



# Cloud Security and Compliance for cloud services development and assessment

## Additional Materials: Security Foundation

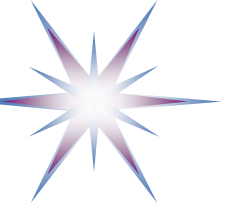
Dr. Yuri Demchenko  
University of Amsterdam

SLICES-SC Summer School  
Open-RAN/Core/Edge Solutions for Cloud-Native Telco Experimental  
Platforms  
July 2022, Volos, Greece



# Additional Information

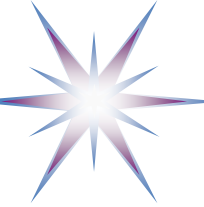
- Addendum 1
  - European Cybersecurity Regulation
  - Security Maturity Model
- Addendum 2
  - Security and Trust zones in cloud usage scenario
  - Java security model, sandboxes
  - Mobile security
- Addendum 3
  - Cybersecurity Maturity Model Certification (CMMC)
  - CWE Top 25 Most Dangerous Software Weaknesses
- Addendum 4
  - Secure Development Lifecycle
  - DevSecOps
- Tutorial and practice materials  
[https://drive.google.com/drive/folders/1nCqA0n51bS\\_98ACUudrAeqM468aZI7hq?usp=sharing](https://drive.google.com/drive/folders/1nCqA0n51bS_98ACUudrAeqM468aZI7hq?usp=sharing)



# Addendum 1

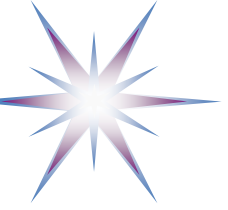
## European Cybersecurity Regulation

- EU Cybersecurity Act (2019) - <https://eur-lex.europa.eu/eli/reg/2019/881/oj>
  - General principles, definitions
  - ENISA functions formalisation
    - Market and technology research, strategic analysis
    - Innovation, awareness rising, education
  - European Cybersecurity Certification Group (ECCG) and National bodies
  - European Cybersecurity certification requirements and elements
- EU Cybersecurity Certification (CC) Framework, by ENISA (2021)  
<https://www.enisa.europa.eu/publications/cybersecurity-certification-eucc-candidate-scheme-v1-1.1>
  - General principles, requirements
  - Reference to Intl standards: ISO 27000
  - Examples for Integrated Circuits, Smart Cards and similar devices



# EU Cybersecurity Act (2019) – Principles and Requirements

- **Goals:**
  - Increase trust to ICT, setup common certification process in EU
  - Provide basis for cybersecurity and trustworthiness by design
- **Scope:** All elements and layers of the cyber infrastructure and data infrastructure
- **3 Cybersecurity conformance levels**
  - Basic (art. 88)
  - Substantial (art. 89)
  - High (art. 90)
- **Conformity certification and self-assessment**
  - Technical requirements to European Cybersecurity Certification Scheme (ECCS)
  - Reference to EU Regulation No 1025/2012 (Annex II – requirements to technical specification) - also known as the Standardisation Regulation
  - Conformance to ECCS is voluntary while Conformity Statement to help users to make informed choices (art. 93)



# EU Cybersecurity Act: 3 Cybersecurity conformance levels

- **Basic (art. 88)**
  - For assurance level ‘basic’, the evaluation should be guided at least by the following assurance components: the evaluation should at least include a review of the technical documentation of the ICT product, ICT service or ICT process by the conformity assessment body. Where the certification includes ICT processes, the process used to design, develop and maintain an ICT product or ICT service should also be subject to the technical review. Where a European cybersecurity certification scheme provides for a conformity self-assessment, it should be sufficient that the manufacturer or provider of ICT products, ICT services or ICT processes has carried out a self-assessment of the compliance of the ICT product, ICT service or ICT process with the certification scheme.
  - **Self-assessment corresponds to Basic level**
- **Substantial (art. 89)**
  - For assurance level ‘substantial’, the evaluation, in addition to the requirements for assurance level ‘basic’, should be guided at least by the verification of the compliance of the security functionalities of the ICT product, ICT service or ICT process with its technical documentation.
- **High (art. 90)**
  - For assurance level ‘high’, the evaluation, in addition to the requirements for assurance level ‘substantial’, should be guided at least by an efficiency testing which assesses the resistance of the security functionalities of ICT product, ICT service or ICT process against elaborate cyberattacks performed by persons who have significant skills and resources.

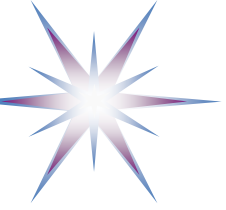


# EU Cybersecurity Certification Framework

- Implements Cybersecurity Act (CSA) by articles
- Evaluations are based on the standards
  - Common Criteria for Information Technology Security Evaluation, under their applicable ISO/IEC 15408 (<https://www.commoncriteriaportal.org/cc/>)
  - Common Methodology for Information Technology Security Evaluation, under its applicable ISO/IEC 18045 version
  - ISO/IEC 17065, for the conformity assessment body or national authority in charge of the activities of certification, hereinafter designated as certification body (CB)
  - ISO/IEC 17025, for the part of a third-party conformity assessment body or national authority
- The EUCC scheme covers assurance levels ‘substantial’ and ‘high’ of the CSA (level ‘basic’ achieved with self-assessment)
- Rules for monitoring compliance and non-compliance: Renewal period and termination
- Period of validity and Peer assessment
- Rules for handling vulnerabilities
  - Previously undetected vulnerability shall be reported and handled in accordance with the general rules of ISO/IEC 30111 and ISO/IEC 29147, adapted for this scheme, with the additional possibility of patch management

## Acronyms

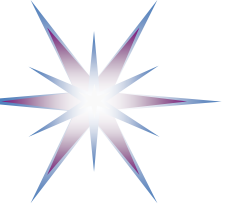
CB – Certification Body  
CSA - Cybersecurity Act  
TOE - Target of Evaluation  
CEM – Common Evaluation Methodology  
SAR - Security Assurance Requirement  
SFR - Security Functional Requirement  
ITSEF – Testing laboratory/Evaluation Facility



# References

- **CSA (Cybersecurity Act) REGULATION (EU) 2019/881 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 17 April 2019 on ENISA (the European Union Agency for Cybersecurity) and on information and communications technology cybersecurity certification and repealing Regulation (EU) No 526/2013.**
- **SOG-IS MRA Mutual Recognition Agreement of Information Technology Security Evaluation Certificates VERSION 3.0, MANAGEMENT COMMITTEE, January 2010.**
- **CCRA ARRANGEMENT on the Recognition of Common Criteria Certificates In the field of Information Technology Security, July 2, 2014.**

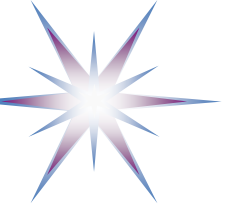
Reference	Title
ISO/IEC 15408	Information technology - Security techniques - Evaluation criteria for IT security
ISO/IEC 18045	Information technology - Security techniques - Methodology for IT security evaluation
ISO/IEC 17000	Conformity assessment - Vocabulary and general principles
ISO/IEC 17065	Conformity assessment - Requirements for bodies certifying products, processes and services
ISO/IEC 17025	Testing and calibration laboratories
ISO/IEC 19896-3	IT security techniques — Competence requirements for information security testers and evaluators — Part 3: Knowledge, skills and effectiveness requirements for ISO/IEC 15408 evaluators
ISO/IEC WD TS 23532-1	IT Security Techniques — Requirements for the competence of IT security testing and evaluation laboratories — Part 1: Testing and evaluation for ISO/IEC 15408
ISO/IEC 27001	Information technology - Security techniques - Information security management systems – Requirements
ISO/IEC 27002	Information technology - Security techniques - Code of practice for information security management controls
ISO/IEC 27005	Information technology - Security techniques - Information security risk management
ISO/IEC 29147	Information technology - Security techniques - Vulnerability disclosure
ISO/IEC 30111	Information technology - Security techniques - Vulnerability handling processes
ISO/IEC 7816-4	Identification cards - Integrated circuit cards - Part 4: Organization, security and commands for interchange



## ANNEX 2: MINIMUM SITE SECURITY REQUIREMENTS (pp 79-120)

- Specifies site security requirement according CC based on ISO27001 and Common Criteria
- Example: For typical products related to the Technical Domain hardware devices with security boxes, the following segments of the life cycle and forms of the TOE (Target of Evaluation) apply.
  - Design and Development Phase
  - Manufacturing Phase
  - Preparation Phase
  - Component Supply\*
  - Assembly\*
  - Initialization
  - Security Data Generation and Insertion\*
  - Storage distribution\*
  - Repair\*
  - Installation and Calibration Phase\*
  - Inspection and Calibration\*
  - Activation, Pairing or Coupling\*
  - Operational Phase
  - Delivery Process preparation (Shipments)
  - End of Life-Handling
- Important assets beside the TOE or parts of it are typically:
  - Security: access control and alarm system, keys, access codes.
  - Relevant information for the knowledge of the TOE: specifications, design documentation, guidance, source code, IC and embedded software representation, penetration tests results.
  - Sensitive data used during the development phase of the TOE: keys, passwords, memory profile, integrity evidence.
  - Information: databases, data files, contracts, system documentation, R&D information, archived information, production related data.
  - Software: R&D tools, applications, system software, development tools, CM systems.
  - Physical assets: computer equipment, communication equipment, removable media.
  - Services: computing and communications services, general utilities (power, air conditioning, lighting), storage and shipment





# Example of using Cybersecurity Certification in Appendices

- ANNEX 3: APPLICATION OF CC TO INTEGRATED CIRCUITS
- ANNEX 4: SECURITY ARCHITECTURE REQUIREMENTS (ADV\_ARC) FOR SMART CARDS AND SIMILAR DEVICES
- ANNEX 5: CERTIFICATION OF "OPEN" SMART CARD PRODUCTS
- ANNEX 6: COMPOSITE PRODUCT EVALUATION FOR SMART CARDS AND SIMILAR DEVICES
- ANNEX 7: APPLICATION OF ATTACK POTENTIAL TO SMARTCARDS AND SIMILAR DEVICES
- ANNEX 8: MINIMUM ITSEF REQUIREMENTS FOR SECURITY EVALUATIONS OF SMART CARDS AND SIMILAR DEVICES
- ANNEX 9: APPLICATION OF ATTACK POTENTIAL TO HARDWARE DEVICES WITH SECURITY BOXES
- ANNEX 10: MINIMUM ITSEF REQUIREMENTS FOR SECURITY EVALUATIONS OF HARDWARE DEVICES WITH SECURITY BOXES

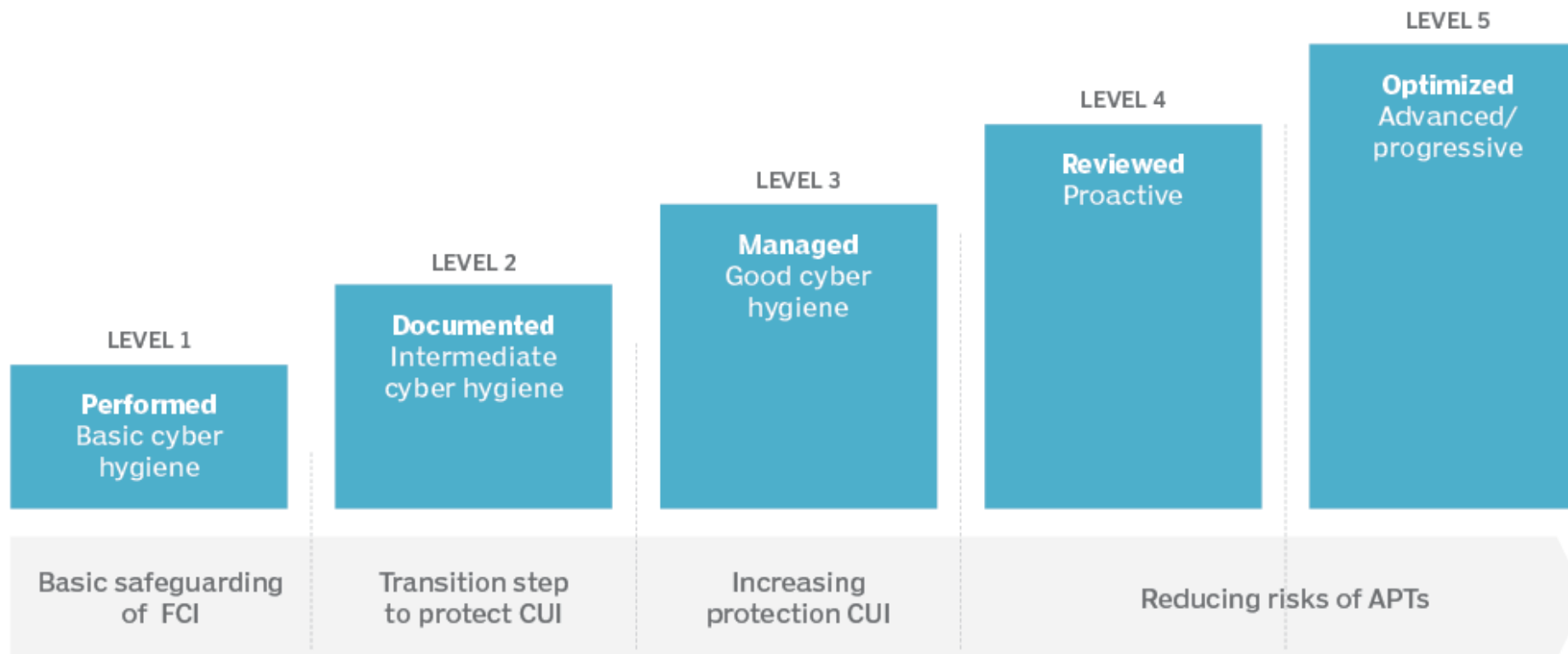


# Addendum 1

## Cybersecurity Maturity Model Certification (CMMC) framework

- Joint Carnegie Mellon University and Johns Hopkins University Applied Physics Laboratory LLC called the Cybersecurity Maturity Model Certification (CMMC) framework
  - <https://searchcompliance.techtarget.com/tip/The-5-CMMC-levels-and-how-to-achieve-compliance>
  - [https://www.acq.osd.mil/cmmc/docs/CMMC\\_Model\\_Main\\_20200203.pdf](https://www.acq.osd.mil/cmmc/docs/CMMC_Model_Main_20200203.pdf)
- Originally designed for the U.S. Department of Defense, the CMMC framework addresses the needs of the DOD for protecting classified uncontrolled information during the procurement of products and services from the defense industrial base.
- Third-party contract assessors certified by the DOD to audit CMMC compliance are responsible for conducting certifications.

# The five levels of DoD's Cybersecurity Maturity Model Certification



# CMMC processes, by level

MATURITY LEVEL	MATURITY LEVEL DESCRIPTION	PROCESSES
1	Performed	No maturity processes assessed at Level 1
2	Documented	Establish a policy that includes a domain name, document CMMC practices that implement domain policy
3	Managed	Establish, maintain and resource a plan that addresses the selection domain
4	Reviewed	Review and measure the domain activities for effectiveness
5	Optimized	Standardize and optimize a documented approach for the domain across all applicable organization units

SOURCE: U.S. DOD'S CYBERSECURITY MATURITY MODEL FRAMEWORK

©2020 TECHTARGET. ALL RIGHTS RESERVED 

# CMMC domains

DESIGNATION	DOMAIN NAME
AC	Access control
AM	Asset management
AU	Audit and accountability
AT	Awareness and training
CM	Configuration management
IA	Identification and authentication
IR	Incident response
MA	Maintenance
MP	Media protection
PS	Personnel security
PE	Physical protection
RE	Recovery
RM	Risk management
CA	Security assessment
SA	Situational awareness
SC	System and communications protection
SI	System and information integrity

SOURCE: U.S. DOD'S CYBERSECURITY MATURITY MODEL FRAMEWORK  
©2020 TECHTARGET. ALL RIGHTS RESERVED

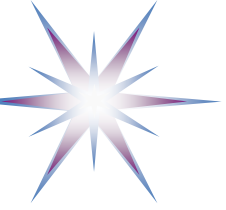
- CMMC has 17 domains that were previously defined in U.S. [Federal Information Processing Standards Publication 200](#) and [NIST Special Publication 800-171](#), Protecting Controlled Unclassified Information in Nonfederal Systems and Organizations.



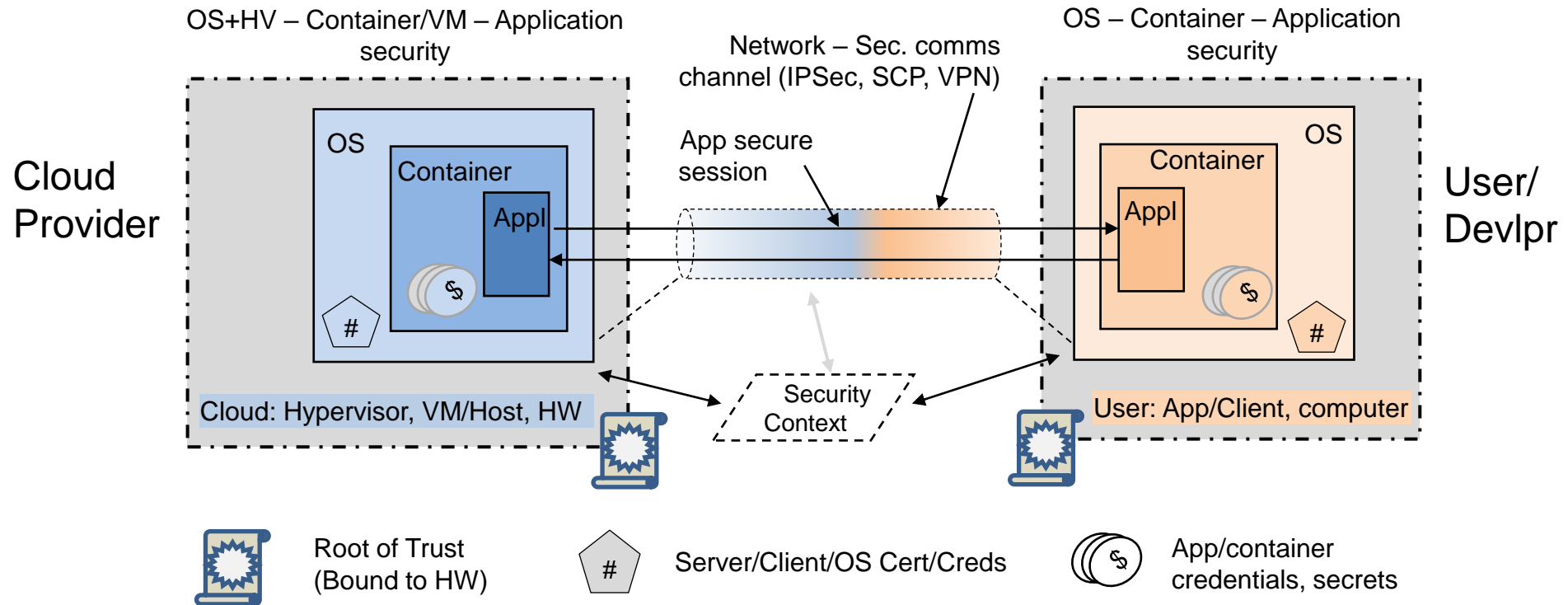
# Addendum 2

## Services Security and Trust Zones Model

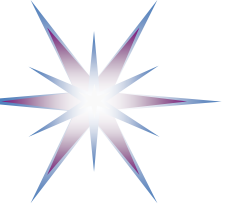
---



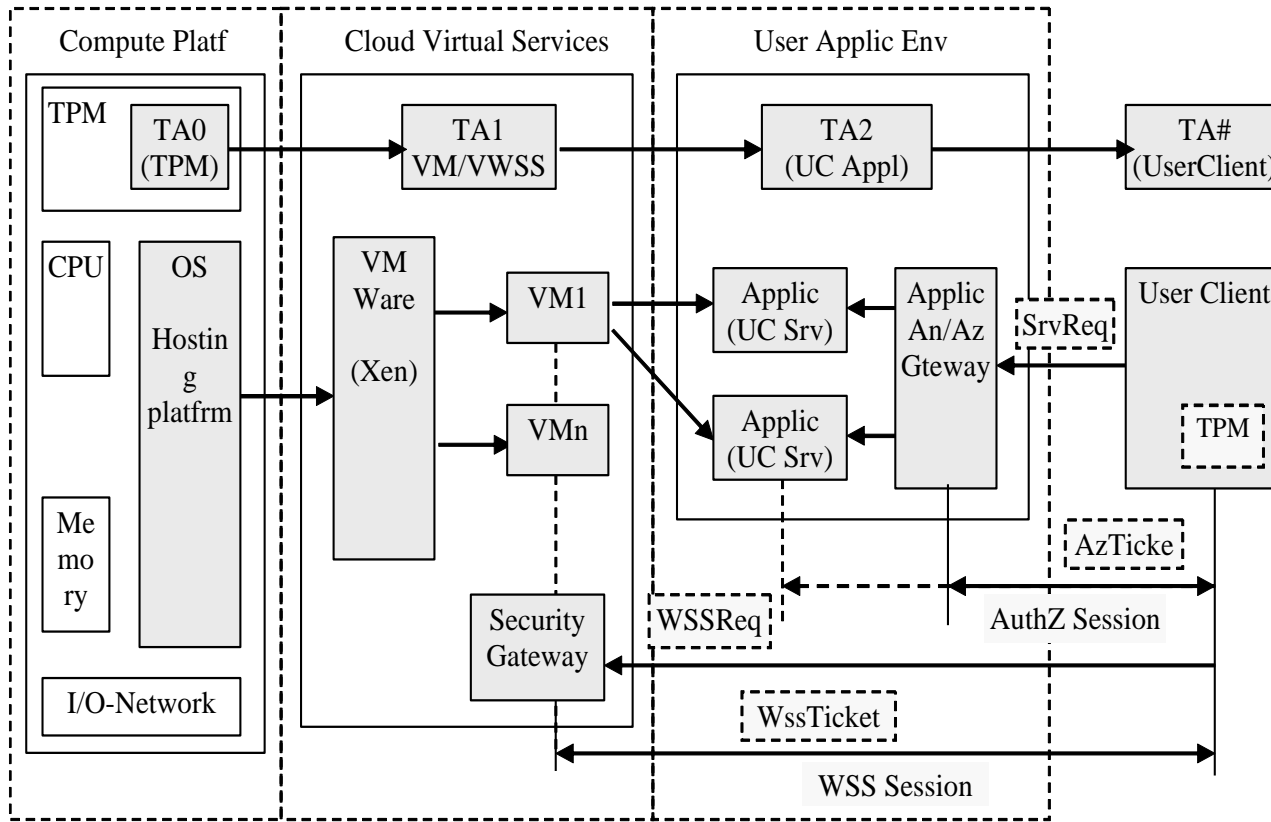
# Cloud, OS, Network and Applications Trust Layers



- Consistent security must provide security at all layers correspondingly relying on trust credentials at each layer
  - Application – Container - Operating systems (security kernel) + Cloud platform
  - Network/communication – Runtime - Storage
- Two security models: Trusted Computing Base (TCB) for cloud platform and OSI/Internet security cloud based applications
  - Client/server and Service Oriented Architecture vs OS and hypervisor run-time
- Root of trust is based on the security credentials bound to hardware mediated through OS to runtime environment

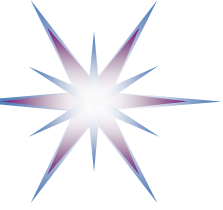


# TCPA based Trusted Collaborative Environment

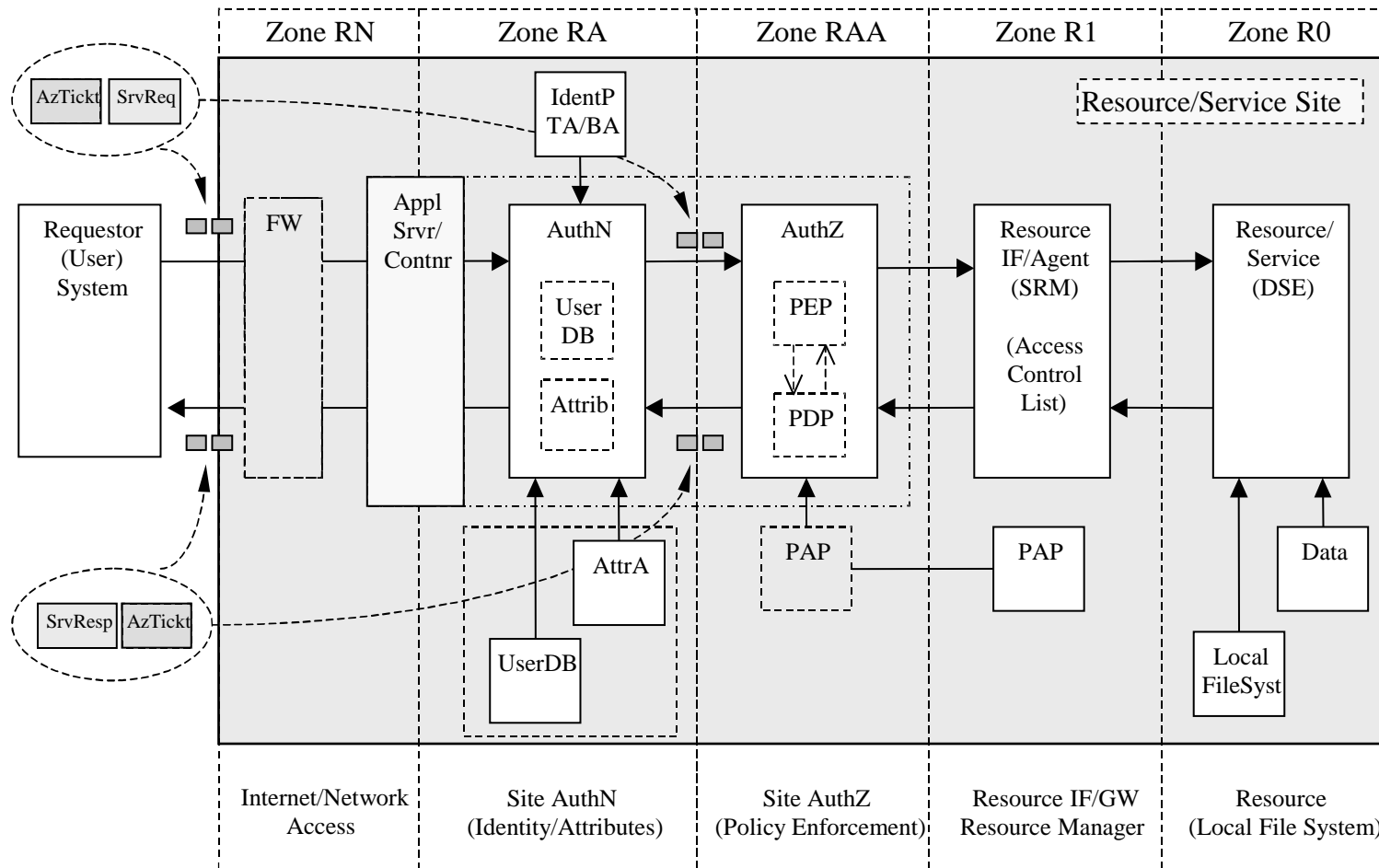


- 3 layer User Controlled VWSS (VWSS-UC)
- Trusted Computing Platform
- Virtual Workspace Service (VWSS)
- Application/Resource (dynamic) access control service



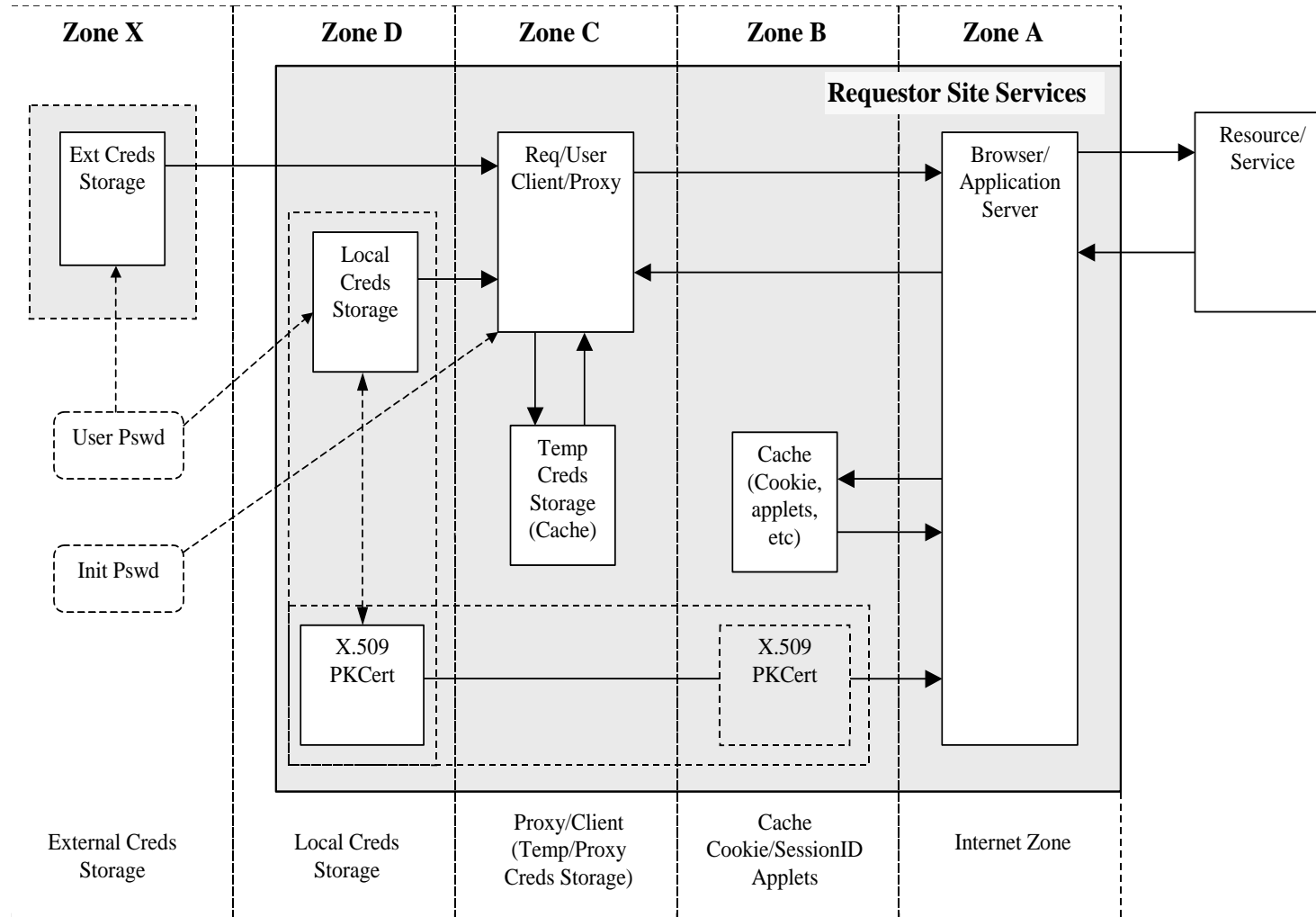


# Resource Zone Security model for Cloud/Web Services





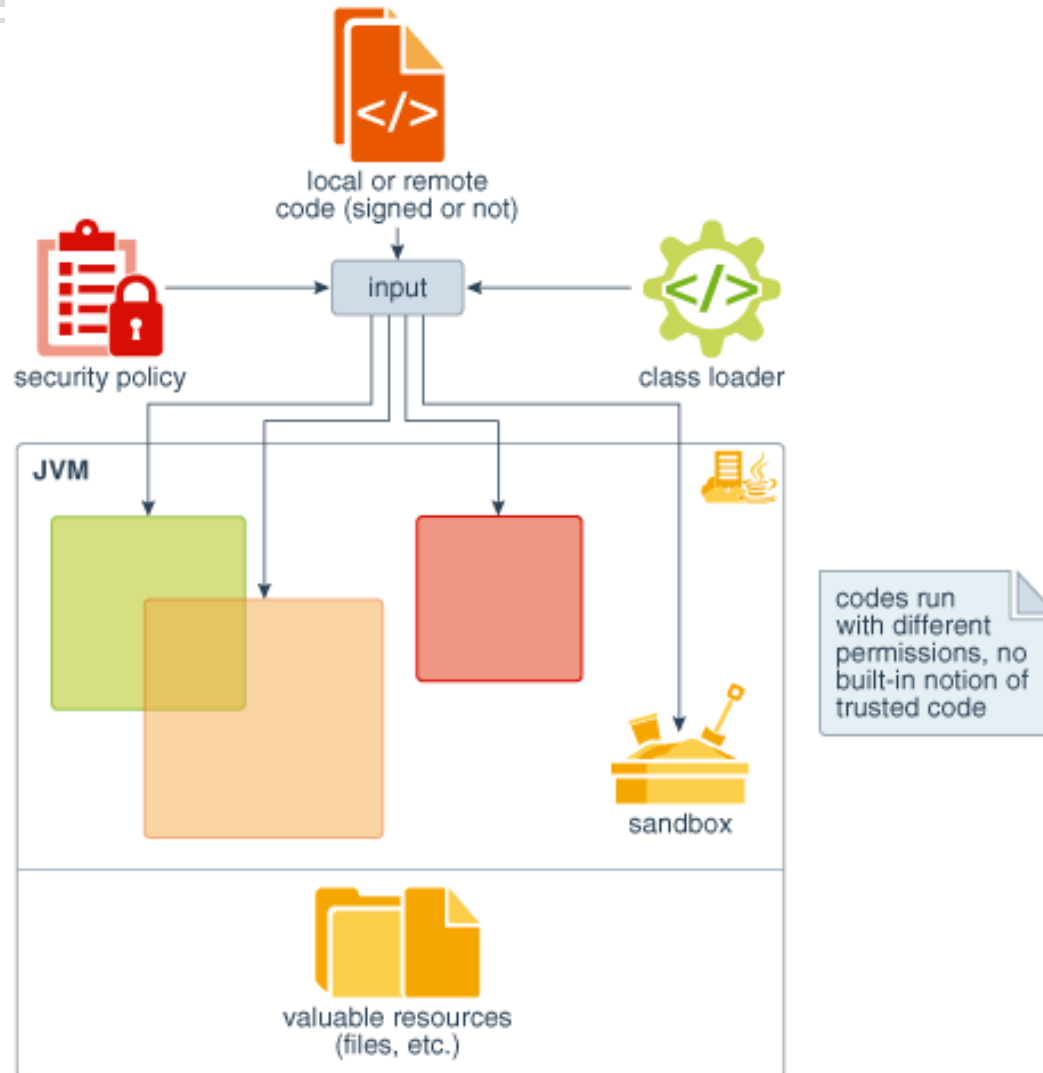
# Requestor Zone Security model for Cloud Services





# Addendum 2

## Java and Browser Security Model



- Browser runs as Java applet in the sandbox (as untrusted code)
- Sandbox applications don't access to local system resources
- Strict Java security policy

<https://docs.oracle.com/en/java/javase/13/security/java-se-platform-security-architecture.html>



# Addendum 2

## Mobile devices and smartphone security

- Mobile devices security
- Baseband processor architecture



# Mobile devices security

- See overview and comparison of the security of mobile platforms: IOS, Android, Windows  
<http://meseec.ce.rit.edu/551-projects/fall2015/3-2.pdf>  
<https://crypto.stanford.edu/cs155old/cs155-spring15/lectures/17-mobile-platforms.pdf>
- Every smartphone or other device with mobile communications capability (e.g. 3G or LTE) has **two processors and runs two operating systems by design**
  - Application Processor/OS (Android, iOS, Windows)
  - Broadband Processor and **proprietary** RTOS that manages everything related to radio, e.g. Qualcomm's Infineon and chip  
[http://www.osnews.com/story/27416/The\\_second\\_operating\\_system\\_hiding\\_in\\_every\\_mobile\\_phone](http://www.osnews.com/story/27416/The_second_operating_system_hiding_in_every_mobile_phone)
    - Implements std protocols GSM, UMTS, HSDPA, etc
    - Runs Hayes commands for controlling modem function
    - Existing bug allows multiple attacks <https://www.infoworld.com/article/2625180/smartphones/coming-soon--a-new-way-to-hack-into-smartphones.html>



# Baseband processor architecture

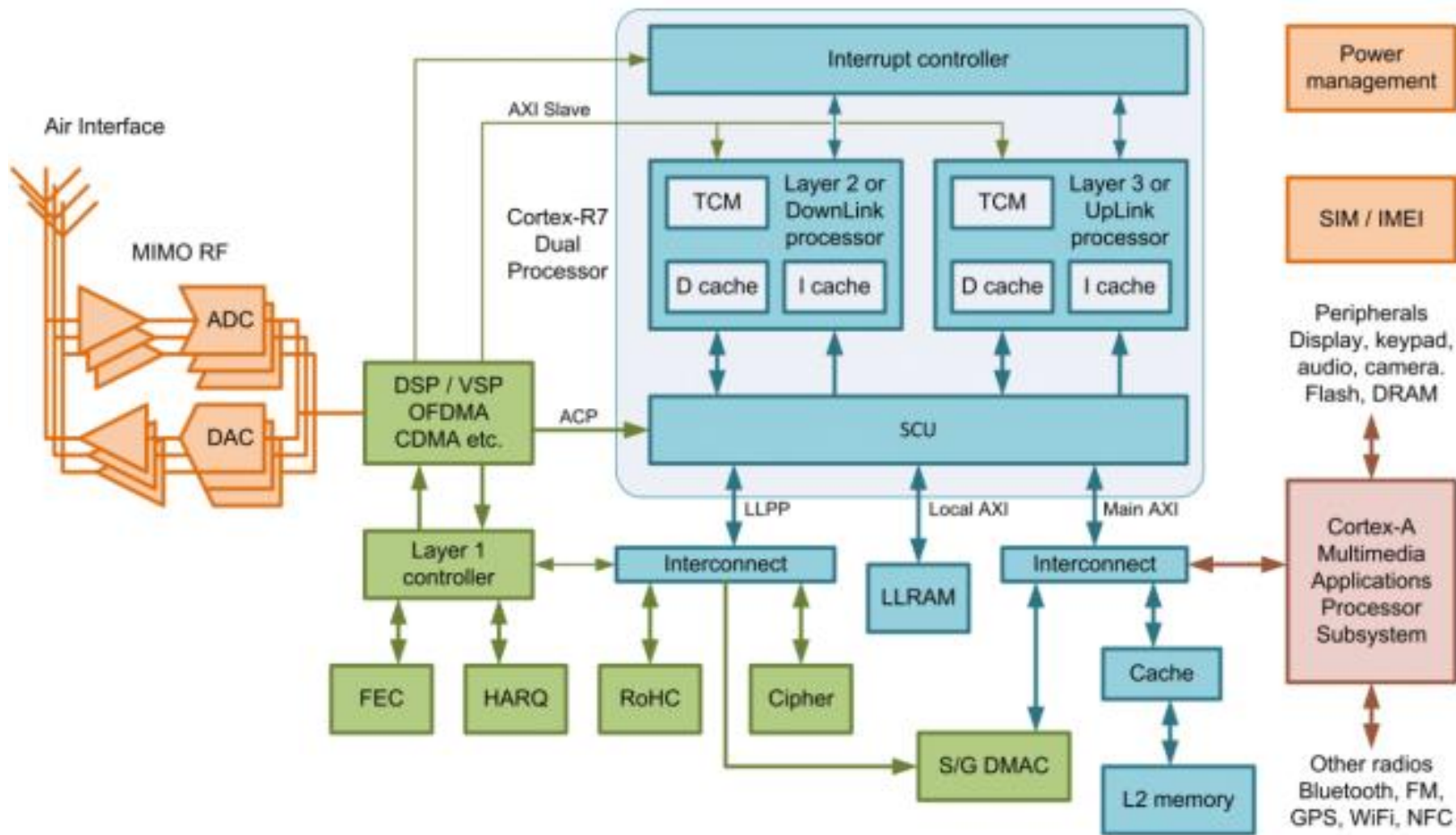
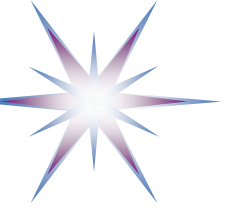


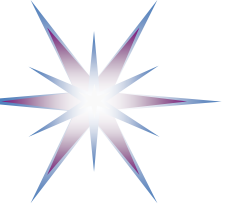
Figure 3: Illustrative baseband architecture

- Baseband processor is the digital system for transmitting and receiving data over the radio. Baseband processor is divided in two parts - <https://www.androidauthority.com/smart-phones-have-a-second-os-317800/>
  - Modem to modulate and demodulate the radio signal
  - Protocol stack processor which manages the communication between base station and mobile terminal by establishing connections, managing radio resources, handling errors and packetizing incoming and outgoing data
- Patent <https://encrypted.google.com/patents/U S9191823>



# Addendum 3 – Software Vulnerabilities and Exposures

- CWE Top 25 Most Dangerous Software Weaknesses
- OWASP Top Ten



# The most often occurring Attacks

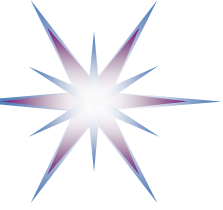
## Network level attacks

- SYNC flooding
- IP address and domain name spoofing
- DNS attacks
  - DNS cache poisoning (Kaminski attack)
  - DNS Pharming

## Application or user involved attacks

- Authentication: weak password
- MITM – Man In The Middle
- Replay attacks
- Social engineering
  
- CVE (Common Vulnerabilities and Exposure)
  - <https://cve.mitre.org/index.html>
- CWE Top 25 Most Dangerous Software Weaknesses
- OWASP (Open Web Application Security Project)
  - OWASP Top Application Security Risks - <https://owasp.org/www-project-top-ten/>





# 2022 CWE Top 25 Most Dangerous Software Weaknesses

[https://cwe.mitre.org/top25/archive/2022/2022\\_cwe\\_top25.html](https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25.html)

Rank	ID	Name	Score	KEV Count (CVEs)	Rank Change vs. 2021
4	<a href="#">CWE-20</a>	Improper Input Validation	20.63	20	0
5	<a href="#">CWE-125</a>	Out-of-bounds Read	17.67	1	-2 ▼
6	<a href="#">CWE-78</a>	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	17.53	32	-1 ▼
7	<a href="#">CWE-416</a>	Use After Free	15.50	28	0
8	<a href="#">CWE-22</a>	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.08	19	0
9	<a href="#">CWE-352</a>	Cross-Site Request Forgery (CSRF)	11.53	1	0
10	<a href="#">CWE-434</a>	Unrestricted Upload of File with Dangerous Type	9.56	6	0
11	<a href="#">CWE-476</a>	NULL Pointer Dereference	7.15	0	+4 ▲
12	<a href="#">CWE-502</a>	Deserialization of Untrusted Data	6.68	7	+1 ▲
13	<a href="#">CWE-190</a>	Integer Overflow or Wraparound	6.53	2	-1 ▼
14	<a href="#">CWE-287</a>	Improper Authentication	6.35	4	0
15	<a href="#">CWE-798</a>	Use of Hard-coded Credentials	5.66	0	+1 ▲
16	<a href="#">CWE-862</a>	Missing Authorization	5.53	1	+2 ▲
17	<a href="#">CWE-77</a>	Improper Neutralization of Special Elements used in a Command ('Command Injection')	5.42	5	+8 ▲
18	<a href="#">CWE-306</a>	Missing Authentication for Critical Function	5.15	6	-7 ▼
19	<a href="#">CWE-119</a>	Improper Restriction of Operations within the Bounds of a Memory Buffer	4.85	6	-2 ▼
20	<a href="#">CWE-276</a>	Incorrect Default Permissions	4.84	0	-1 ▼
21	<a href="#">CWE-918</a>	Server-Side Request Forgery (SSRF)	4.27	8	+3 ▲
22	<a href="#">CWE-362</a>	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	3.57	6	+11 ▲
23	<a href="#">CWE-400</a>	Uncontrolled Resource Consumption	3.56	2	+4 ▲
24	<a href="#">CWE-611</a>	Improper Restriction of XML External Entity Reference	3.38	0	-1 ▼
25	<a href="#">CWE-94</a>	Improper Control of Generation of Code ('Code Injection')	3.32	4	+3 ▲

- Common Weakness Enumeration (CWE™) is a community-developed list of common software and hardware weakness types that have security ramifications.
- CWE helps developers and security practitioners to:
  - Describe and discuss **software and hardware weaknesses** in a common language.
  - **Check for weaknesses** in existing software and hardware products.
  - Evaluate coverage of tools targeting these weaknesses.
  - Leverage a common baseline **standard for weakness identification, mitigation, and prevention efforts.**
  - **Prevent software and hardware vulnerabilities prior to deployment.**



# CWE mapping and additional views

<https://cwe.mitre.org/data/index.html>

## CWE mapping to other frameworks

- CWE Top 25 (2022)
- Most Important Hardware Weaknesses List (2021)
- OWASP Top Ten (2021)
- Seven Pernicious Kingdoms
- Software Fault Pattern Clusters
- SEI CERT Oracle Coding Standard for Java
- SEI CERT C Coding Standard
- SEI CERT Perl Coding Standard
- CISQ Quality Measures (2020)
- CISQ Data Protection Measures
- SEI ETF Security Vulnerabilities in ICS
- Architectural Concepts

Additional helpful views are based on a specific criteria and hope to provide insight for a certain domain or use case.

- Introduced During Design
- Introduced During Implementation
- Quality Weaknesses with Indirect Security Impacts
- Software Written in C
- Software Written in C++
- Software Written in Java
- Software Written in PHP
- Weaknesses in Mobile Applications
- CWE Composites
- CWE Named Chains
- CWE Cross-Section
- CWE Simplified Mapping
- CWE Entries with Maintenance Notes
- CWE Deprecated Entries
- CWE Comprehensive View
- Weaknesses without Software Fault Patterns
- Weakness Base Elements



# OWASP Top Application Security Risks - 2017

## **A1:2017-Injection**

Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

## **A2:2017-Broken Authentication**

Application functions related to authentication and session management are often implemented incorrectly.

## **A3:2017-Sensitive Data Exposure**

Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII.

## **A4:2017-XML External Entities (XXE)**

Many older or poorly configured XML processors evaluate external entity references within XML documents.

## **A5:2017-Broken Access Control**

Restrictions on what authenticated users are allowed to do are often not properly enforced.

## **A6:2017-Security Misconfiguration**

Security misconfiguration is the most commonly seen issue.

## **A7:2017-Cross-Site Scripting (XSS)**

XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

## **A8:2017-Insecure Deserialization**

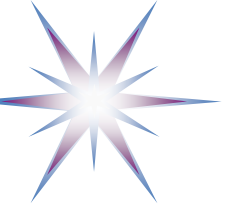
Insecure deserialization often leads to remote code execution.

## **A9:2017-Using Components with Known Vulnerabilities**

Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application.

## **A10:2017-Insufficient Logging&Monitoring**

Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems. Most breach studies show time to detect a breach is over 200 days.



# JavaScript Vulnerabilities and examples

- Security Concerns With Javascript Development – Secure Practices And Tips - <https://www.algoworks.com/blog/security-concerns-with-javascript-development/>
  - 1 - Cross-Site Scripting (XSS)
  - 2 - Cross-Site Request Forgery (CSRF)

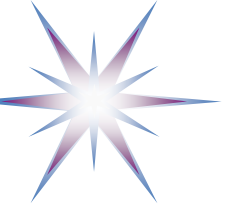
```
<img src= "<a class="vglnk" href="http://myweb/app/sendFunds" rel="nofollow">
<span>http</span><span>://</span><span>myweb</span><span>/</span><span>app</span>
<span>/</span> <span>sendFunds</span></a>
?amount=9880&targetAct=hackerAcct#" />
```
  - 3 - Client-side Logic And Data Storage
  - 4 - Server-side JavaScript Injection (SSJI)
- JavaScript security best practices - <https://wpvip.com/documentation/vip-go/vip-code-review/javascript-security-best-practices/>
  - Cross-Site Scripting
  - Escaping Dynamic JavaScript Values
  - Stripping Tags
  - Using encodeURIComponent()
  - Don't use eval() e.g. in password checking
- OWASP: [https://cheatsheetseries.owasp.org/cheatsheets/DOM\\_based\\_XSS\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.html)
- Google Apps Security: <https://www.google.com/about/appsecurity/learning/xss/>



# Same-Origin Policy - JavaScript Security Model

- *“XMLHttpRequest is subject to the browser's same origin policy in that, for security reasons, requests will only succeed if they are made to the same server that served the original web page. There are alternative ways to circumvent this policy if required.”* (Source: Wikipedia)

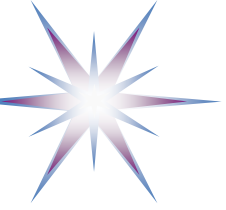
Source URL: http://www.example.com/dir/page.html		
Compared URL	Outcome	Reason
http://www.example.com/dir/page.html	Success	Same protocol and host
http://www.example.com/dir2/other.html	Success	Same protocol and host
http://www.example.com: <u>81</u> /dir/other.html	Failure	Same protocol and host but different port
https://www.example.com/dir/other.html	Failure	Different protocol
http:// <u>en</u> .example.com/dir/other.html	Failure	Different host
http:// <u>example.com</u> /dir/other.html	Failure	Different host (exact match required)
http:// <u>v2</u> .www.example.com/dir/other.html	Failure	Different host (exact match required)



## Addendum 4

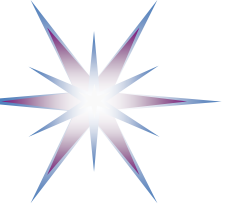
# Design approach and measures to ensure security in Cloud

- DevSecOps
- Cloud Security Configuration Monitoring – AWS Tools



# DevSecOps and Automated Security Testing

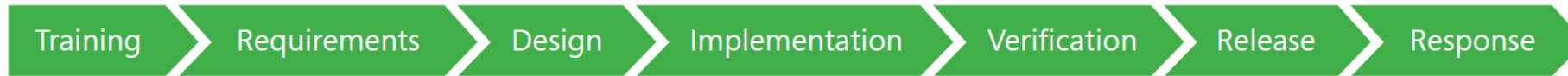
- Waterfall roots:
  - Often done at the end of product development and in short timeline
  - Security ends up as blocking release
- The rise of compliance: Passing Audit = Security
  - Risk management and security
- DevOps meets Security -> DevSecOps
  - Change the culture from Waterfall
  - Security developer to embrace the role of Enabler: How can we help?
    - Seek to Added Value
  - Secure the Software Supply Chain
    - **Recent SolarWinds attack and supply chain hack**
      - Around 18,000 customers installed affected update into their systems
      - <https://www.solarwinds.com/sa-overview/securityadvisory>



# DevSecOps and SSDL

- SSDL – Security Services Development Lifecycle
  - Developed by Microsoft in 2000s and widely accepted by industry

SSDL = Security and Privacy by Design



- Security design principles by big software vendors Amazon, Apple, Google
- DevOps meets Security -> DevSecOps
- DevSecOps as alternative to Waterfall model where security is treated as non-functional requirement and is addressed at later stages of development





# Secure Continuous Delivery Pipeline

- Design
  - What do you intend your software to do? Did you make sensitive flows secure?
- Inherit
  - What software have you inherited, such as libraries and dependencies?
- Develop
  - As you develop, do you write security tests?
- Build
  - As you build your software, do you have security acceptance?
- Deploy
  - What happens to get software into production? Are the secrets and config being kept safe?
- Operate
  - Are you under active attack at this moment? What is getting attacked?



# Secure Continuous Delivery Pipeline

- Design
  - What do you intend your software to do? Did you make sensitive flows secure?
- Inherit
  - What software have you inherited, such as libraries and dependencies?
- Develop
  - As you develop, do you write security tests?
- Build
  - As you build your software, do you have security acceptance?
- Deploy
  - What happens to get software into production? Are the secrets and config being kept safe?
- Operate
  - Are you under active attack at this moment? What is getting attacked?

**security checking at all stages – Important part of DevSecOps**



# Secure Development

- 3 key practices
  - **Threat modeling**
  - **Development Standards**
  - Static code analysis
- Threat modeling using STRIDE by Microsoft
  - Spoofing of user identity
  - Tampering
  - Repudiation
  - Information disclosure
  - Denial of Service
  - Elevation of privilege
- More agile tools based on OWASP standard
  - OWASP App Threat Modeling Cheat Sheet
  - OWASP Application Security Verification Standards
  - Mozilla Rapid Risk Assessment



# Security Development Practices/Tools

- SAST vs DAST – Static/Dynamic Apps Security Testing
  - White vs black box testing == Develop vs Build testing
- Static code analysis
  - Open Source SAST options – some SAST work in IDE
    - PHP – Phan
    - Java Web Apps – Find Security Bug
    - Node – NodeJsScan
    - Golang/Go - GoSec
  - Commercial
    - Veracode
    - Checkmarx
    - Synopsys
  - Problem: high level of false positive
- **git-secrets and git-hound**



# Cloud Security Configuration Monitoring

- AWS Tools
  - AWS Config – Monitor configuration changes
  - AWS CloudTrail - Create a trail to retain a record of events
  - Amazon Inspector - analyzes the behavior of AWS resources and helps identify potential security issues
  - Amazon GuardDuty – Activity monitoring & Intelligent threat detection
- Third party tools
  - <https://www.threatstack.com>
  - <https://www.alienvault.com>
  - <https://evident.io> – multicloud solution
- InSpec is compliance as code service <https://www.inspec.io>
  - Turns compliance, security, and other policy requirements into automated tests
  - Includes compliance requirements into code