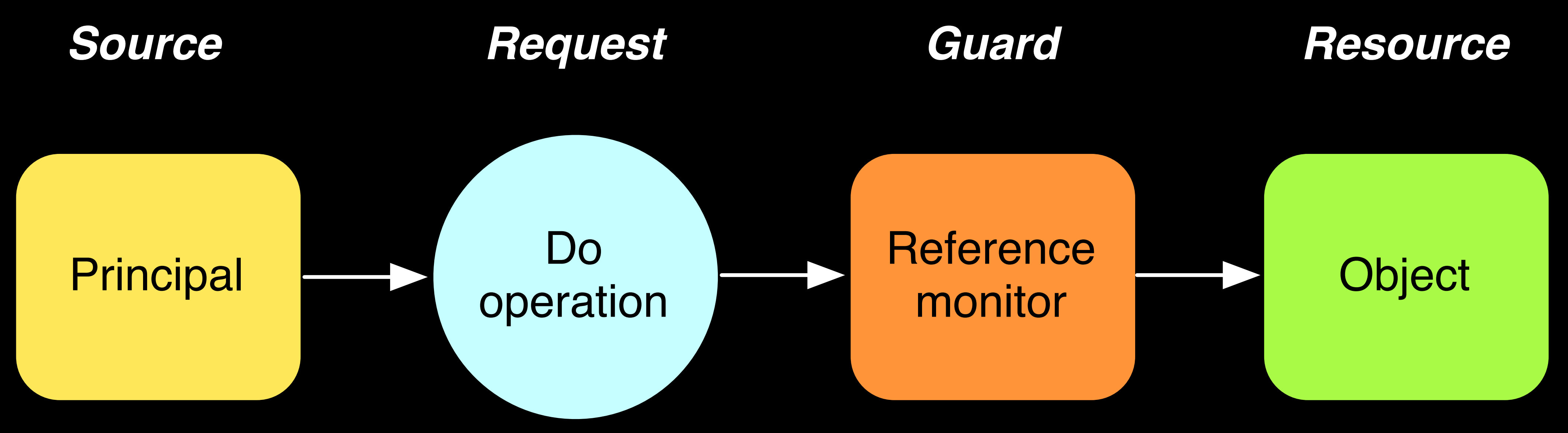
Some important concept

- A concept called *Trusted Computing Base*, or TCB
  - It contain *all things in the trusted part* of the OS necessary to enforce the *security policy*
  - Important that TCB is *small, clearly written, easy to see that it does not contain design or logical flaws, and that it is protected against alterations and tampering*

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

Some important concept

- Reference monitor



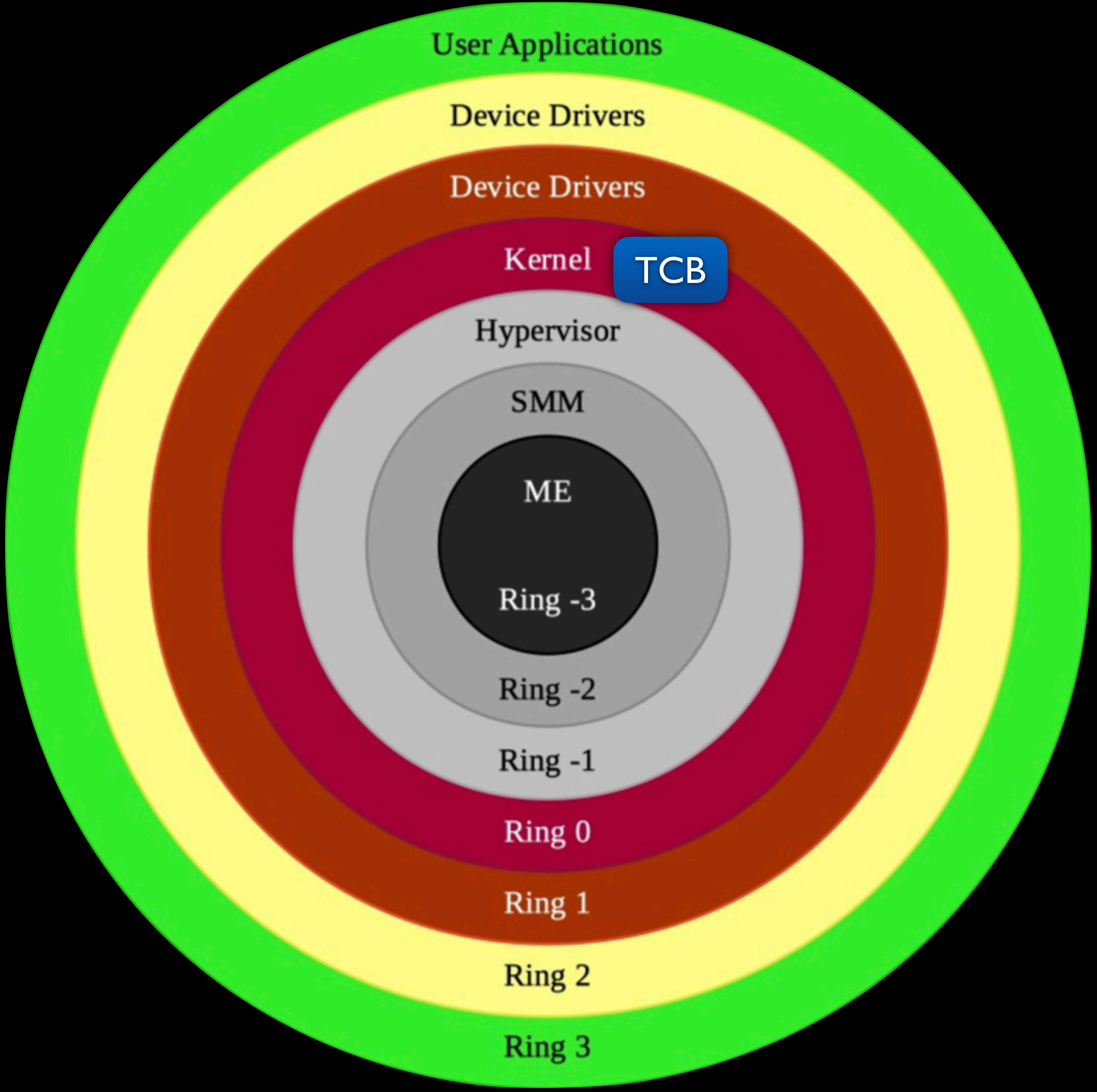
- A Reference Monitor is an abstract security component that enforces access control rules. It ensures that every access attempt to system resources complies with security policies.
- The TCB is the *entire collection of system components that enforce and maintain security, including the Reference Monitor.*

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

The classical *ring model*, *updated!*

Other rings

-1	**Hypervisor**	Allow guest OS ”ring 0”
-2	**System Management Mode (SMM)**	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 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 <u>from</u> *host operating system* <u>against</u> *guest operating systems* in a virtual machine environment

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*	

How do you create security in the OS?

How do you create security in the OS?

- Follow well-known design principles
- Use well-known pattern
- Use ordinary developer best-practises
- Decide to use principles, e.g. secure-by-default
- Use programming languages that support secure practices
- Have the design and implementation evaluated and certified

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/

Comparing security in Operating systems (1/6)

- When in time was the system developed?
  - What was the state-of-the-art at that time?
  - What trends where currently in fashion?
  - What languages was available for creating the operating system?

OS	Developed	Released
Unix	1969	1971
Mach Kernel	1985	1986
Windows NT	1988	1993
Linux	1991	1991
MacOS	Late 1990's	2001
iOS	2005-07	2007
Android	2003	2008

Comparing security in Operating systems (2/6)

- In what language is an operating system developed?
  - **Unix**: assembler (1969), C (1973)
  - **Windows**: C/C++, C#&.NET
  - **Linux**: C/C++/asm, Python/bash/perl, Rust (2022-)
  - **MacOS X/iOS**: C/C++/Objective-C (1999–2010), Swift (2014-)
  - **Android**: C/C++/Java (2008–2016), Kotlin (2014-), Rust (2020-)

Comparing security in Operating systems (3/6)

"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?

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?

Yes, more code is often 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.

There is often a relation between complexity, size of program and bugs

Code Quality Level	Bugs per 1,000 SLOC (Defect Density)
Typical commercial software	10 - 50 bugs
Well-tested open-source software	1 - 5 bugs
Mission-critical software (NASA, avionics, medical, etc.)	0.1 - 1 bugs
High-reliability software (e.g., space systems, nuclear control)	< 0.1 bugs

There is often a relation between complexity, size of program and bugs

Comparing security in Operating systems (6/6)

- 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,
    - local privilege escalations (LPE),
    - execution of remotely uploaded malicious code (RCE),
    - etc

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, as a result, affect the security in *applications* and *subsystems* (databases, middle ware, etc)

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

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

Some concepts and terms

Information disclosure bugs

Time related bugs

Memory corruption bugs

File inclusion

File/object permissions

Directory traversal

Stack smashing *Stack overflow*

Heap overflow

Race conditions

Time-of-Check to Time-of-Use (TOCTTOU)

Some concepts and terms

Vulnerability

Exploit

0day exploit

1day exploit/Nday

Foreverday exploit

vulnerability *unknown* to vendor & no patch *available*

vulnerability *disclosed* but *not widely patched*

unpatchable or *unsupported systems*

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
  - Used by security companies for marketing their knowledge and brand

DIRTY COW

FORESHADOW

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.
- Attack surface - exposed parts that an attacker can reach, i.e. all the different attack vectors
- Reverse engineering (RE) - To re-create the original design by observing the final result, in computer science - to re-create some source code by examing a binary.

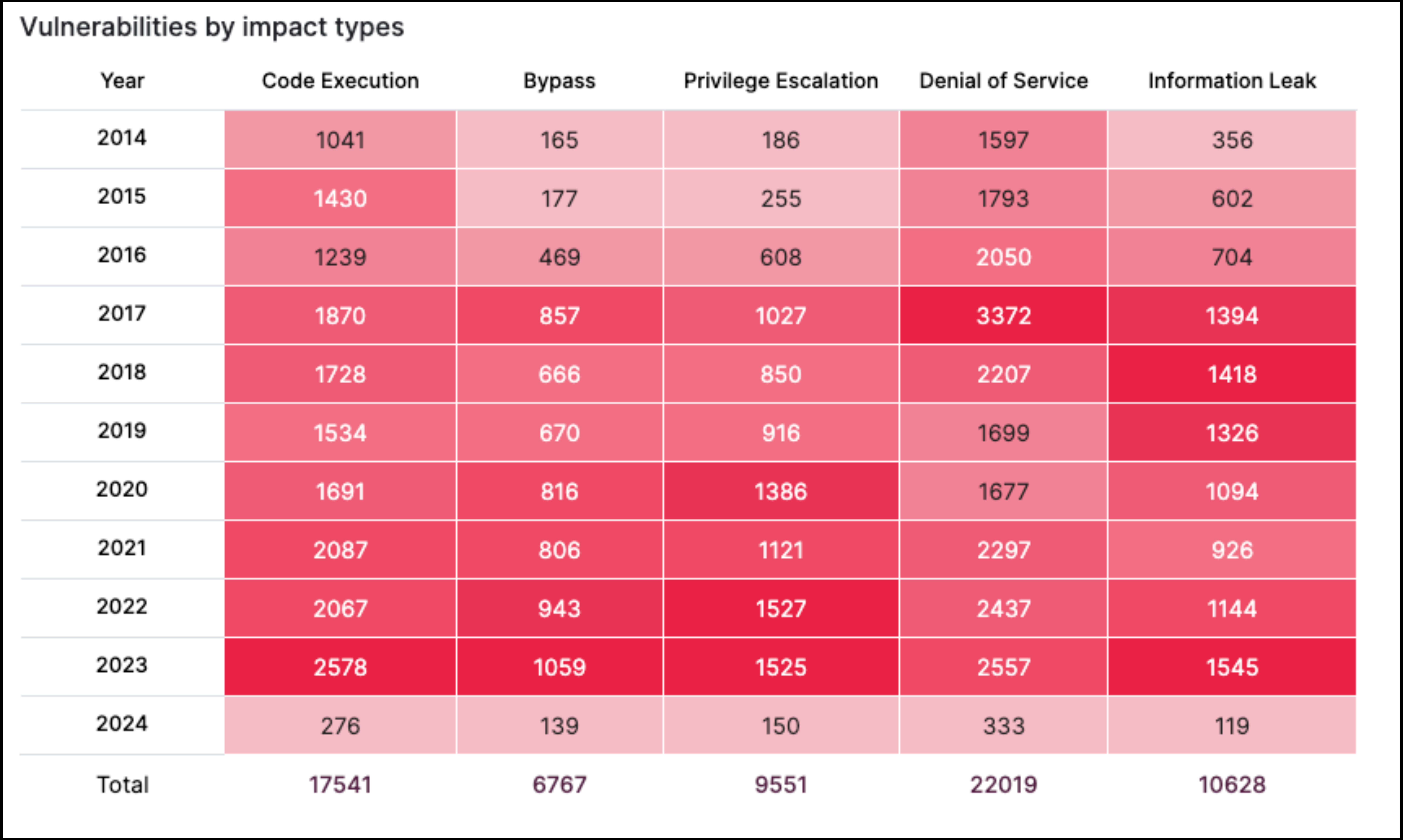
Year	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	File Inclusion	CSRF	XXE	SSRF	Open Redirect	Input Validation
2015	1037	1104	221	776	152	6	249	50	8	46	379
2016	1180	1173	97	497	99	12	87	41	16	33	519
2017	2478	1542	505	1500	282	155	334	109	57	97	936
2018	2084	1731	503	2042	569	112	479	188	118	85	1248
2019	1205	2030	544	2387	488	126	560	137	103	121	908
2020	1218	1879	465	2201	436	110	415	119	131	100	815
2021	1664	2530	742	2724	548	91	520	126	192	133	678
2022	1863	3369	1788	3404	729	96	769	126	230	146	779
2023	1673	2298	2121	5136	769	116	1392	127	243	181	627
2024	1781	2534	2650	7455	945	257	1435	112	378	121	133
2025	237	207	416	1683	134	46	426	16	58	13	0
Total	16420	20397	10052	29805	5151	1127	6666	1151	1534	1076	7022

https://www.cvedetails.com/vulnerabilities-by-types.php

Vulnerabilities by type

Input validation
Open redirect
SSRF
XXE
CSRF
File inclusion
Directory traversal
XSS
Overflow
Memory corruption
SQL injection

🟦	Overflow
🟧	Memory corruption
🟩	SQL injection
🟦	XSS
🟦	Directory traversal
🟨	File inclusion
🟩	CSRF
🟧	XXE
🟦	SSRF
🟥	Open redirect
🟦	Input validation



Vulnerabilities by impact types

Year	Code Execution	Bypass	Privilege Escalation	Denial of Service	Information Leak
2014	1041	165	186	1597	356
2015	1430	177	255	1793	602
2016	1239	469	608	2050	704
2017	1870	857	1027	3372	1394
2018	1728	666	850	2207	1418
2019	1534	670	916	1699	1326
2020	1691	816	1386	1677	1094
2021	2087	806	1121	2297	926
2022	2067	943	1527	2437	1144
2023	2578	1059	1525	2557	1545
2024	276	139	150	333	119
Total	17541	6767	9551	22019	10628

https://www.cvedetails.com/vulnerabilities-by-types.php

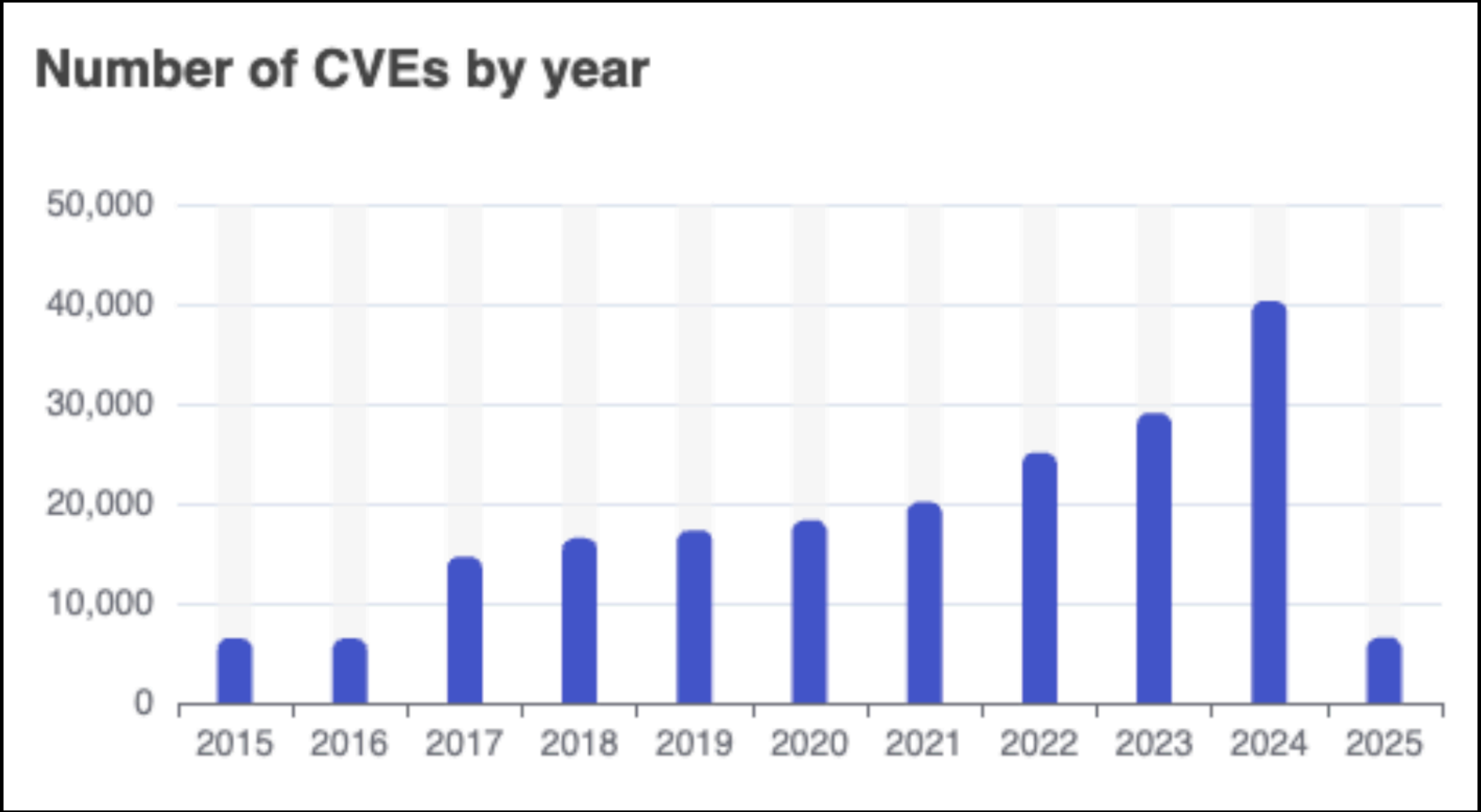
Intro - *the basics*

- Many vulnerabilities also gets ”formal name”, i.e. CVE*, and a scoring CVSS**
  - e.g. CVE-2024-21762 (A out-of-bounds write in Fortinet FortiOS) with CVSS score of 9.8
- A CVE is assigned by a CNA, a *CVE numbering authority*
- All issued CVE is stored in central database

- Not all vulnerabilities gets an CVE
- Not all issued CVE numbers ends up being used in public vulnerability info

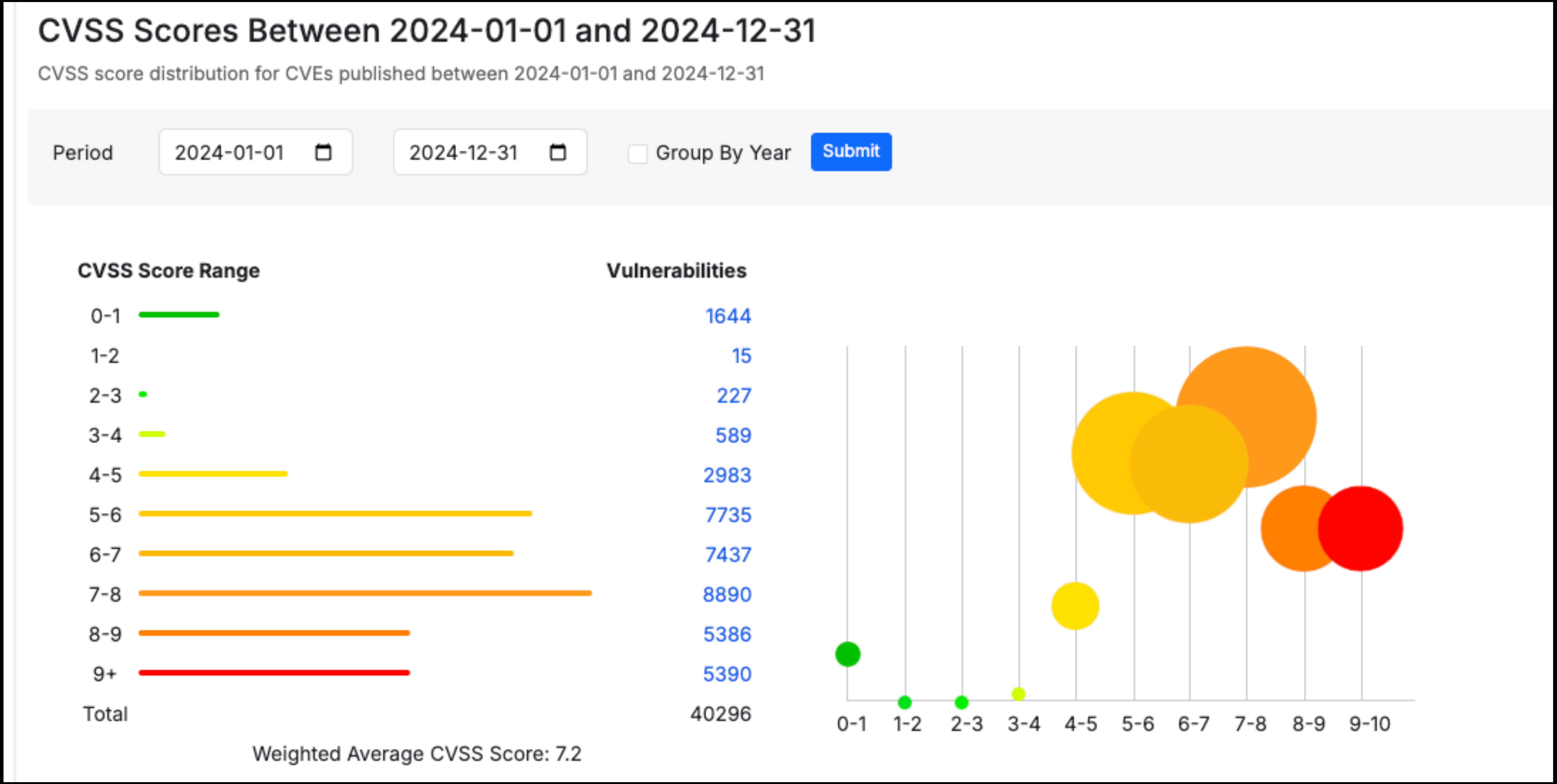
* ”Common Vulnerabilities and Exposures;” https://cve.mitre.org/ ** https://www.first.org/cvss/specification-document**

Intro - *the basics*



https://www.cvedetails.com/browse-by-date.php

Intro - *the basics*



CVSS Scores Between 2024-01-01 and 2024-12-31

CVSS score distribution for CVEs published between 2024-01-01 and 2024-12-31

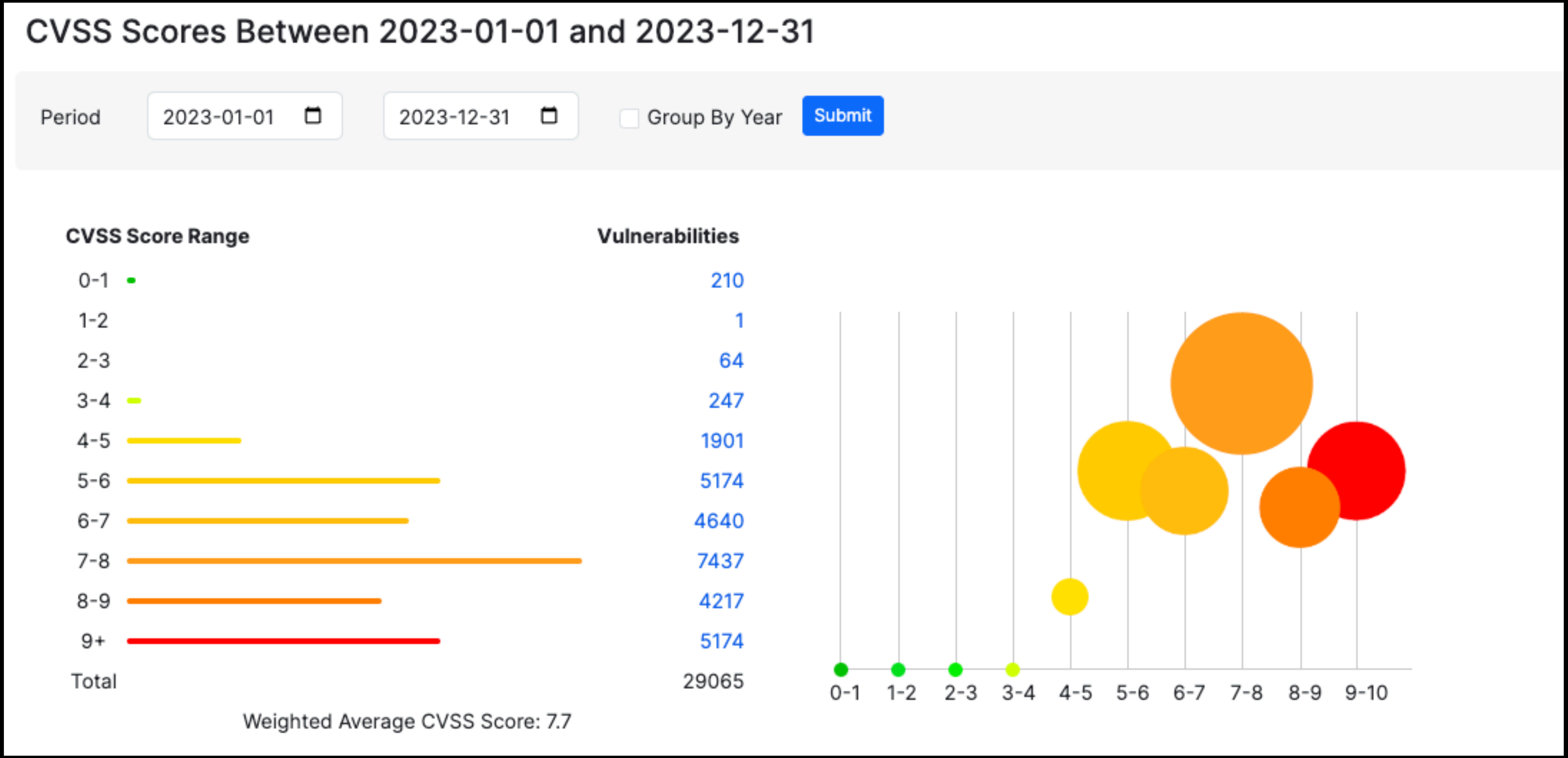
Period 2024-01-01 2024-12-31 ☐ Group By Year Submit

CVSS Score Range	Vulnerabilities
0-1	1644
1-2	15
2-3	227
3-4	589
4-5	2983
5-6	7735
6-7	7437
7-8	8890
8-9	5386
9+	5390
Total	40296

Weighted Average CVSS Score: 7.2

https://www.cvedetails.com/cvss-score-charts.php?fromform=1&vendor_id=&product_id=&startdate=2024-01-01&enddate=2024-12-31

Intro - *the basics*



CVSS Scores Between 2023-01-01 and 2023-12-31

Period 2023-01-01 2023-12-31 Group By Year Submit

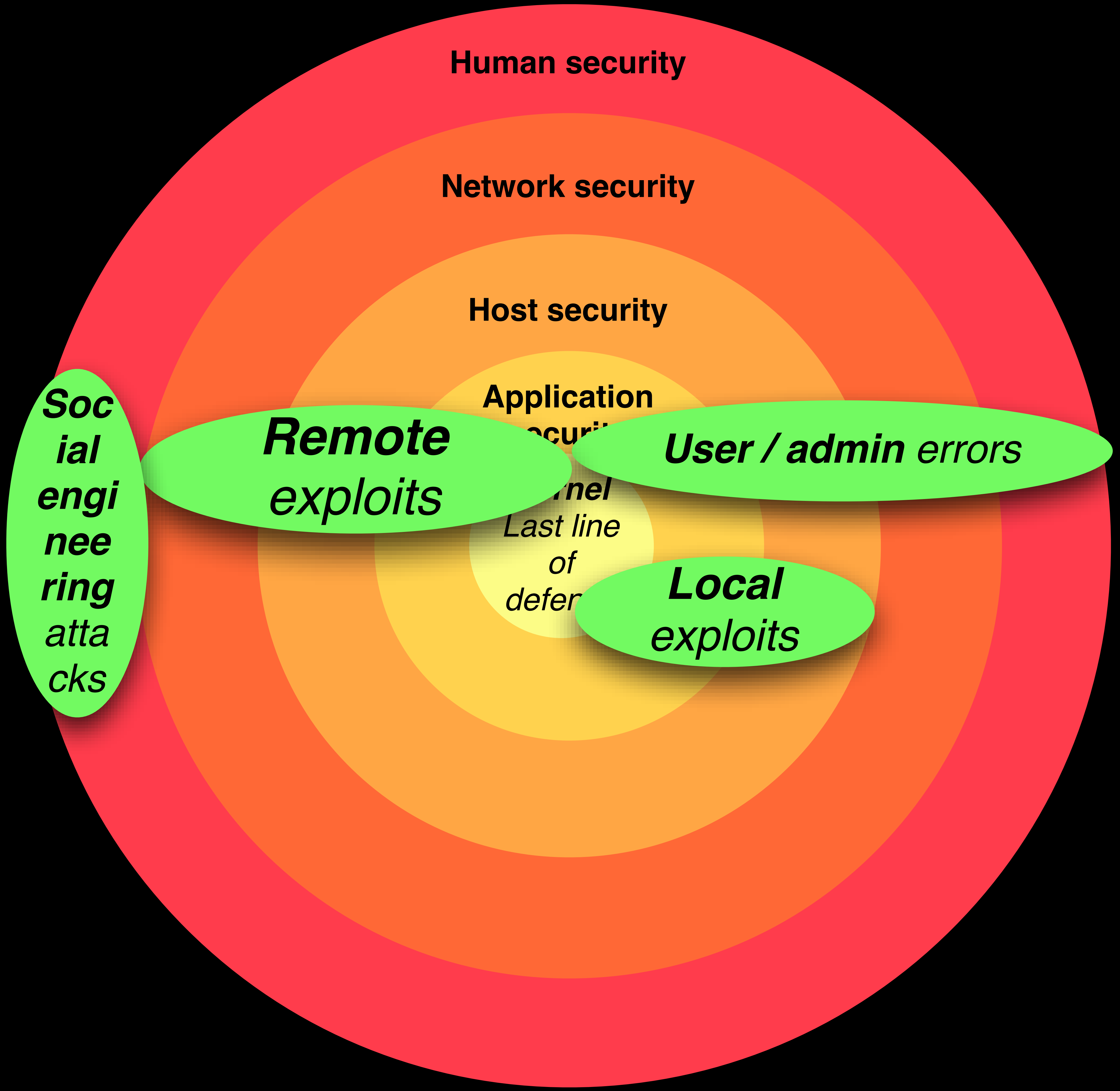
CVSS Score Range	Vulnerabilities
0-1	210
1-2	1
2-3	64
3-4	247
4-5	1901
5-6	5174
6-7	4640
7-8	7437
8-9	4217
9+	5174
Total	29065

Weighted Average CVSS Score: 7.7

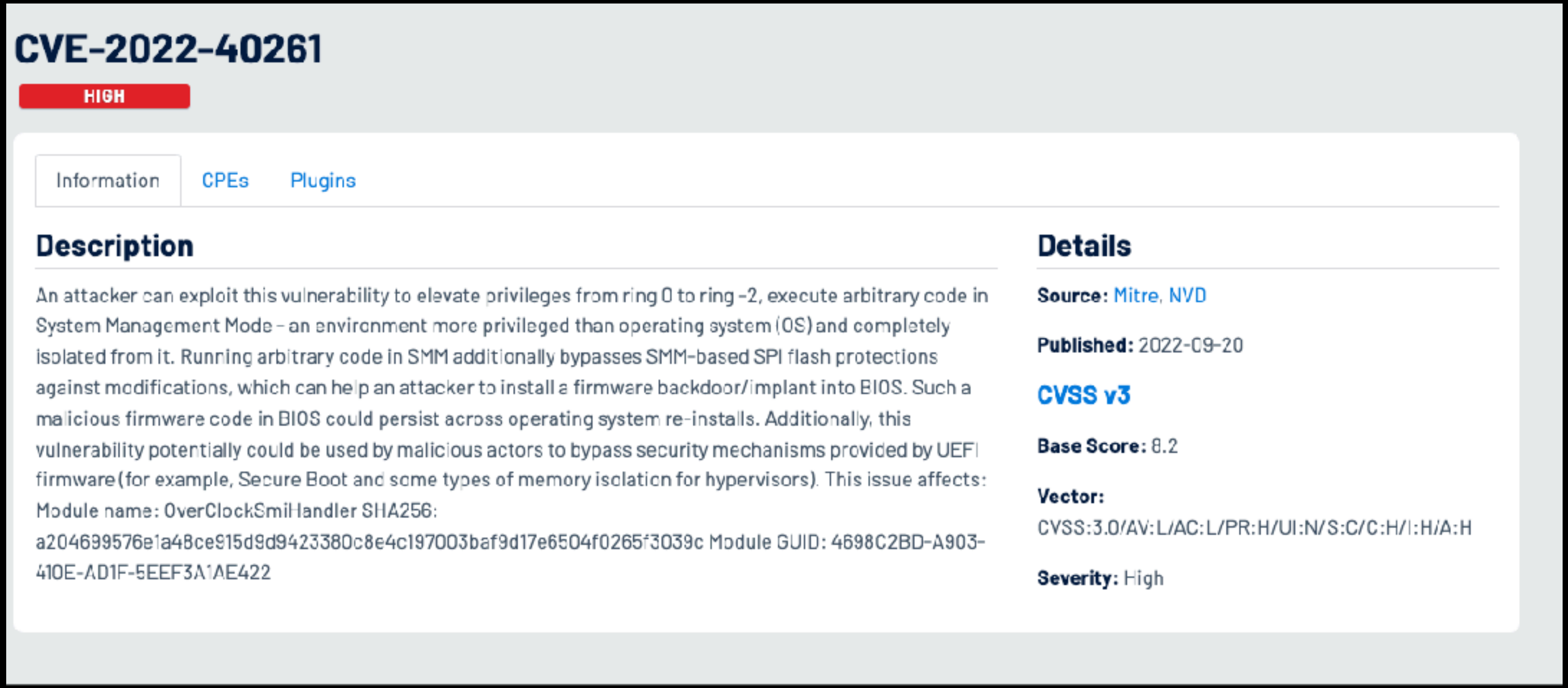
https://www.cvedetails.com/cvss-score-charts.php?fromform=1&vendor_id=&product_id=&startdate=2023-01-01&enddate=2023-12-31

Examples of vulnerabilities and attacks

Where do attacks occur?



The classical *ring model*, *updated!*



CVE-2022-40261

HIGH

Information | CPEs | Plugins

Description

An attacker can exploit this vulnerability to elevate privileges from ring 0 to ring -2, execute arbitrary code in System Management Mode - an environment more privileged than operating system (OS) and completely isolated from it. Running arbitrary code in SMM additionally bypasses SMM-based SPI flash protections against modifications, which can help an attacker to install a firmware backdoor/implant into BIOS. Such a malicious firmware code in BIOS could persist across operating system re-installs. Additionally, this vulnerability potentially could be used by malicious actors to bypass security mechanisms provided by UEFI firmware (for example, Secure Boot and some types of memory isolation for hypervisors). This issue affects: Module name: OverClockSmiHandler SHA256: a204699576e1a48ce915d9d9423380c8e4c197003baf9d17e6504f0265f3039c Module GUID: 4698C2BD-A903-410E-AD1F-5EEF3A1AE422

Details

Source: Mitre, NVD

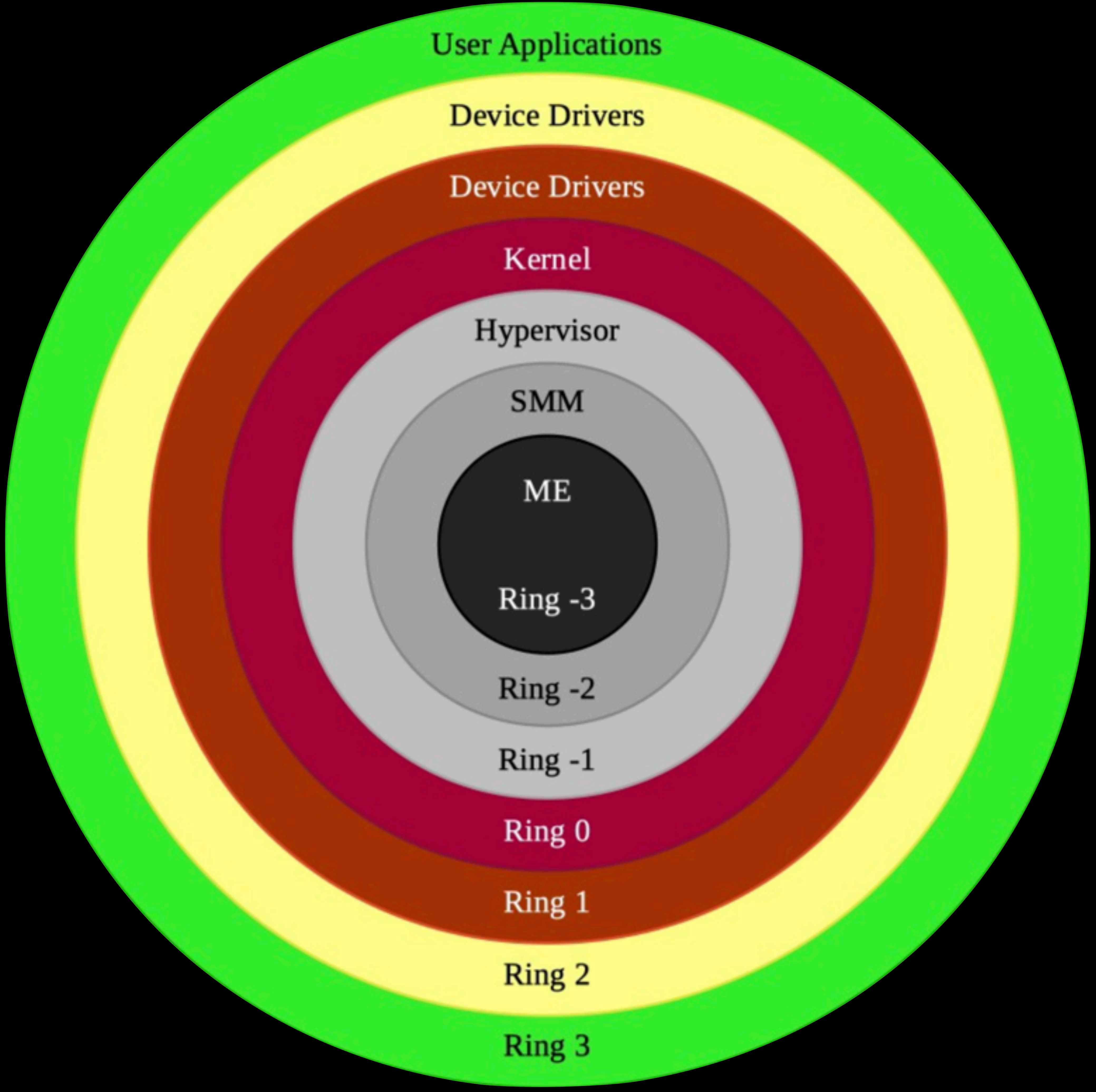
Published: 2022-09-20

CVSS v3

Base Score: 8.2

Vector: CVSS:3.0/AV:L/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H

Severity: High



https://www.tenable.com/cve/CVE-2022-40261

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

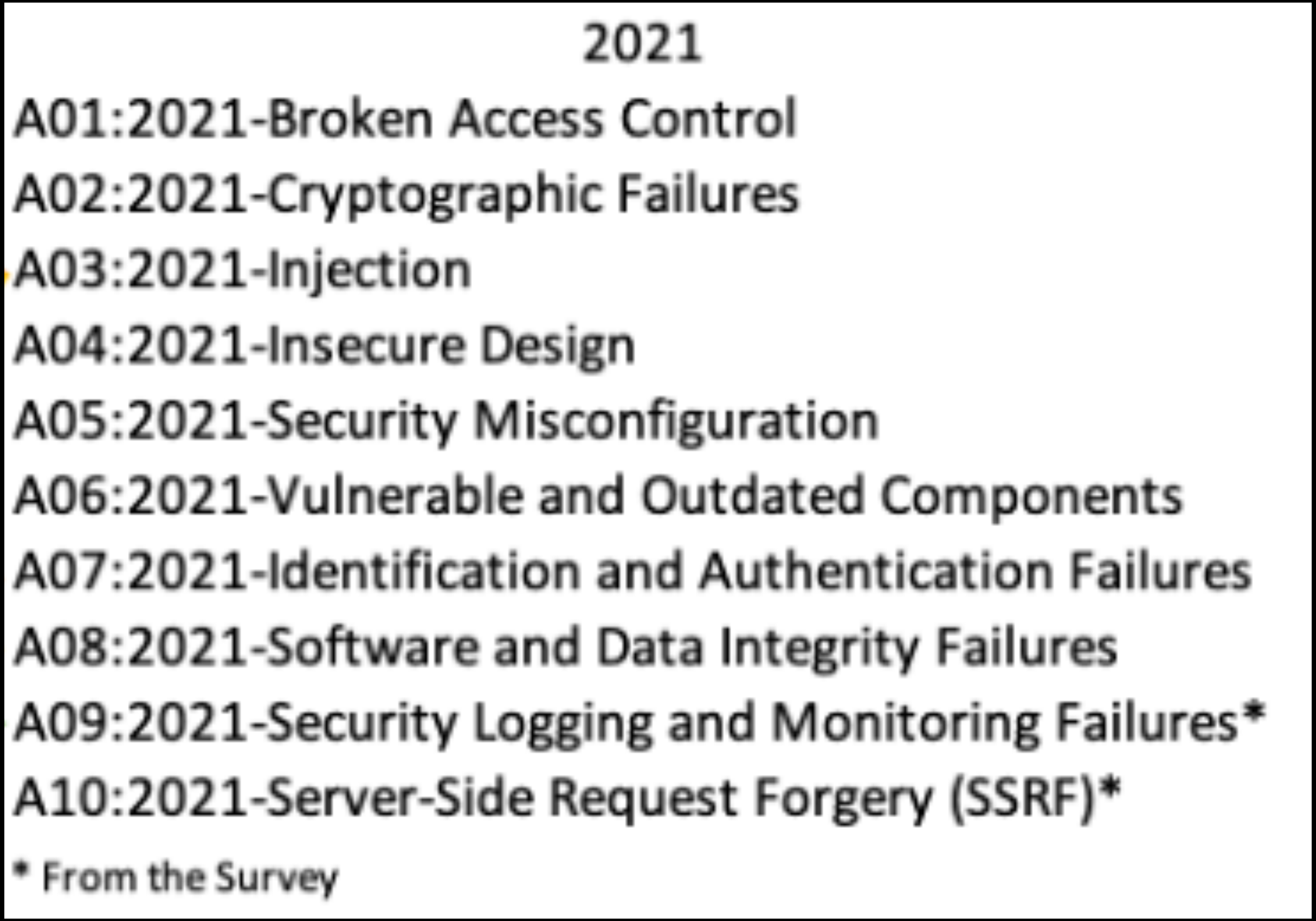
* An *mbuf* is a basic unit of memory management in the kernel IPC subsystem

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

Most common attacks?
OWASP top-10 list



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

https://owasp.org/www-project-top-ten/

Most common attacks?

2017

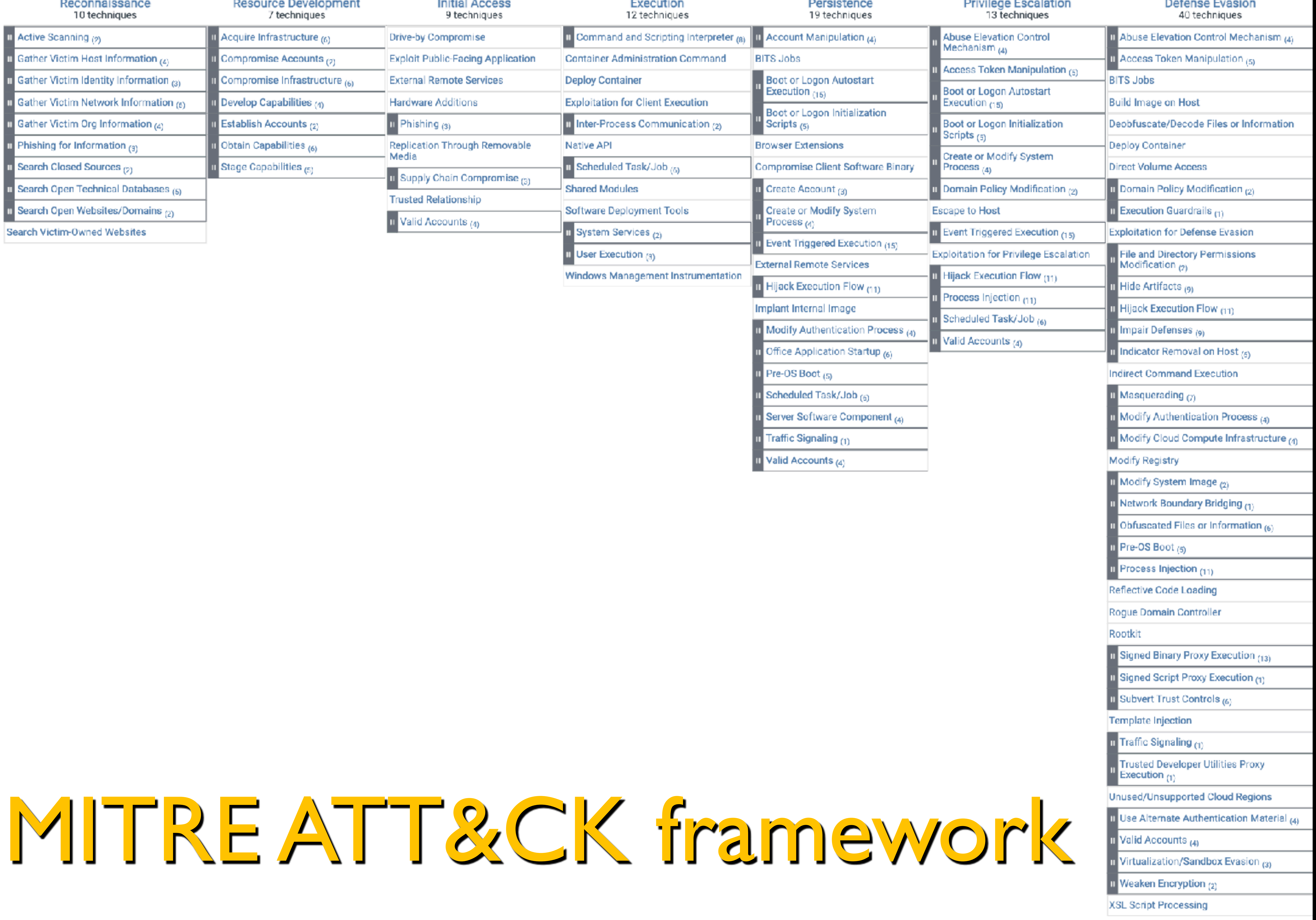
- A01:2017-Injection
- A02:2017-Broken Authentication
- A03:2017-Sensitive Data Exposure
- A04:2017-XML External Entities (XXE)
- A05:2017-Broken Access Control
- A06:2017-Security Misconfiguration
- A07:2017-Cross-Site Scripting (XSS)
- A08:2017-Insecure Deserialization
- A09:2017-Using Components with Known Vulnerabilities
- A10:2017-Insufficient Logging & Monitoring

2021

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

* From the Survey

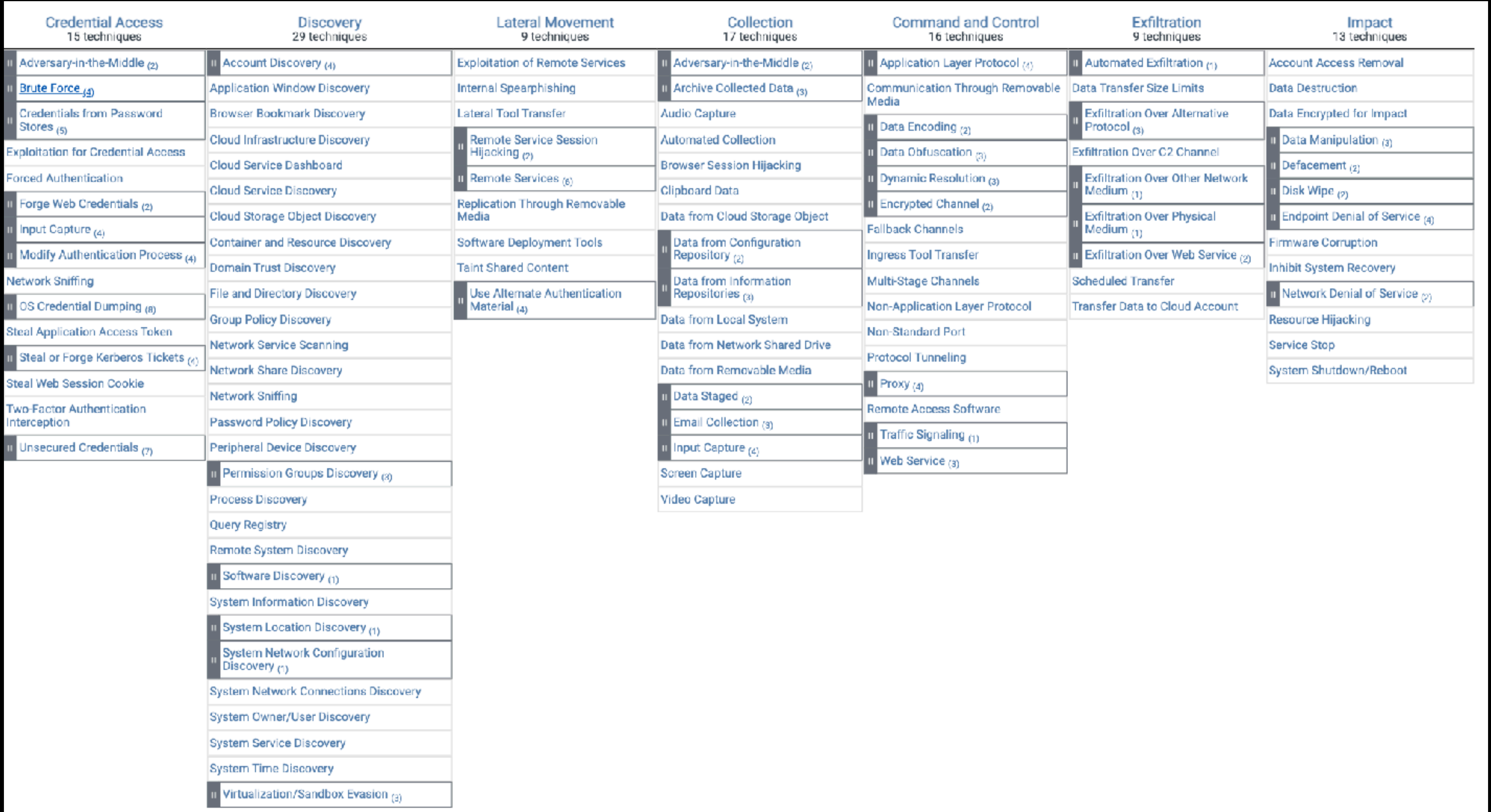
OWASP top-10 list



Reconnaissance 10 techniques	Resource Development 7 techniques	Initial Access 9 techniques	Execution 12 techniques	Persistence 19 techniques	Privilege Escalation 13 techniques	Defense Evasion 40 techniques
Active Scanning (3)	Acquire Infrastructure (6)	Drive-by Compromise	Command and Scripting Interpreter (8)	Account Manipulation (4)	Abuse Elevation Control Mechanism (4)	Abuse Elevation Control Mechanism (4)
Gather Victim Host Information (4)	Compromise Accounts (2)	Exploit Public-Facing Application	Container Administration Command	BITS Jobs	Access Token Manipulation (5)	Access Token Manipulation (5)
Gather Victim Identity Information (3)	Compromise Infrastructure (6)	External Remote Services	Deploy Container	Boot or Logon Autostart Execution (15)	Boot or Logon Autostart Execution (15)	BITS Jobs
Gather Victim Network Information (6)	Develop Capabilities (4)	Hardware Additions	Exploitation for Client Execution	Boot or Logon Initialization Scripts (5)	Boot or Logon Initialization Scripts (5)	Build Image on Host
Gather Victim Org Information (4)	Establish Accounts (2)	Phishing (3)	Inter-Process Communication (2)	Browser Extensions	Create or Modify System Process (4)	Deobfuscate/Decode Files or Information
Phishing for Information (3)	Obtain Capabilities (6)	Replication Through Removable Media	Native API	Compromise Client Software Binary	Domain Policy Modification (2)	Deploy Container
Search Closed Sources (2)	Stage Capabilities (5)	Supply Chain Compromise (3)	Scheduled Task/Job (5)	Create Account (3)	Escape to Host	Direct Volume Access
Search Open Technical Databases (5)		Trusted Relationship	Shared Modules	Create or Modify System Process (4)	Event Triggered Execution (15)	Domain Policy Modification (2)
Search Open Websites/Domains (2)		Valid Accounts (4)	Software Deployment Tools	Event Triggered Execution (15)	Exploitation for Privilege Escalation	Execution Guardrails (1)
Search Victim-Owned Websites			System Services (2)	External Remote Services	Hijack Execution Flow (11)	Exploitation for Defense Evasion
			User Execution (3)	Hijack Execution Flow (11)	Process Injection (11)	File and Directory Permissions Modification (2)
			Windows Management Instrumentation	Implant Internal Image	Scheduled Task/Job (6)	Hide Artifacts (9)
				Modify Authentication Process (4)	Valid Accounts (4)	Hijack Execution Flow (11)
				Office Application Startup (6)		Impair Defenses (9)
				Pre-OS Boot (5)		Indicator Removal on Host (5)
				Scheduled Task/Job (5)		Indirect Command Execution
				Server Software Component (4)		Masquerading (7)
				Traffic Signaling (1)		Modify Authentication Process (4)
				Valid Accounts (4)		Modify Cloud Compute Infrastructure (4)
						Modify Registry
						Modify System Image (2)
						Network Boundary Bridging (1)
						Obfuscated Files or Information (6)
						Pre-OS Boot (5)
						Process Injection (11)
						Reflective Code Loading
						Rogue Domain Controller
						Rootkit
						Signed Binary Proxy Execution (13)
						Signed Script Proxy Execution (1)
						Subvert Trust Controls (6)
						Template Injection
						Traffic Signaling (1)
						Trusted Developer Utilities Proxy Execution (1)
						Unused/Unsupported Cloud Regions
						Use Alternate Authentication Material (4)
						Valid Accounts (4)
						Virtualization/Sandbox Evasion (3)
						Weaken Encryption (2)
						XSL Script Processing

MITRE ATT&CK framework

https://attack.mitre.org/

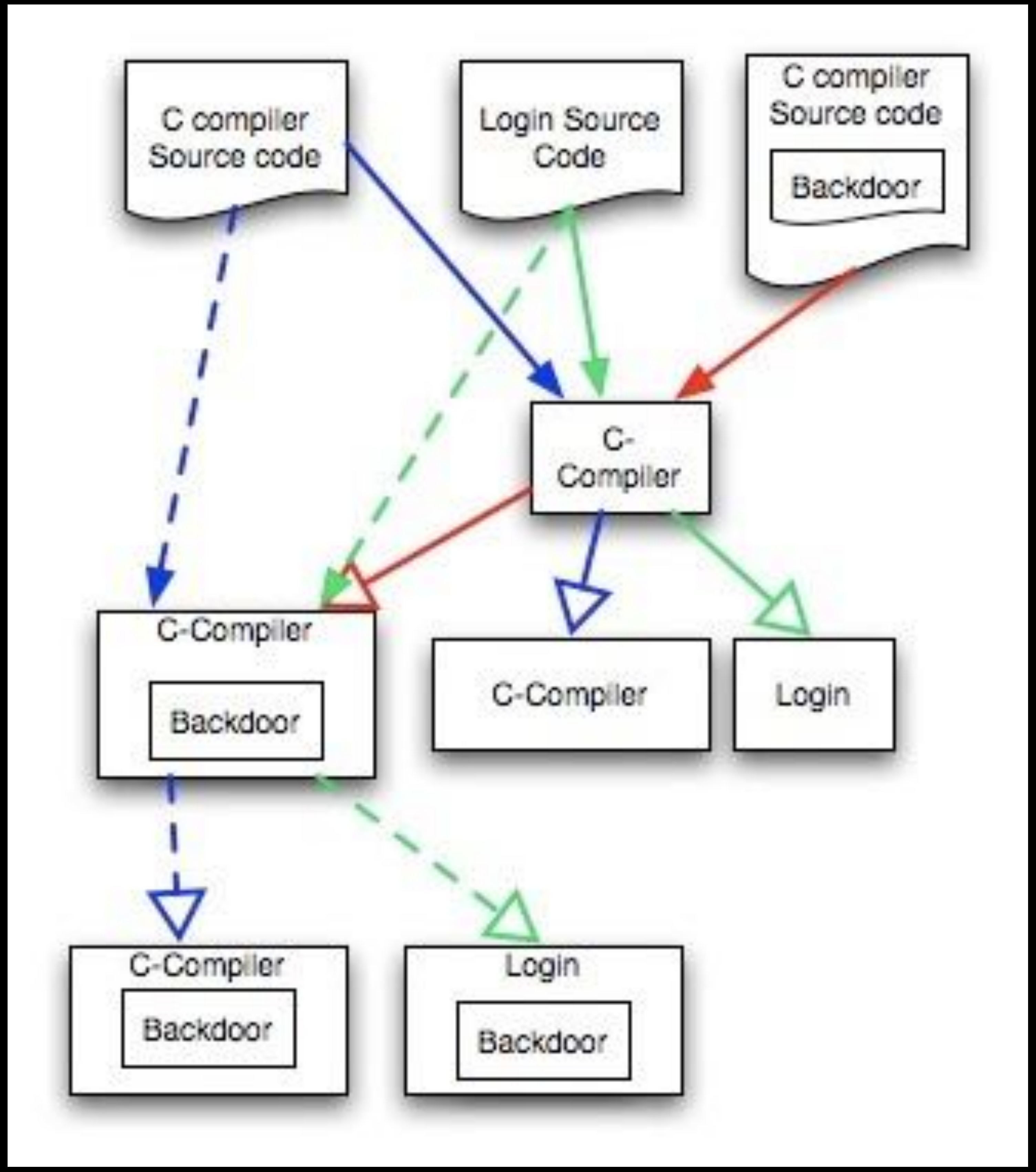


Credential Access 15 techniques	Discovery 29 techniques	Lateral Movement 9 techniques	Collection 17 techniques	Command and Control 16 techniques	Exfiltration 9 techniques	Impact 13 techniques
Adversary-in-the-Middle (2)	Account Discovery (4)	Exploitation of Remote Services	Adversary-in-the-Middle (2)	Application Layer Protocol (4)	Automated Exfiltration (1)	Account Access Removal
Brute Force (4)	Application Window Discovery	Internal Spearphishing	Archive Collected Data (3)	Communication Through Removable Media	Data Transfer Size Limits	Data Destruction
Credentials from Password Stores (5)	Browser Bookmark Discovery	Lateral Tool Transfer	Audio Capture	Data Encoding (2)	Exfiltration Over Alternative Protocol (3)	Data Encrypted for Impact
Exploitation for Credential Access	Cloud Infrastructure Discovery	Remote Service Session Hijacking (2)	Automated Collection	Data Obfuscation (3)	Exfiltration Over C2 Channel	Data Manipulation (3)
Forced Authentication	Cloud Service Dashboard	Remote Services (6)	Browser Session Hijacking	Dynamic Resolution (3)	Exfiltration Over Other Network Medium (1)	Defacement (2)
Forge Web Credentials (2)	Cloud Service Discovery	Replication Through Removable Media	Clipboard Data	Encrypted Channel (2)	Exfiltration Over Physical Medium (1)	Disk Wipe (2)
Input Capture (4)	Cloud Storage Object Discovery	Software Deployment Tools	Data from Cloud Storage Object	Fallback Channels	Exfiltration Over Web Service (2)	Endpoint Denial of Service (4)
Modify Authentication Process (4)	Container and Resource Discovery	Taint Shared Content	Data from Configuration Repository (2)	Ingress Tool Transfer	Scheduled Transfer	Firmware Corruption
Network Sniffing	Domain Trust Discovery	Use Alternate Authentication Material (4)	Data from Information Repositories (3)	Multi-Stage Channels	Transfer Data to Cloud Account	Inhibit System Recovery
OS Credential Dumping (8)	File and Directory Discovery		Data from Local System	Non-Application Layer Protocol		Network Denial of Service (2)
Steal Application Access Token	Group Policy Discovery		Data from Network Shared Drive	Non-Standard Port		Resource Hijacking
Steal or Forge Kerberos Tickets (4)	Network Service Scanning		Data from Removable Media	Protocol Tunneling		Service Stop
Steal Web Session Cookie	Network Share Discovery		Data Staged (2)	Proxy (4)		System Shutdown/Reboot
Two-Factor Authentication Interception	Network Sniffing		Email Collection (3)	Remote Access Software		
Unsecured Credentials (7)	Password Policy Discovery		Input Capture (4)	Traffic Signaling (1)		
	Peripheral Device Discovery		Screen Capture	Web Service (3)		
	Permission Groups Discovery (3)		Video Capture			
	Process Discovery					
	Query Registry					
	Remote System Discovery					
	Software Discovery (1)					
	System Information Discovery					
	System Location Discovery (1)					
	System Network Configuration Discovery (1)					
	System Network Connections Discovery					
	System Owner/User Discovery					
	System Service Discovery					
	System Time Discovery					
	Virtualization/Sandbox Evasion (3)					

MITRE ATT&CK framework

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

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*

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 ;-)

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

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

Virtualization and isolation

sandboxes, containers, hypervisors, etc

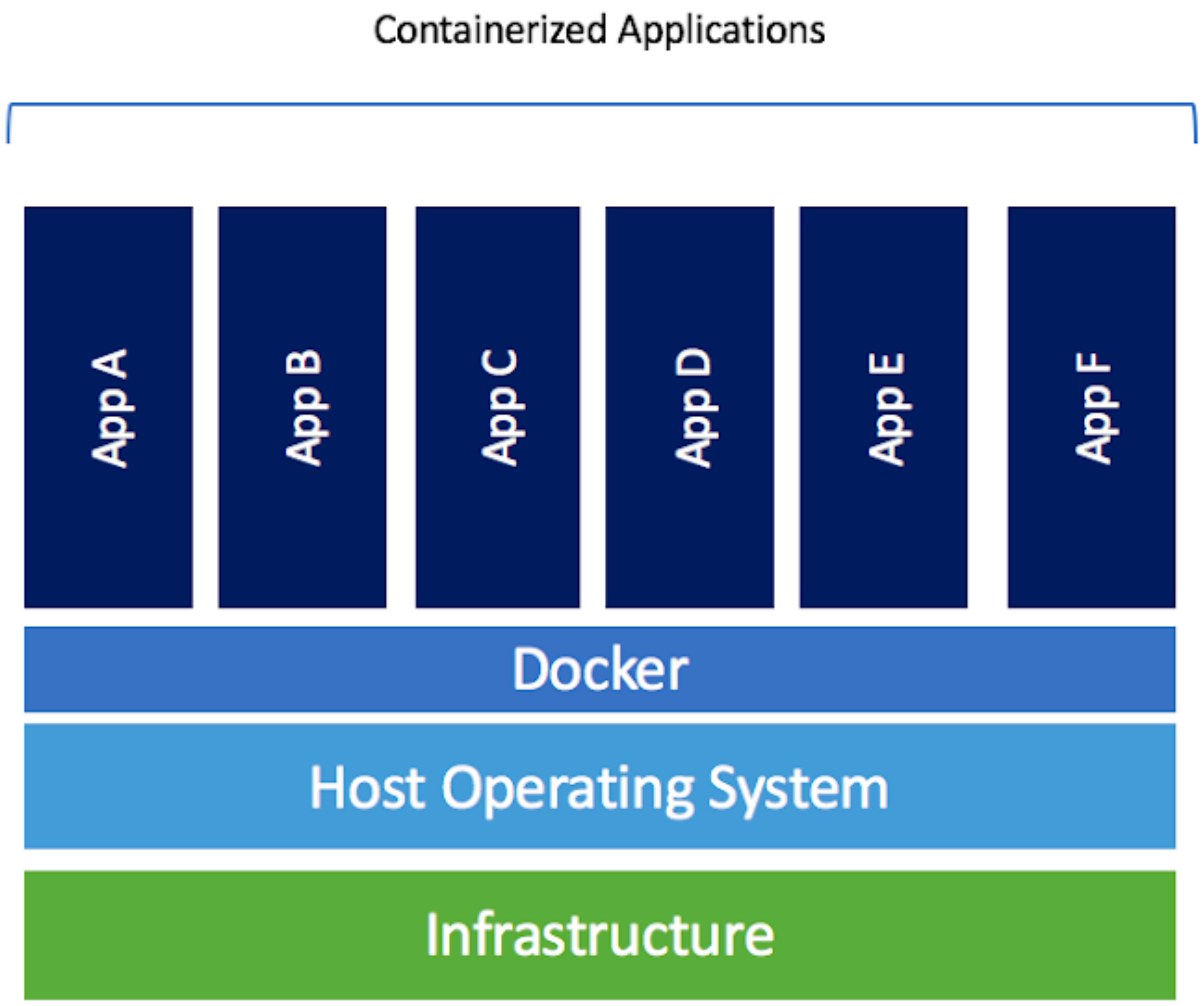
Sandboxing

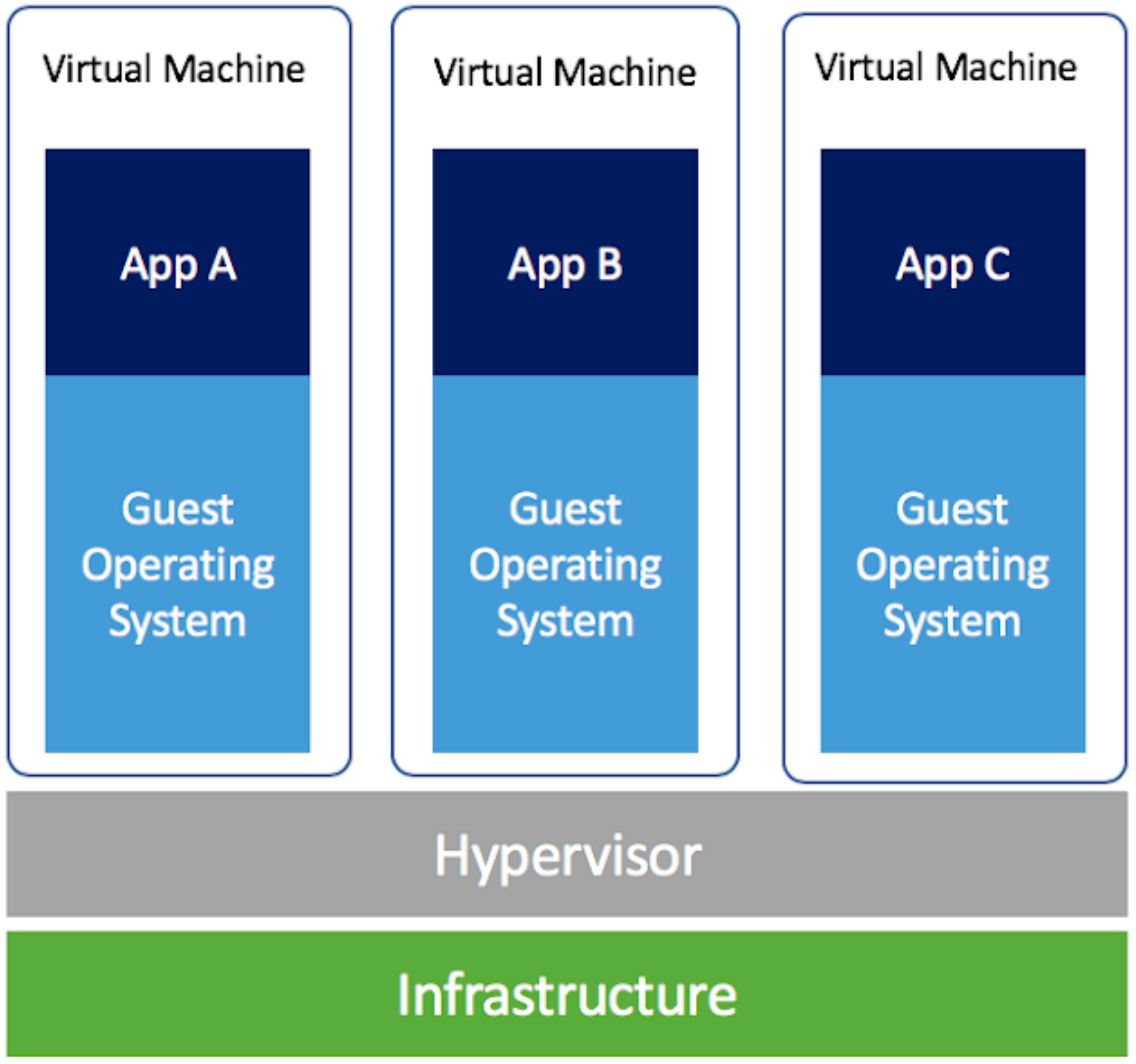
- Various types of OS supported or application supported ***sandboxing*** is good as a way to get defense-in-depth
- Create ***temporary execution environments*** for certain tasks
  - test of exe files to lure out malicious code execution
  - perform certain tasks that is more prone to attacks
  - perform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

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

VM's vs Containers vs WebAssembly

Feature	Virtual Machines	Containers	WebAssembly (Wasm)
Isolation	Full OS virtualization, strong	OS-level isolation	Sandboxed execution (virtual CPU)
Performan ce	Slower (heavy overhead)	Near-native	Near-native, optimized
Startup	Minutes	Seconds	Milliseconds
Size	GBs (full OS image)	MBs (includes OS dependencies)	KBs to MBs (minimal overhead)
System Access	Full OS access (kernel, drivers)	Shares OS kernel	No direct OS access (sandboxed)
Security	Strong (separate OS instances)	Moderate (kernel shared)	Strong (sandboxed, minimal attack surface)
Portability	Limited (OS-dependent)	Cross-platform (container runtimes)	Universal (Wasm runtimes)
Use Cases	Legacy applications, multi-OS environments	Cloud-native applications, microservices	Edge computing, serverless, high-security apps

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 firmware, drivers, kernels, applications

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*

Attacking
UEFI Runtime Services
and Linux

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

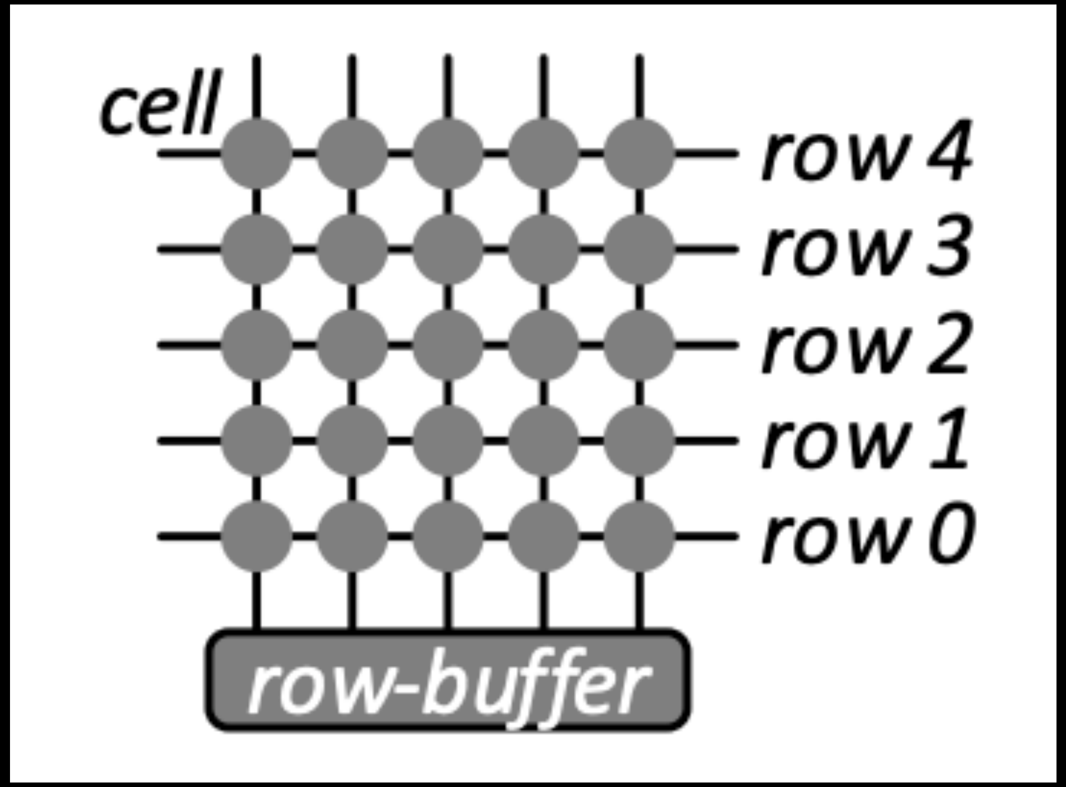
Example of attacks: *HW implants*

(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

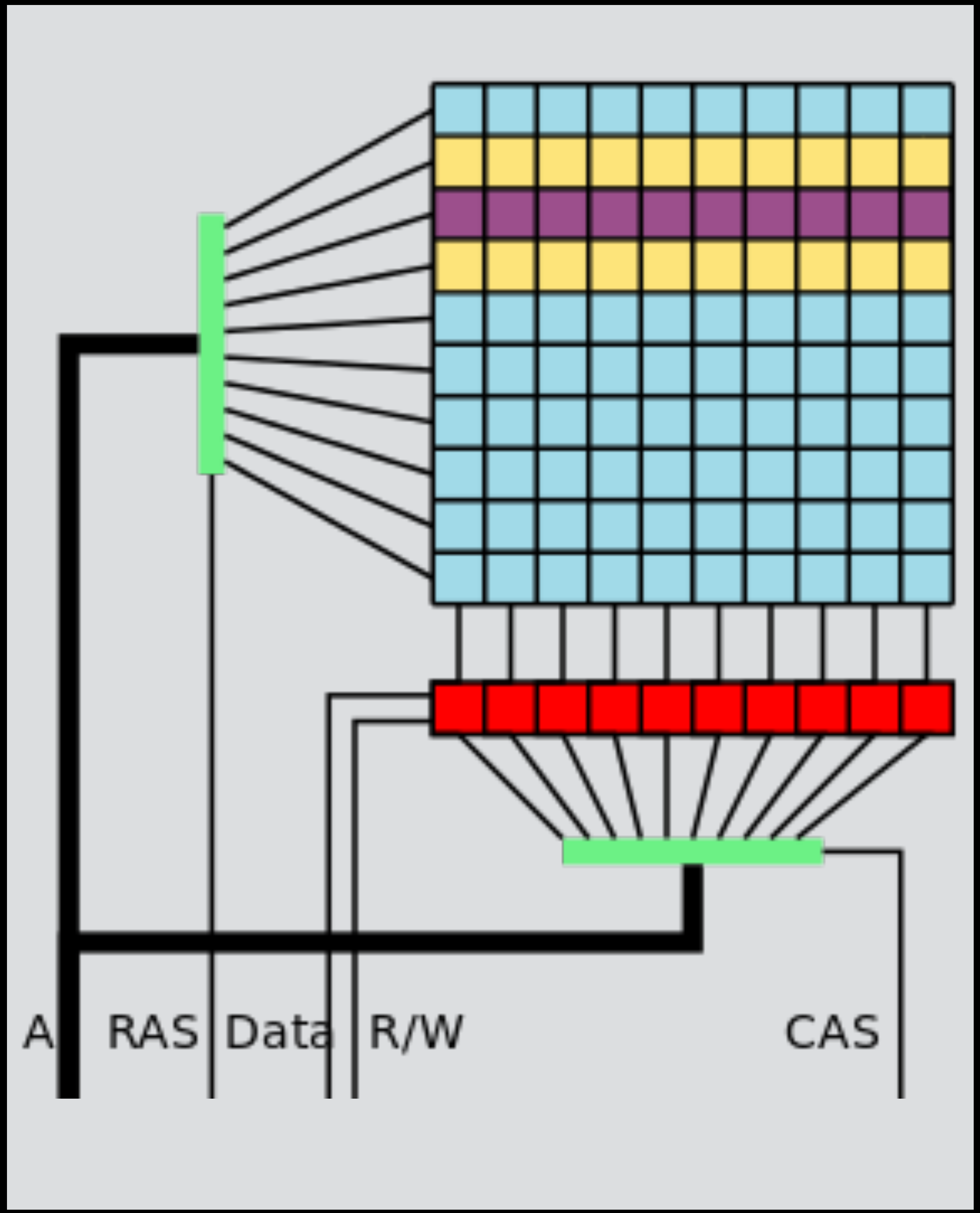
- *Rowhammer**
  - *Flipping bits without accessing them*
  - *Method of reading writing memory cells so that memory cells in adjacent rows become changed*
  - *Based on an unintended side effect in dynamic random-access memory (DRAM) that causes memory cells to leak their charges and interact electrically 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
  - Rowhammer.js (2015)
  - Blacksmith (2022)
  - Half-double (2021)
  - Zenhammer (2024, AMD architecture)
  - RISC-Hammer (2024, RISC-Y architecture)
- Hardware solutions to protect against it have been circumvented

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***
  - *Initial research published January 2018*
  - *Microarchitectural bugs in CPU*
  - *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

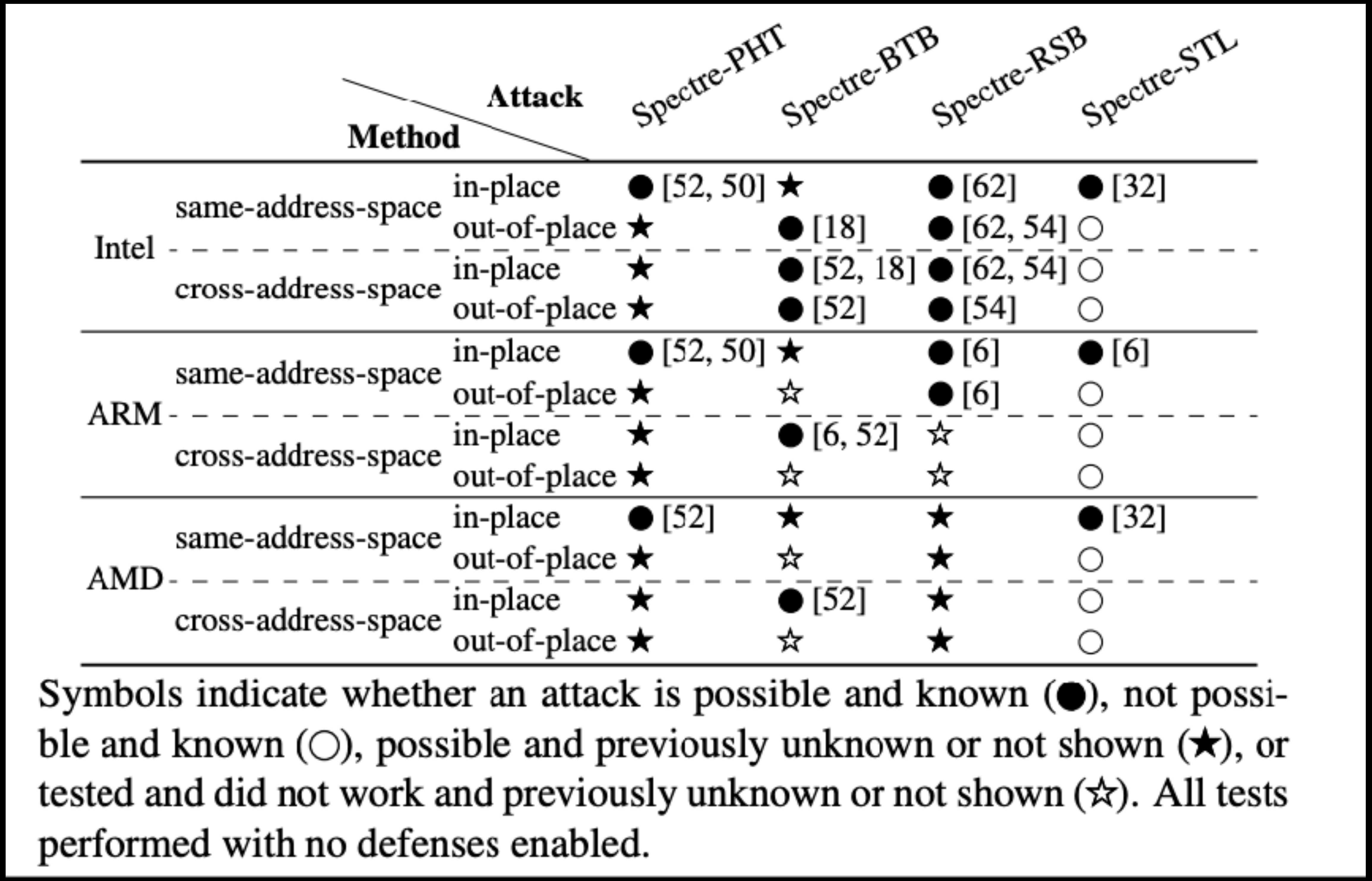
** Kocker et al "Spectre Attacks: Exploiting Speculative Execution" https://spectreattack.com/spectre.pdf

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*

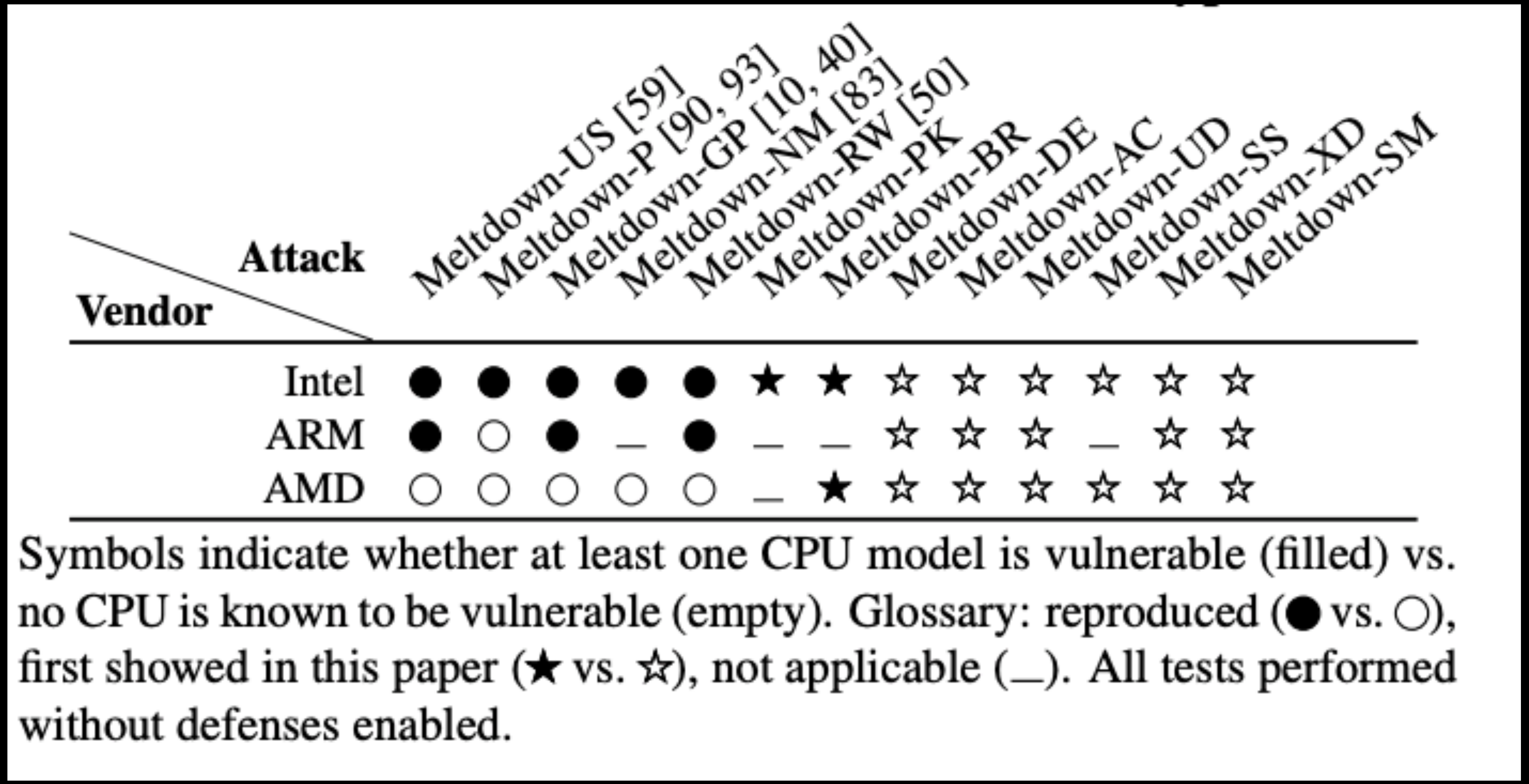
Advanced attacks

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



Vendor	Method		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.

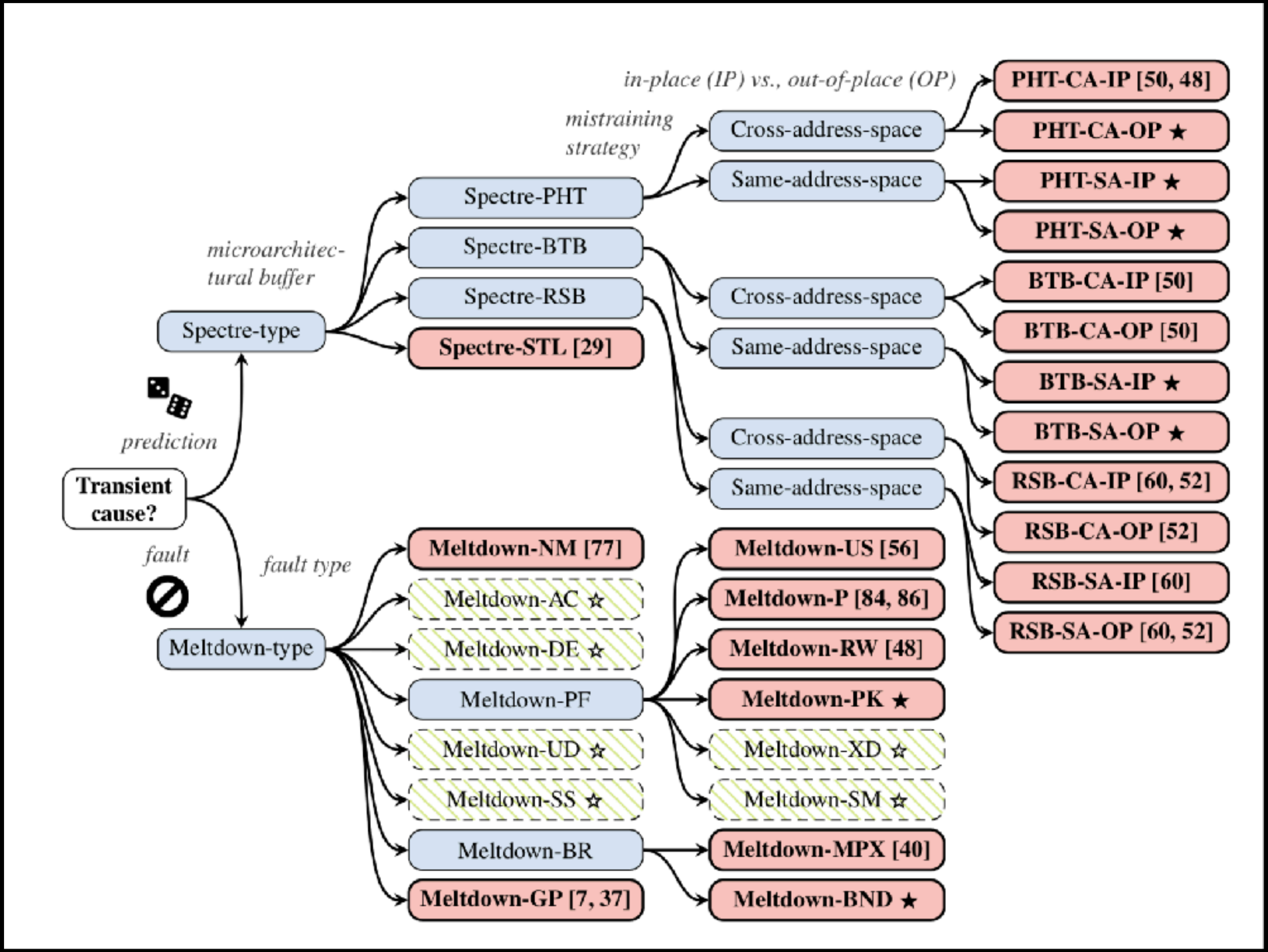


Vendor	Meltdown-US [59]	Meltdown-P [90, 93]	Meltdown-GP [10, 40]	Meltdown-NM [83]	Meltdown-RW [50]	Meltdown-PK	Meltdown-BR	Meltdown-DE	Meltdown-AC	Meltdown-UD	Meltdown-SS	Meltdown-XD	Meltdown-SM
Intel	●	●	●	●	●	★	★	☆	☆	☆	☆	☆	☆
ARM	●	○	●	_	●	_	_	☆	☆	☆	_	☆	☆
AMD	○	○	○	○	○	_	★	☆	☆	☆	☆	☆	☆

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

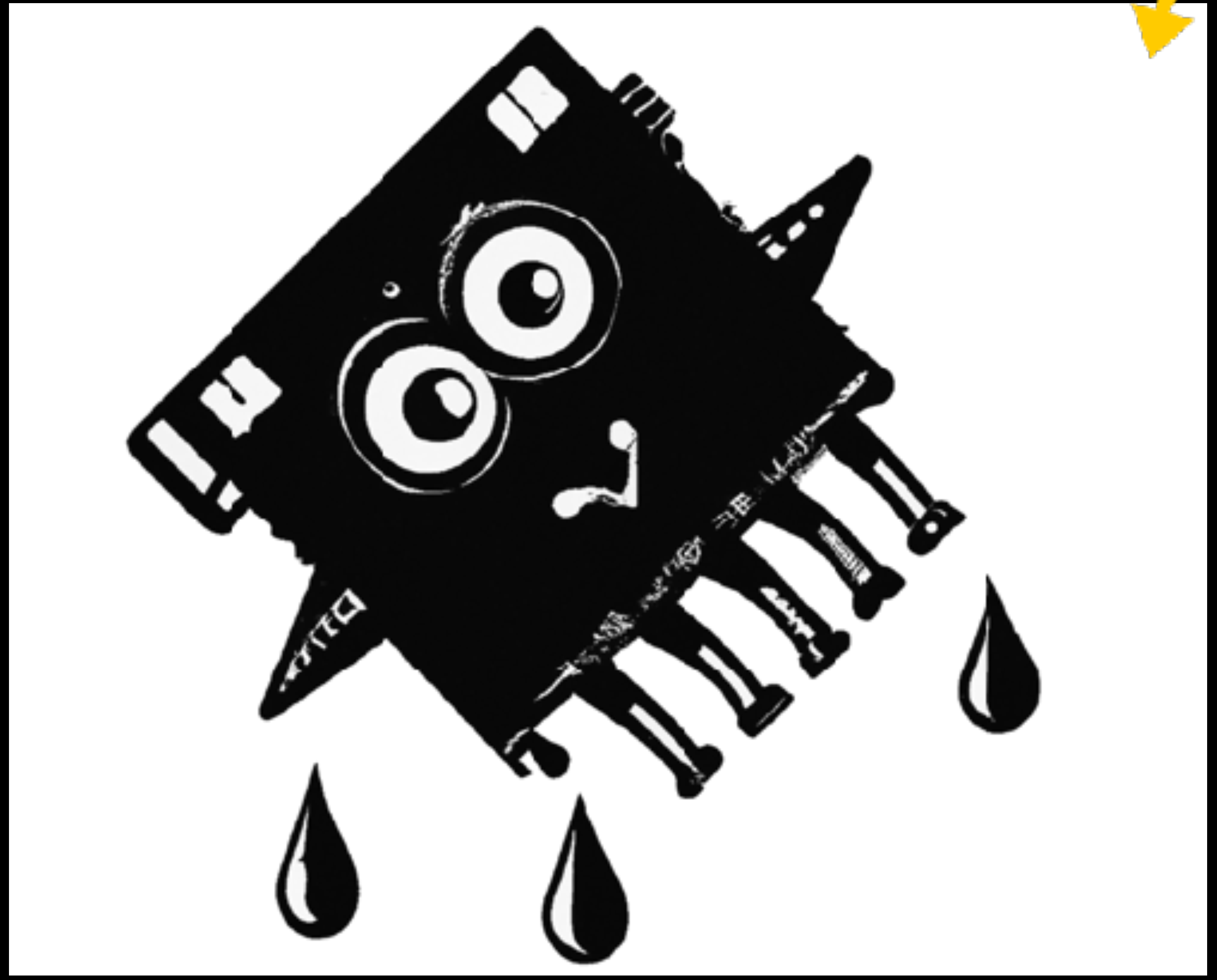
PLUNDER VOLT

FORESHADOW

MDS

ZOMBIELOAD ATTACK

RETURN OF THE LEAKING DEAD



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

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

- *Microarchitecture Data Sampling attacks*
- *Side channel attacks*
- *Timing side channel attacks*
- *Power Analysis side channel attack*

Advanced attacks

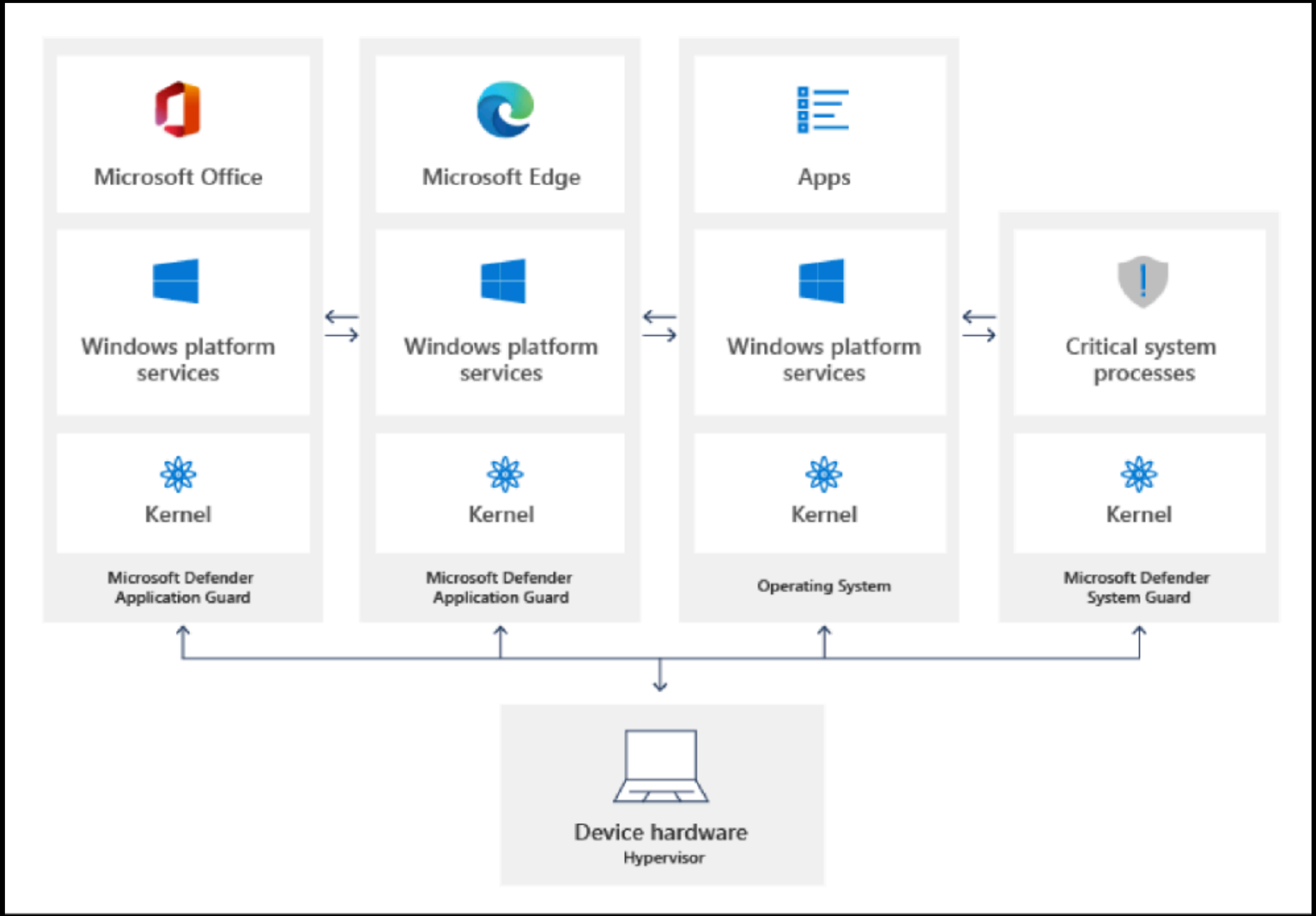
- *Attacks against Intel Management Engine*
  - *Proprietary and non-documented*
  - *Own OS (Minix!)*
  - *Reverse engineered and analysed by attackers*
  - *Found multiple vulnerabilities in Skylake & Kabylake architecture*

Examples of modern security controls

Windows Defender security features in Win 10, Win 11

Windows

- Application Guard, **WDAG**
  - App & browser control
  - Isolation browsing
- WDAC (Windows Defender Application Control) give application & driver whitelisting
- VBS (Virtualization-Based Security)
- WDAC & VBS used to be Windows Device Guard



Microsoft Office
Microsoft Edge
Apps
Windows platform services
Windows platform services
Windows platform services
Critical system processes
Kernel
Kernel
Kernel
Kernel
Microsoft Defender Application Guard
Microsoft Defender Application Guard
Operating System
Microsoft Defender System Guard
Device hardware
Hypervisor

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

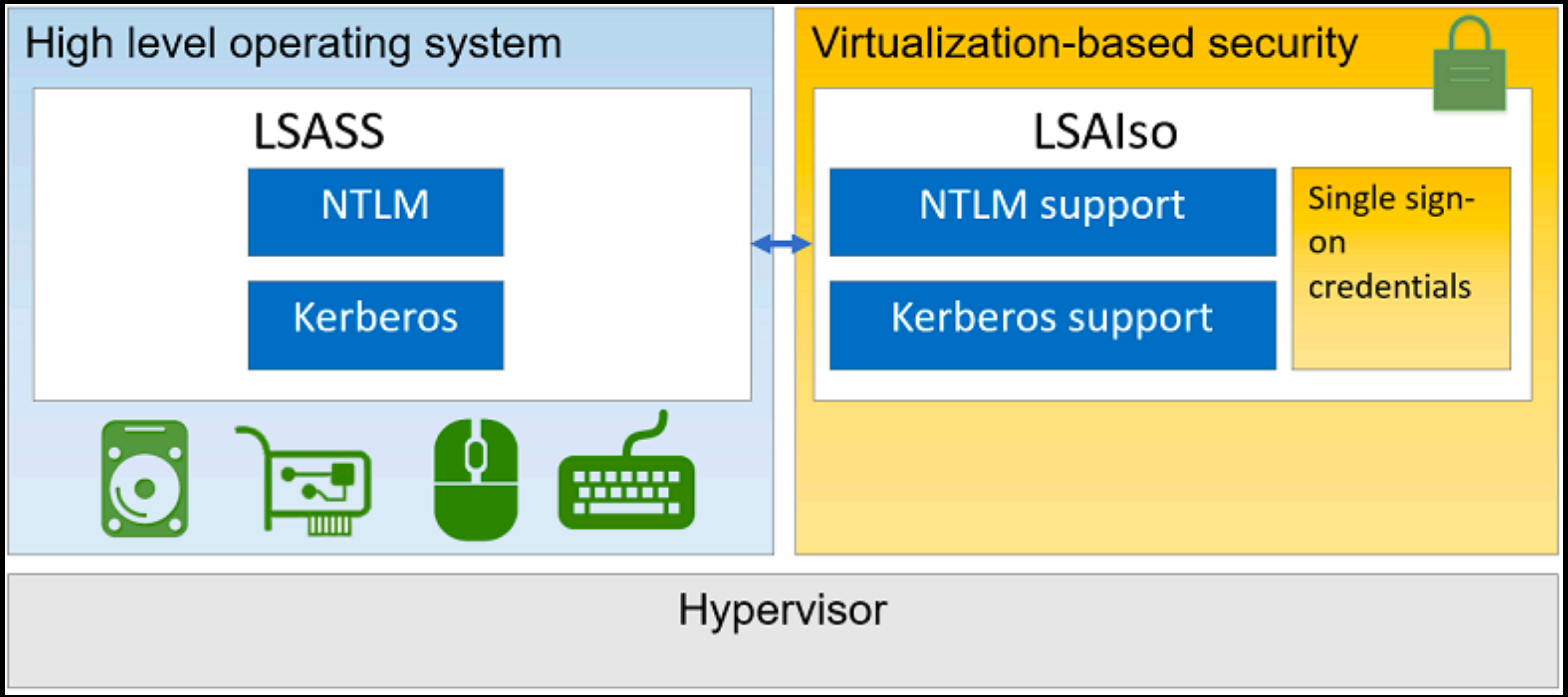
Windows

- 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

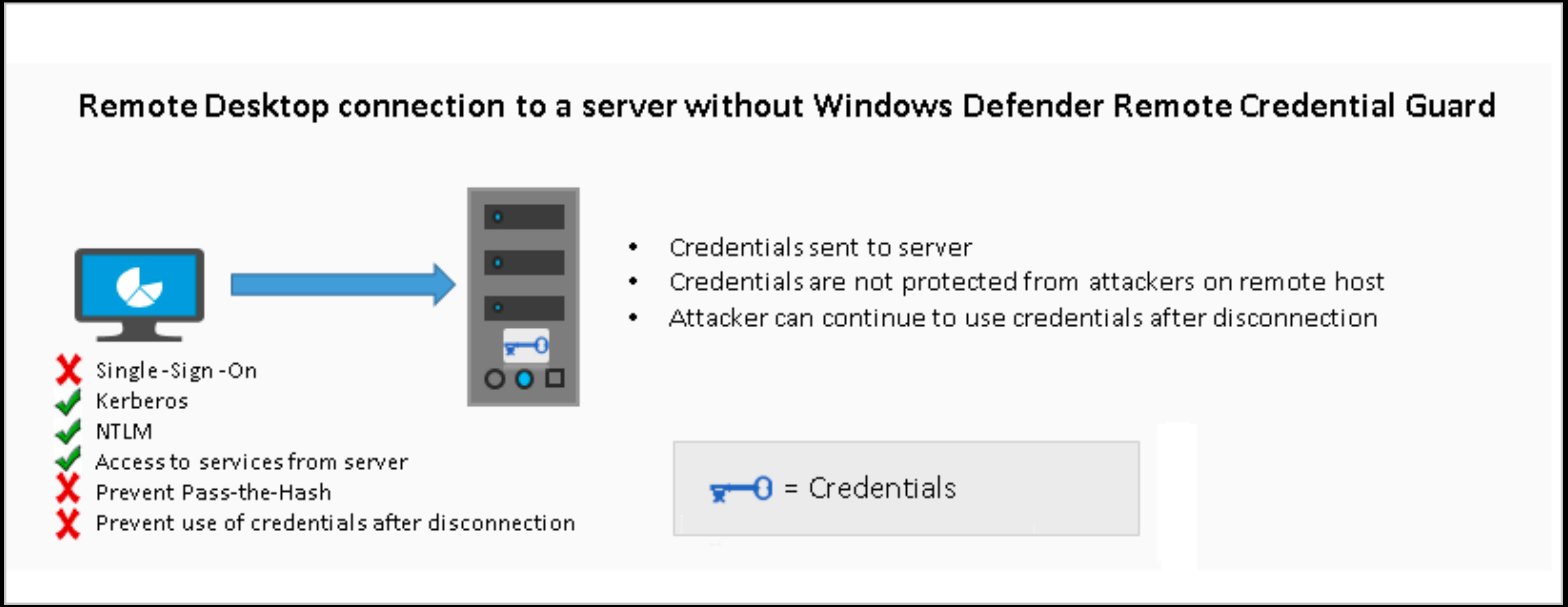
- Windows Credential Guard
  - To protect *Local Security Authority Server Service* (LSASS) by moving it into *LSAIso*
- Build on top of
  - Virtualization Based Security (VBS)
  - Secure boot
  - Trusted Platform Module (TPM)
  - UEFI lock


High level operating system
LSASS
NTLM
Kerberos
Virtualization-based security
LSAIso
NTLM support
Kerberos support
Single sign-on credentials
Hypervisor

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*

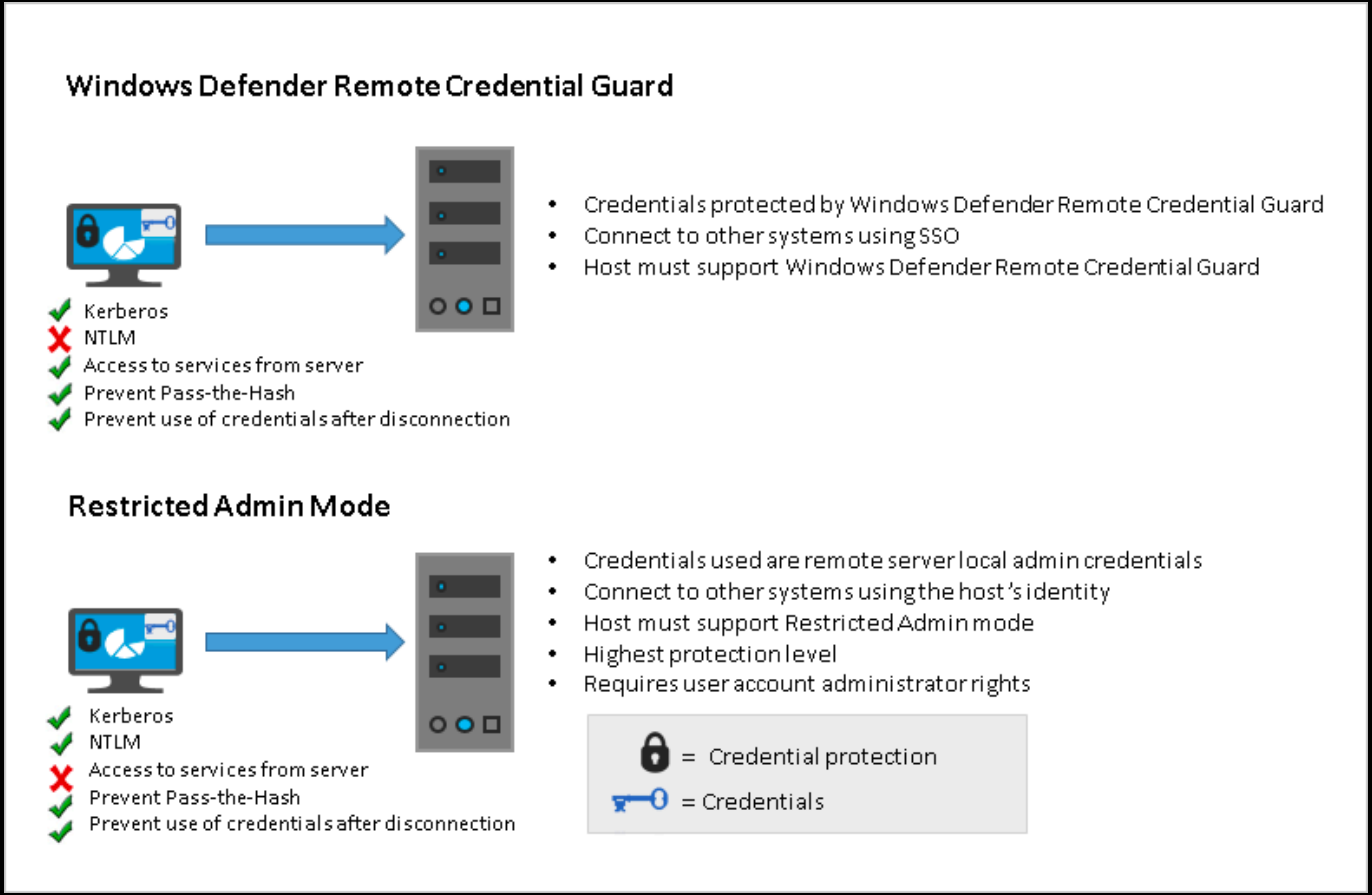

Remote Desktop connection to a server without Windows Defender Remote Credential Guard

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

- Credentials sent to server
- Credentials are not protected from attackers on remote host
- Attacker can continue to use credentials after disconnection

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

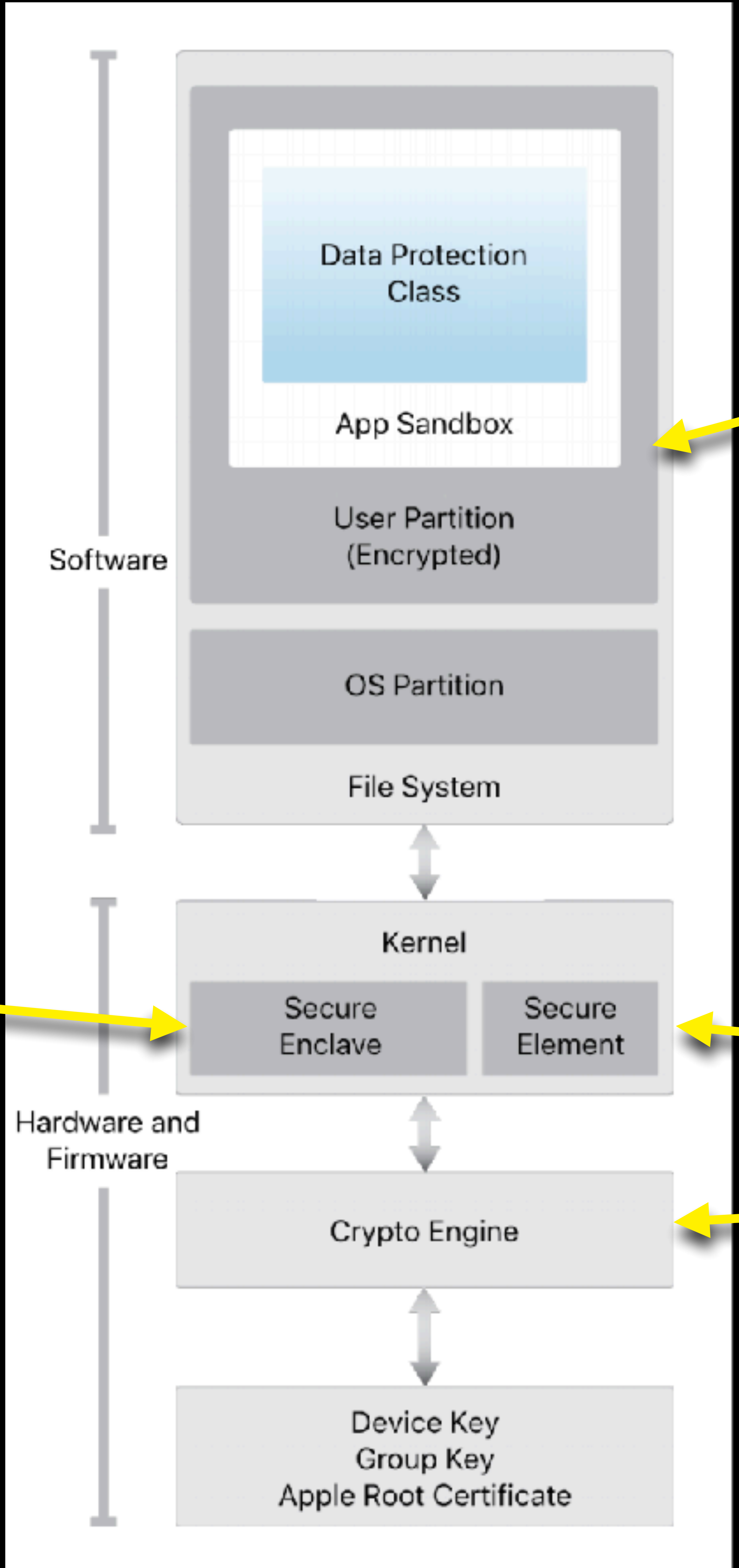
MacOS X

- Hardware based security
  - Secure Enclave
  - Memory Tagging & Pointer Authentication
  - Hardware-Accelerated Encryption
- Isolation
  - App sandbox
- FileVault
  - Full disk encryption
- GateKeeper
  - Checks code signing
- XProtect
  - Malware protection

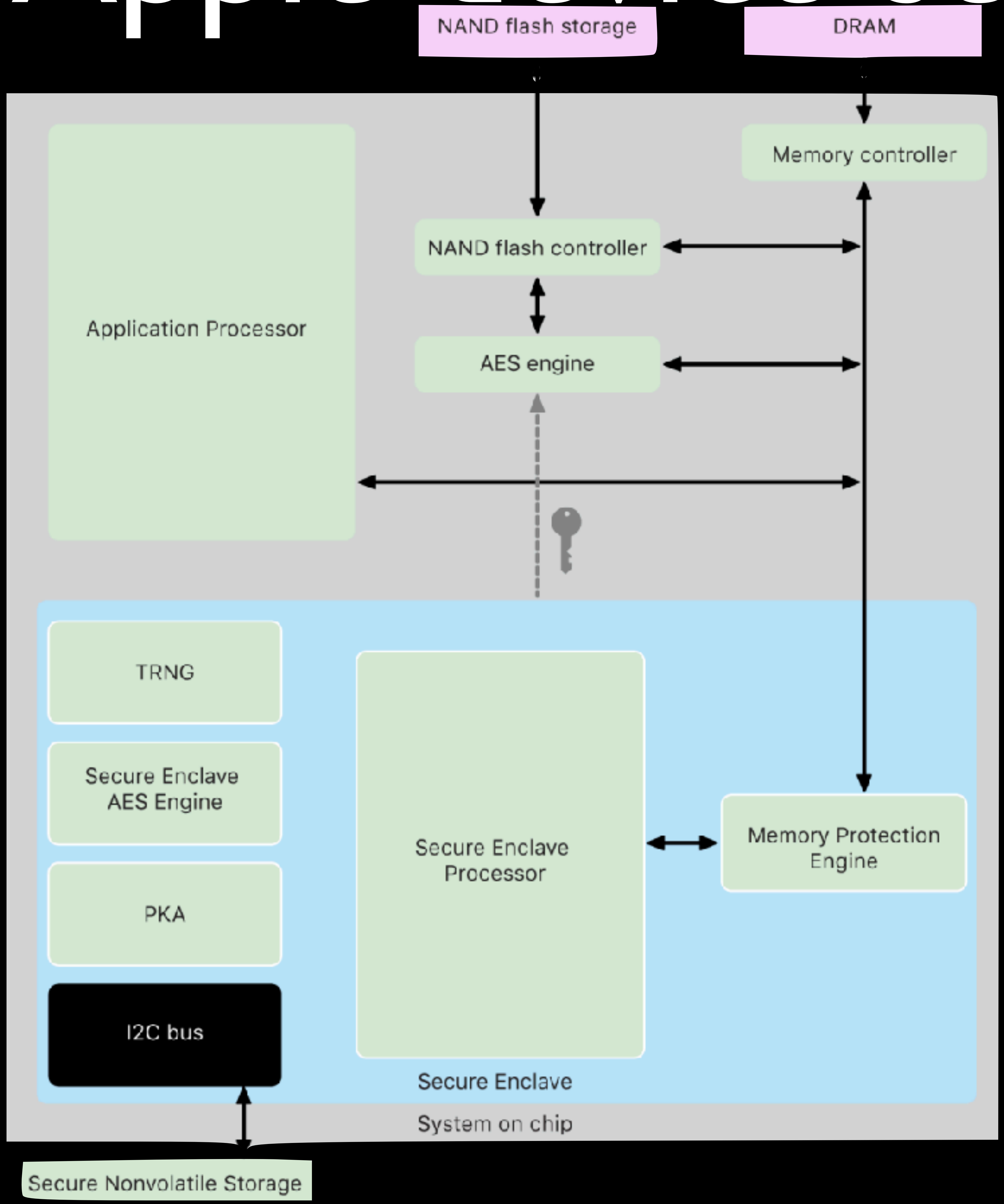
iOS/iPadOS

- Share many security features with MacOS X
- Additional ones include
  - BlastDoor. A way to takes a look at all incoming messages and inspects their content in a secure environment, which prevents any malicious code inside of a message from interacting with iOS or accessing user data.
  - LockDown mode: add many restrictions to applications, e.g. web browsing, messaging, FaceTime, photos, etc

Apple iOS device security



Apple device security



Secure Enclave

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
-

Referenses used during the class

- https://www.cisa.gov/sites/default/files/2024-04/CSRB_Review_of_the_Summer_2023_MEO_Intrusion_Final_508c.pdf
- https://media.ccc.de/v/37c3-12142-breaking_drm_in_polish_trains
- https://www.ccc.de/en/updates/2024/das-ist-vollig-entgleist
- https://attack.mitre.org/
- https://www.cs.cmu.edu/~rdriley/487/papers/Thompson_1984_ReflectionsonTrustingTrust.pdf
-

Referenses used during the class

- https://umatechnology.org/the-truth-about-the-intels-hidden-minix-os-and-security-concerns/
- https://www.bleepingcomputer.com/news/hardware/intels-secret-cpu-on-chip-management-engine-me-runs-on-minix-os/
-