Application Security

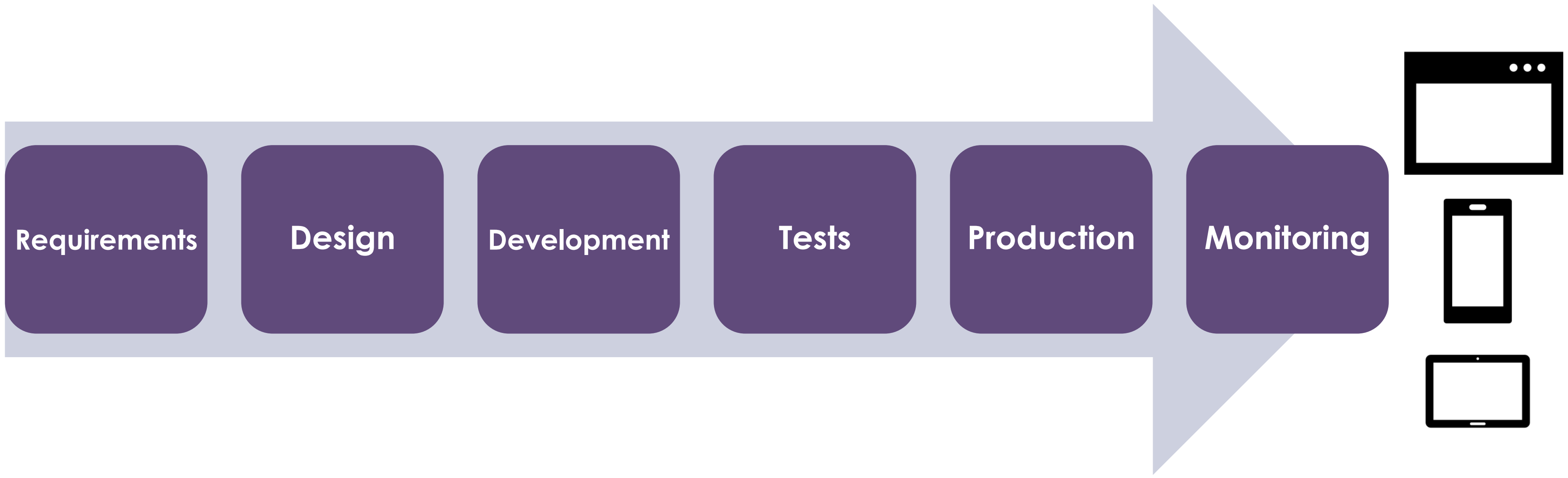
UA.DETI.IES

OWASP

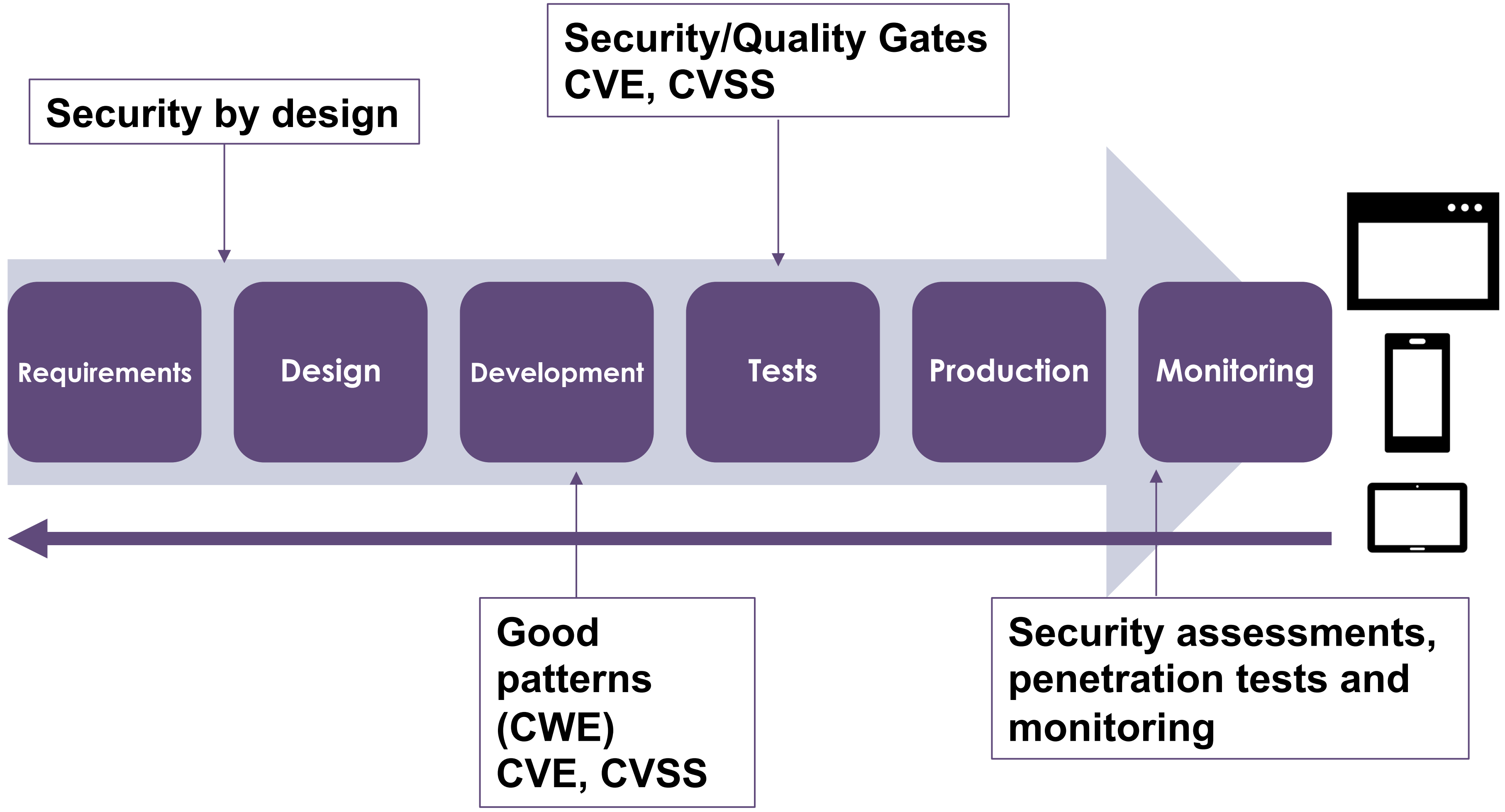
- Open Source Web application Security Project
  - Associação sem fins lucrativos
  - Suportada por uma comunidade internacional
- Opera como um agregador/normalizador de:
  - Boas práticas
  - Ferramentas
  - Metodologias de teste
  - Estratégias
- Estruturado em:
  - Projetos
    - Ferramentas
    - Standards
  - Organizações locais
  - Eventos
  - Grupos

https://owasp.org/

Application Security



Application Security



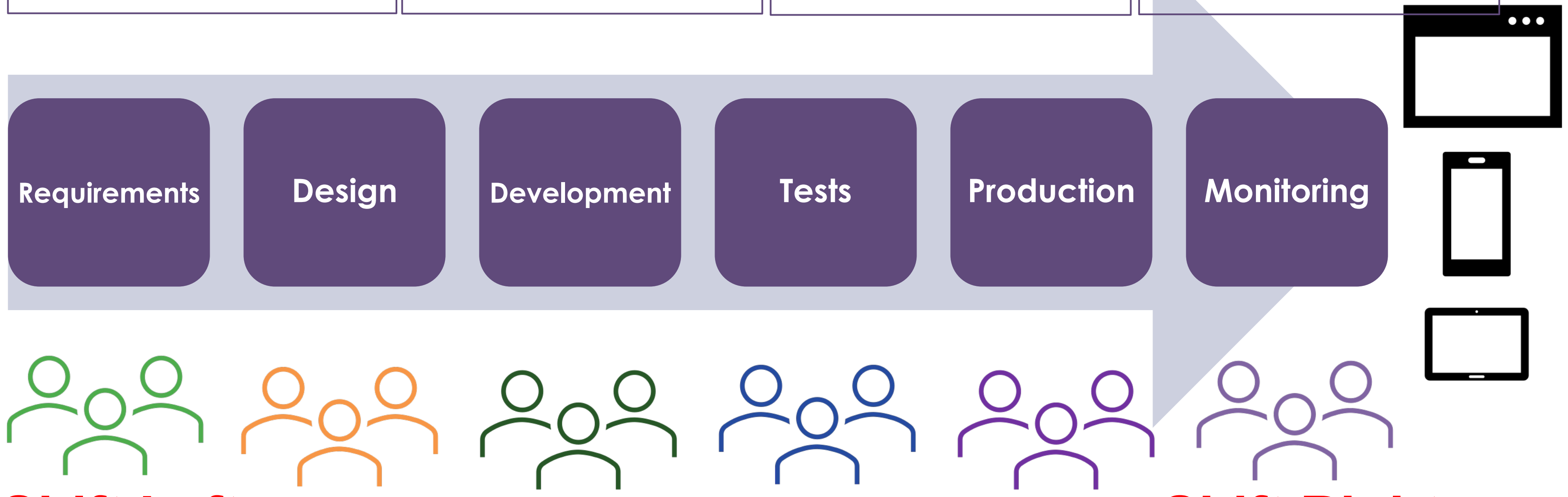
Security shift

Threat Modeling
Risk assessment
Security Requirements
Search

Standards and requirements
IDEs and Tools
Code Reviews
SAST

IAST/DAST
Penetration tests.
Abuse Tests

Customer support
Vulnerability Management
WAF
RASP



Shift Left

Shift Right

Security Shift

- **Shift Left:** Results in barely visible security
  - Application is inherently secure "by design"
  - It is a good practice
  - Legally "required"
  - Generates a safe culture, which understands the risk of the app

- **Shift Right:** Results in very present security
  - Tools throughout the SDLC
  - Reactive to vulnerabilities, introducing features into the backlog
  - You can create Security Gates that make life difficult for programmers
    - Which leads to the creation of exceptions

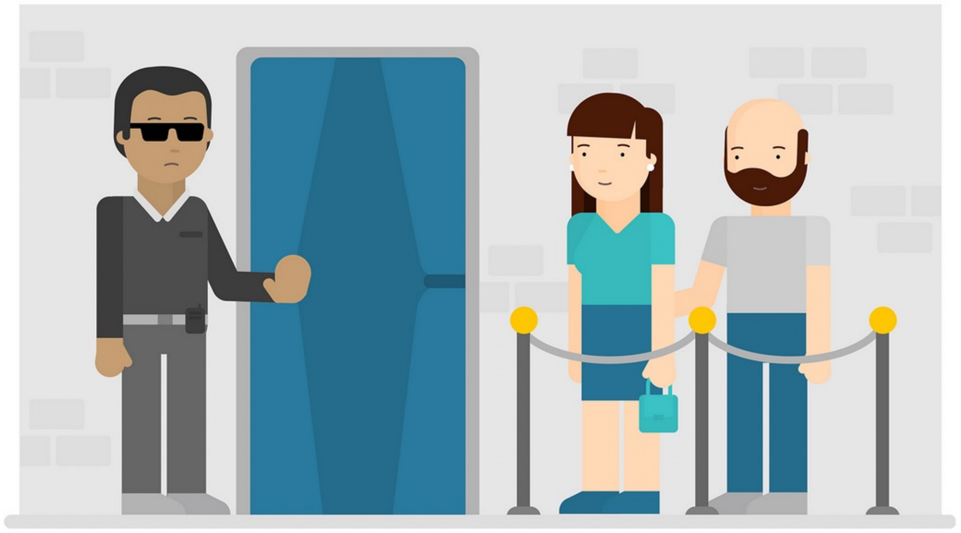
Security Shift - Fails

- **Shift Left** fails because:
  - Implies a long design process
  - Security champions can be sidelined to speed up app
  - Lack of visibility leads to poor value recognition
    - Champions begin to be removed

- **Shift Right** fails because:
  - The tools create a very high weight
    - Exceptions for case A may allow case B
  - Backlog increases with features not aligned with the product
  - Does not require a safety culture
    - Only the ability to pass Security Gates

Security Shift – which one?

- Shift Left implies that the team
  - have adequate training
  - develop a culture that includes safety
  - design application with security requirements
  - understand the attack surface and the risk of an exploit
  - understand vulnerabilities and patterns that create them



Security Shift – which one?

- Consider some **Shift Right** to hold non-design-dependent aspects

- Some aspects:
  - Implemented code bad practices
  - Errors in dependencies
  - Supplier errors
  - Infrastructure errors
  - New vulnerability categories
  - Requirements will not capture all usage/interactions

CWE – Patterns and weaknesses

- Bugs and errors are **introduced by poor programming**
  - Unlike failures that refer to requirements/design
- Effectively fixing a bug/vulnerability implies **implementing the correct code**
  - Who is often not obvious
- Fixing a bug should **focus on the process that causes the bug** and not just the bug

CWE – Patterns and weaknesses

- In an SSDLC it must be considered that **not all bugs are equal**
- Vulnerability is a bug with a possibility of exploitation
  - It then changes the risk profile of an application, increasing it
  - Operational, legal, brand risks…
- Programmers **should not create bugs**
  - Programmers shouldn't even create vulnerabilities

CWE – Patterns and weaknesses

- Stands for Common Weakness Enumeration
- Common language that allows you to **enumerate anti-patterns that lead to weaknesses**
- **Weakness**
  - Condition that contributes to the introduction of vulnerabilities
- Hierarchical organization
  - 839 CWEs

CWE – Patterns and weaknesses

- Describe and discuss weaknesses in a **uniform way**
- **Check for weaknesses** in software and hardware
- Verify that tools address weaknesses
  - SAST tools often provide analyses based on CWEs
- Basis for identifying, mitigating and preventing weaknesses
- Prevent software and hardware from having vulnerabilities

CWE-348: Use of Less Trusted Source

- *The software has two different sources of the same data or information, but it uses the source that has less support for verification, is less trusted, or is less resistant to attack.*

- Details:
  - https://cwe.mitre.org/data/definitions/348.html
  - Describes the standard, provides examples and lists of related CVEs

CWE-348: Use of Less Trusted Source

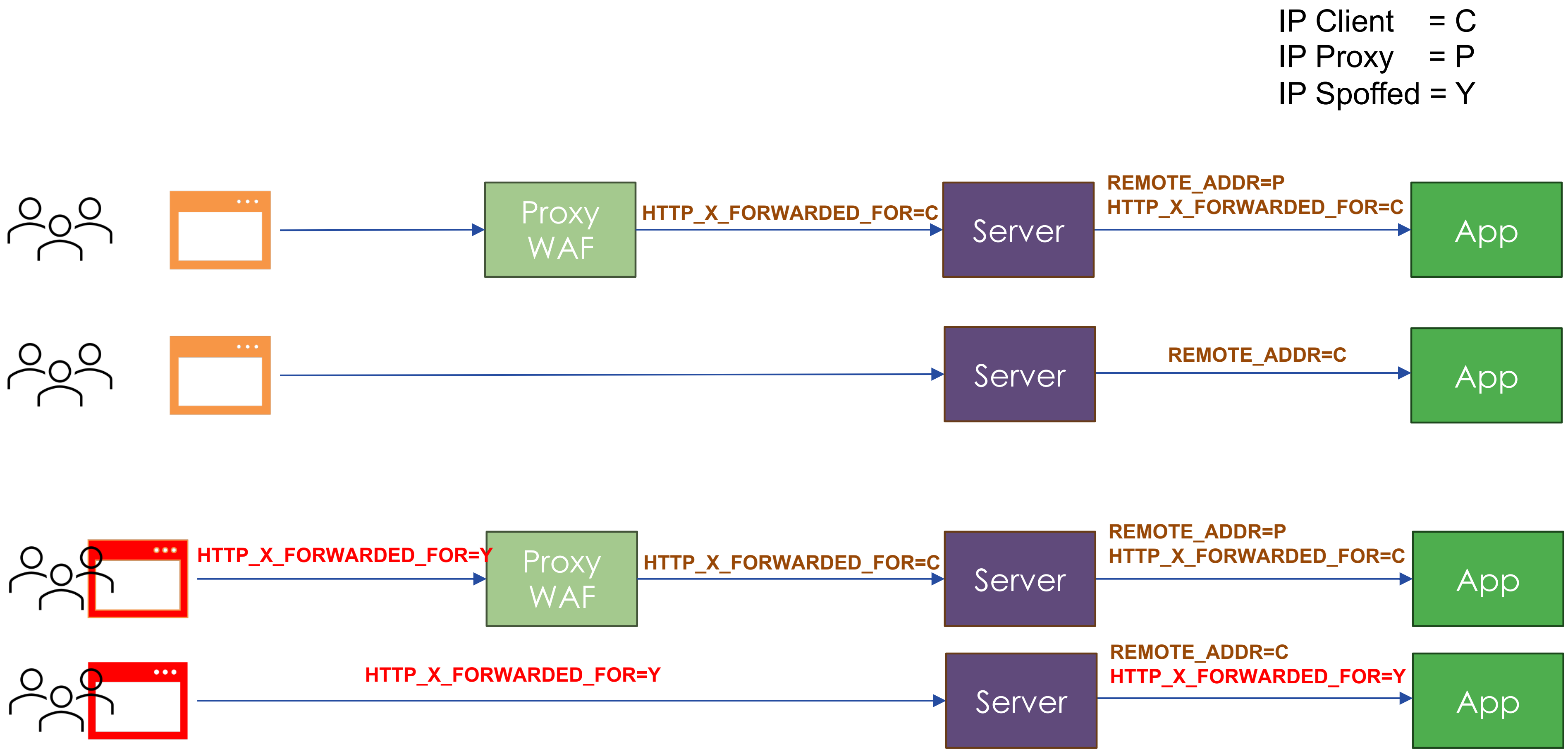
```
$requestingIP = '0.0.0.0';
if (array_key_exists('HTTP_X_FORWARDED_FOR', $_SERVER)) {
        $requestingIP = $_SERVER['HTTP_X_FORWARDED_FOR'];
else{
        $requestingIP = $_SERVER['REMOTE_ADDR'];
}

if(in_array($requestingIP,$ipAllowlist)){
        generatePage();
        return;
}
else{
        echo "You are not authorized to view this page";
        return;
}
```

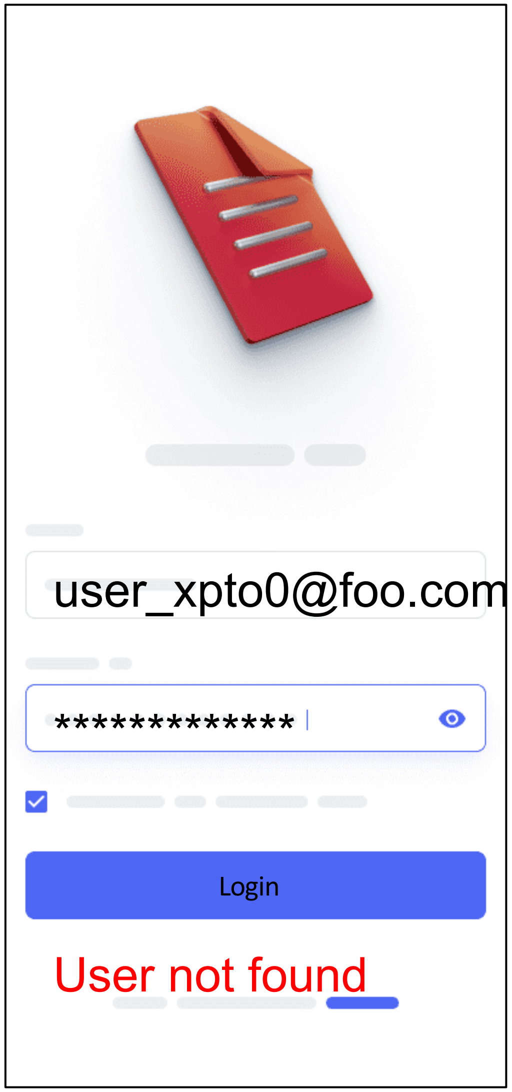
Set by Proxy… or Client

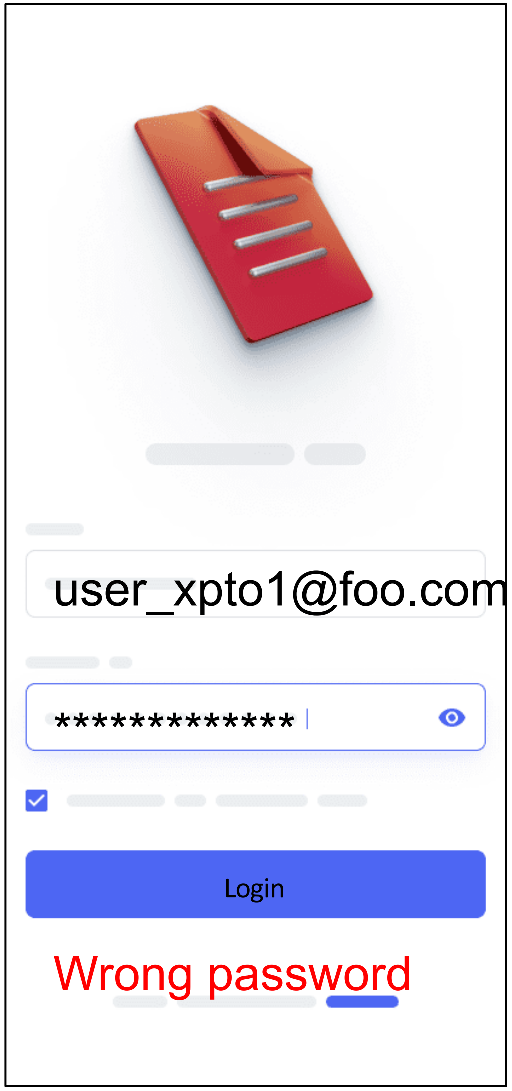
Set by web server

CWE-348: Use of Less Trusted Source



CWE?


user_xpto0@foo.com
****
Login
User not found


user_xpto1@foo.com
****
Login
Wrong password

CWE?

CWE-203: Observable Discrepancy

The product behaves differently or sends different responses under different circumstances in a way that is observable to an unauthorized actor, which exposes security-relevant information about the state of the product, such as whether a particular operation was successful or not.

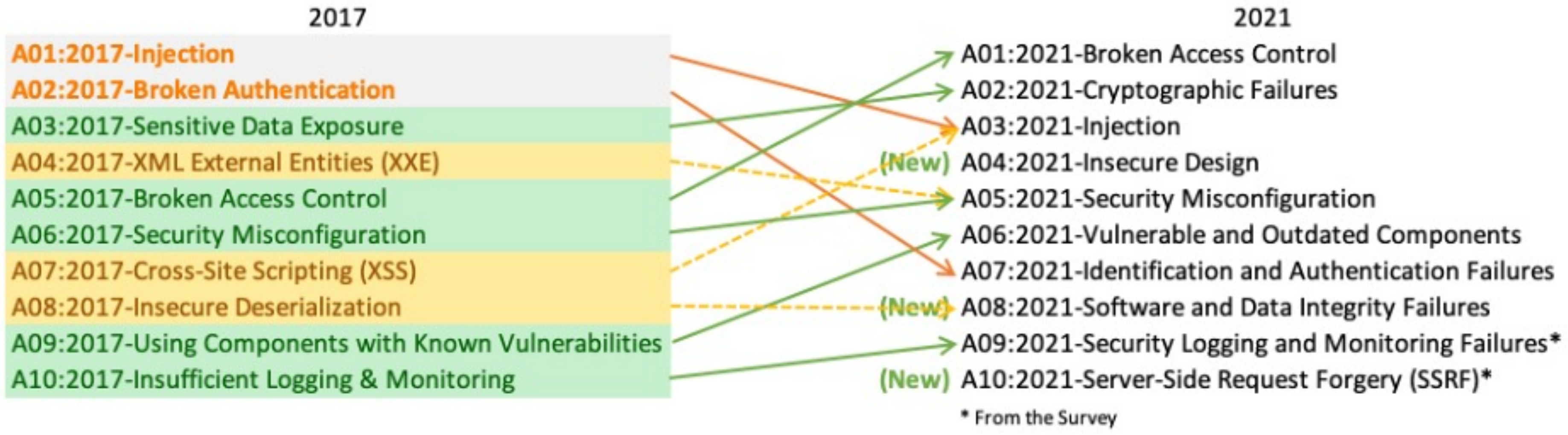
Discrepancies can take many forms, and variations may be detectable in timing, control flow, **communications such as replies** or requests, or general behavior. These discrepancies can reveal information about the product's operation or internal state to an unauthorized actor. In some cases, discrepancies can be used by attackers to form a side channel.

https://cwe.mitre.org/data/definitions/203.html

Risks – OWASP Top 10

- List of the **10 most frequent risks** in web apps
  - Updated every 3-4 years
- List generated after some community consensus
  - Public discussion
  - It should reflect the problems encountered in reality
- Probably one of the most discussed aspects of AppSec

Risks – OWASP Top 10



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

Risks – OWASP Top 10

- It raises awareness but **is not a norm**
  - Limited usefulness in application design

- Not a complete list: just the 10 "most relevant"

- Allows programmers to learn about problems
  - Allows you to create campaigns to avoid problems
  - And know the risk of some "bugs"

- Does **not allow defining an AppSec strategy**

Risks – OWASP Top 10

https://owasp.org/Top10/A01_2021-Broken_Access_Control/

A01:2021 – Broken Access Control

CWEs Mapped	Max Incidence Rate	Avg Incidence Rate	Avg Weighted Exploit	Avg Weighted Impact	Max Coverage	Avg Coverage	Total Occurrenc es	Total CVEs
34	55.97%	3.81%	6.92	5.93	94.55%	47.72%	318,487	19,013

- Describe the problem in its most common aspects
- Very briefly defines how it can be prevented
- Demonstrate with **attack scenarios**
- Maps to CWEs

OWASP ASVS

- Application Security Verification Standard
  - v4.0.3 – October 2021
  - v5.0 in creation

- Base for defining tests and technical controls for web applications
  - as well as requirements for an SSDLC

- Resources:
  - Page: https://owasp.org/www-project-application-security-verification-standard/
  - Development: https://github.com/OWASP/ASVS

OWASP ASVS – Goals

- Define requirements for a secure application
  - Abstract, capable of being implemented in most languages in an environment
  - Different requirements for Web, Mobile and IoT
- Act as a standard for secure applications
  - Be useful early in development
  - Be useful for considering controls
  - Be useful in acquiring applications
- **Based on good practices and existing standards**
- Scalable (by levels)

OWASP ASVS – Alignments

- Alignment with:
  - NIST 800-63-3 Digital Identity Guidelines
  - ISO/IEC 27034-1:2011
  - PCI DSS 3.2.1
  - GDPR
  - OWASP Top 10 2017
  - OWASP Proactive Controls 2018
  - CWE

- Related:
  - Mobile ASVS (MASVS)
  - IoT ASVS

Requirements

- What requirements should you **define for the storage and management of a file made available on a page**?

- We can consider:
  - Be access controlled
  - Served to end customers
  - Web
  - Minimum risk level

- But… service compromise exposes personal data?Does it harm the brand? Violates a law/regulation/normative? Does it allow escalation of privileges?

Requirements

- **What requirements can we define for a process?**

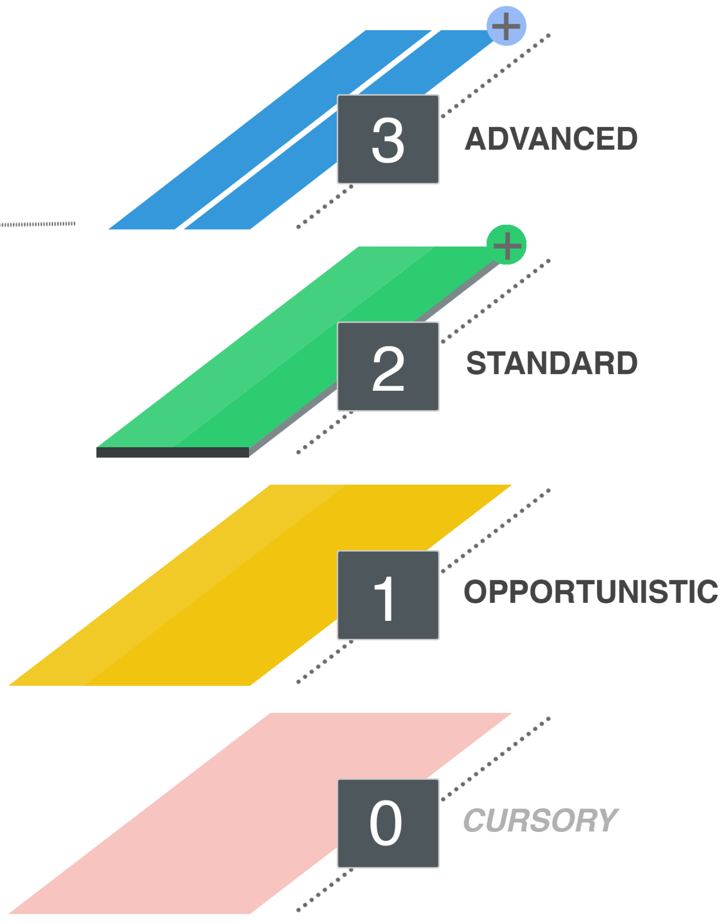
- Will they be enough?
- Will they be aligned with current risks?
- Will they be aligned with the criticality of the application?
- Are they suitable for the legal/regulatory environment?
- For a new process, do we restart the requirements selection process? From what point?

Requirements

- With OWASP Top 10 and Proactive Controls we know:
  - Some problems we should avoid
    - But what about the others?
  - What is the impact on risk
    - Is it the same for all applications?
  - How to avoid some problems
    - Some
- OWASP Top 10 and Proactive Controls are far to the right
- A useful framework is needed at the requirements level (Shift Left)

OWASP ASVS Levels

ASVS DEFINES DETAILED VERIFICATION REQUIREMENTS FOR LEVELS 1 AND ABOVE; WHEREAS LEVEL 0 IS MEANT TO BE FLEXIBLE AND IS CUSTOMIZED BY EACH ORGANIZATION



OWASP ASVS LEVELS

OWASP ASVS Level 0

- Level 0 does not have a set of requirements
- Entities can “cherry pick” requirements
- ASVS continues to be useful, but entities must consider more!
- **Applications at Level 0 do not follow current best practices**

OWASP ASVS Level 1

- Considered the **minimum** for any application
- Completely testable from abroad without documentation
  - Black Box Pen Test (Human Assisted)
  - Partially testable by SAST and DAST applications
  - Considers the most common vulnerabilities and attacks
- **~130 Requirements**

0
1
2
3
OWASP ASVS LEVELS

OWASP ASVS Level 1

- ASVS recommends discontinuation of Penetration tests

- Encourages hybrid audits
  - Code Reviews
  - Grey/White Box

0
1
2
3
OWASP ASVS LEVELS

OWASP ASVS Level 2

- Considered **suitable** for any application
- Defined for applications with sensitive data
  - Areas such as B2B transactions, Commerce, Games

- Intends to protect application from specialized attackers

- **+ ~130 requirements**

0
1
2
3
OWASP ASVS LEVELS

OWASP ASVS Level 2

- It is not testable from the outside without access to internal information

- Some validations are procedural:
  - Do you have a tool to manage defects?
  - Use version control?
  - Do you have an SDLC with security?
  - Do you have Secure and Repeatable Deployments?

0
1
2
3
OWASP ASVS LEVELS

OWASP ASVS Level 3

- Considered as necessary for **critical applications**

- Defined for applications with very sensitive data
  - Areas such as military environments, healthcare, critical infrastructure

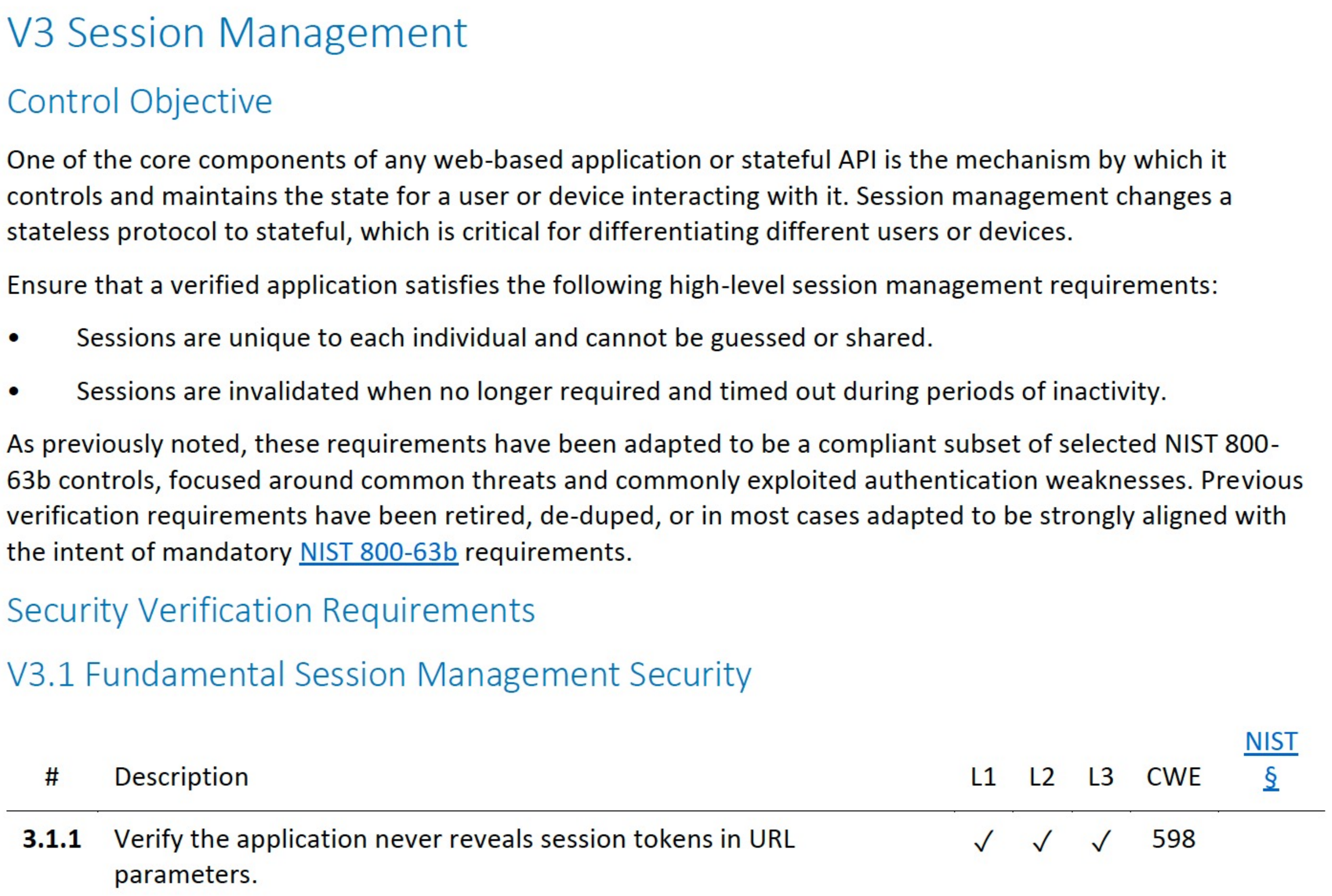
- + ~20 requirements

0
1
2
3
OWASP ASVS LEVELS

OWASP ASVS Structure

- Organized in a fixed structure: **<chapter.section.requirement>**
- **Chapters**: scope of requirements
  - 1.X.X = Architecture
- **Sections**: Further specifies the scope of the chapter
  - 1.7.X = All "Errors, logging and audit architecture" requirements
- **Requirement**: The specific requirement
  - 1.7.2 = Refers to the need for secure remote records (L2)

OWASP ASVS Chapter Structure (3)



V3 Session Management

Control Objective

One of the core components of any web-based application or stateful API is the mechanism by which it controls and maintains the state for a user or device interacting with it. Session management changes a stateless protocol to stateful, which is critical for differentiating different users or devices.

Ensure that a verified application satisfies the following high-level session management requirements:

- Sessions are unique to each individual and cannot be guessed or shared.
- Sessions are invalidated when no longer required and timed out during periods of inactivity.

As previously noted, these requirements have been adapted to be a compliant subset of selected NIST 800-63b controls, focused around common threats and commonly exploited authentication weaknesses. Previous verification requirements have been retired, de-duped, or in most cases adapted to be strongly aligned with the intent of mandatory NIST 800-63b requirements.

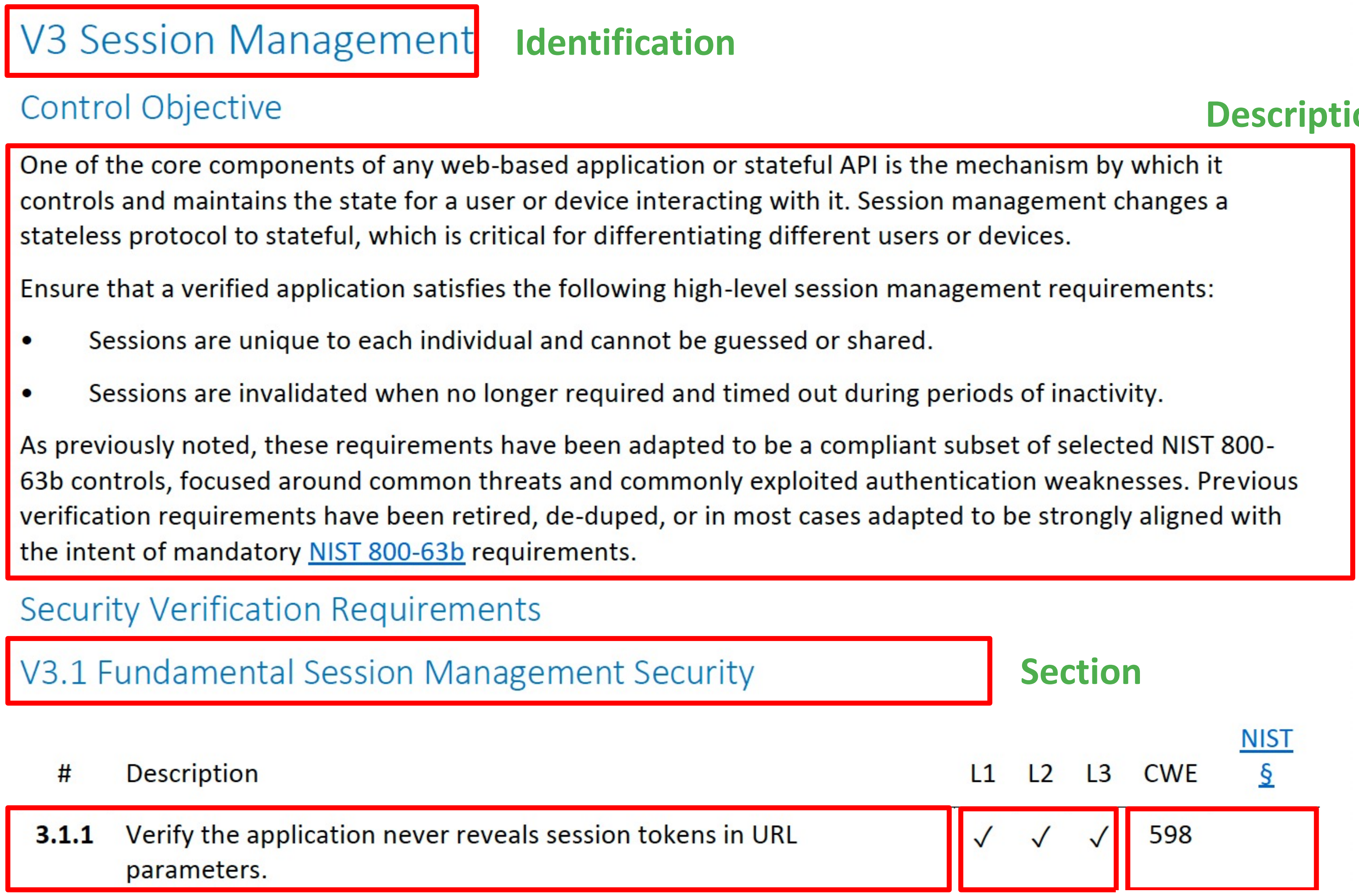
Security Verification Requirements

V3.1 Fundamental Session Management Security

#	Description	L1	L2	L3	CWE	NIST §
3.1.1	Verify the application never reveals session tokens in URL parameters.	✓	✓	✓	598	

UNIVERSIDADE DE AVEIRO

OWASP ASVS Chapter Structure (3)



V3 Session Management

Identification

Control Objective

Description

One of the core components of any web-based application or stateful API is the mechanism by which it controls and maintains the state for a user or device interacting with it. Session management changes a stateless protocol to stateful, which is critical for differentiating different users or devices.

Ensure that a verified application satisfies the following high-level session management requirements:

- Sessions are unique to each individual and cannot be guessed or shared.
- Sessions are invalidated when no longer required and timed out during periods of inactivity.

As previously noted, these requirements have been adapted to be a compliant subset of selected NIST 800-63b controls, focused around common threats and commonly exploited authentication weaknesses. Previous verification requirements have been retired, de-duped, or in most cases adapted to be strongly aligned with the intent of mandatory NIST 800-63b requirements.

Security Verification Requirements

V3.1 Fundamental Session Management Security

Section

#	Description	L1	L2	L3	CWE	NIST §
3.1.1	Verify the application never reveals session tokens in URL parameters.	✓	✓	✓	598	

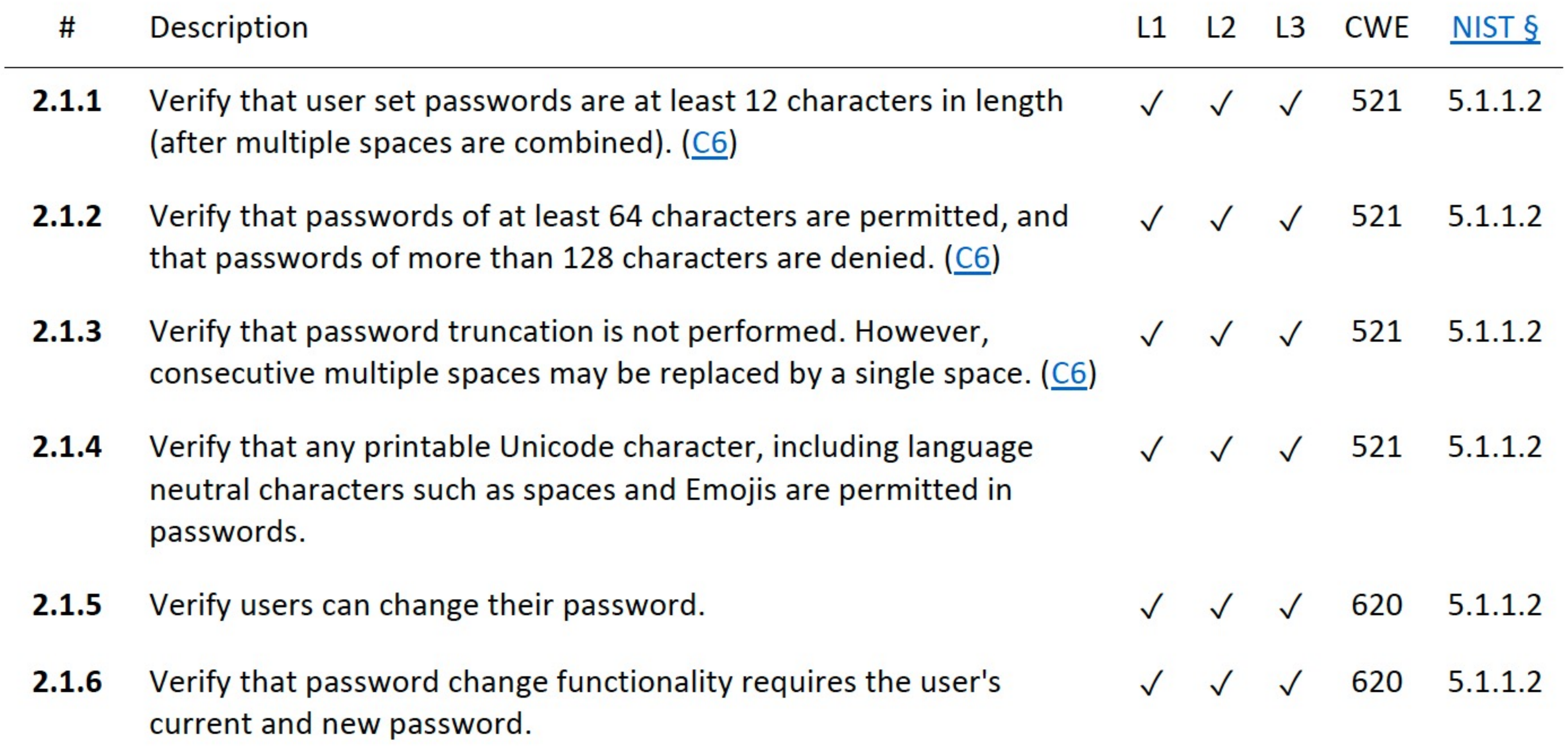
Requirement

Levels

References

OWASP ASVS Important Points

- Aligned with standards and norms



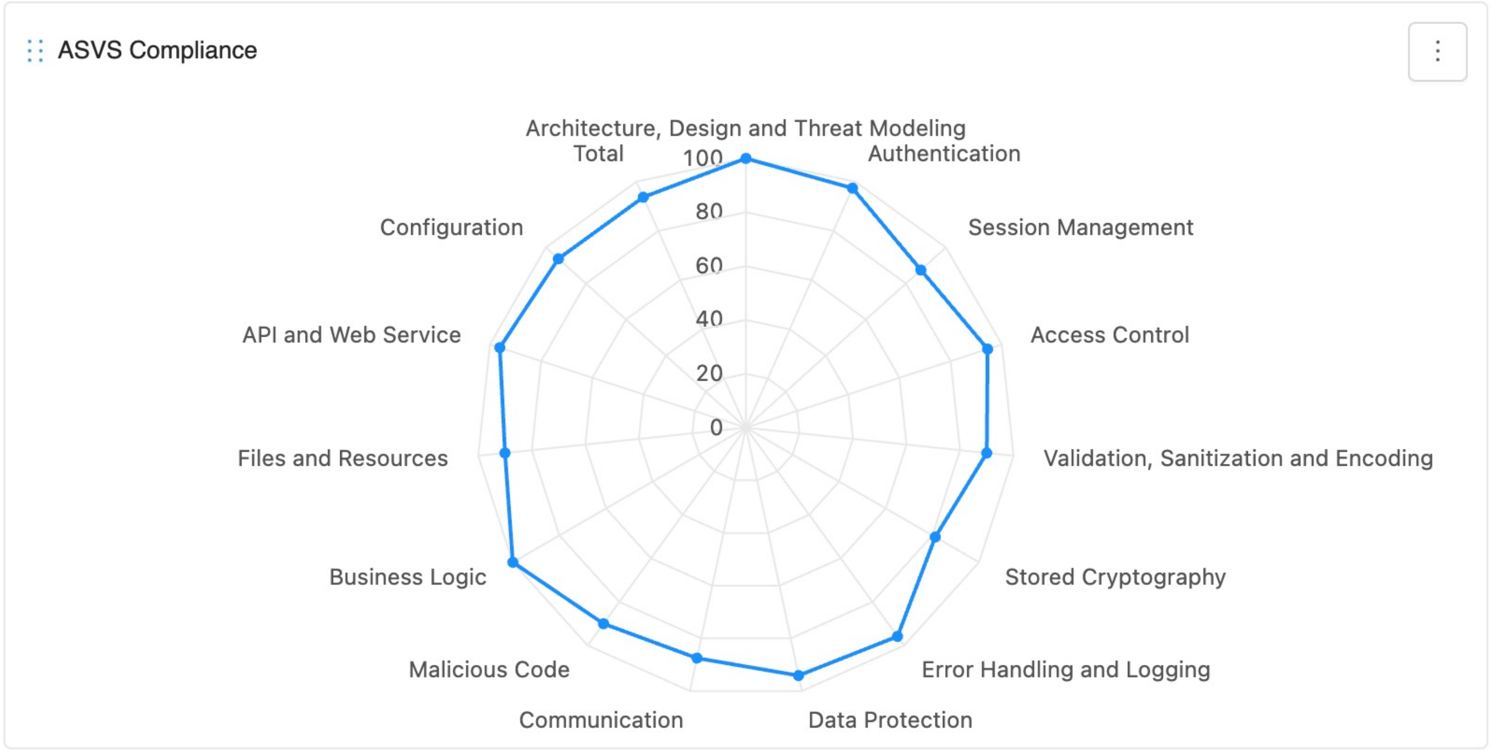
#	Description	L1	L2	L3	CWE	NIST §
2.1.1	Verify that user set passwords are at least 12 characters in length (after multiple spaces are combined). (C6)	✓	✓	✓	521	5.1.1.2
2.1.2	Verify that passwords of at least 64 characters are permitted, and that passwords of more than 128 characters are denied. (C6)	✓	✓	✓	521	5.1.1.2
2.1.3	Verify that password truncation is not performed. However, consecutive multiple spaces may be replaced by a single space. (C6)	✓	✓	✓	521	5.1.1.2
2.1.4	Verify that any printable Unicode character, including language neutral characters such as spaces and Emojis are permitted in passwords.	✓	✓	✓	521	5.1.1.2
2.1.5	Verify users can change their password.	✓	✓	✓	620	5.1.1.2
2.1.6	Verify that password change functionality requires the user's current and new password.	✓	✓	✓	620	5.1.1.2

OWASP ASVS Usage

- In specifying **new product requirements**
- In specifying **security gateways for existing SDLC**
- In identifying/characterization of application risks
- When selecting purchased applications
  - Dependency selection

OWASP ASVS Usage

- Define the desired level
  - Aligned with application purpose and operating environment
  - Try level 2



OWASP ASVS Usage

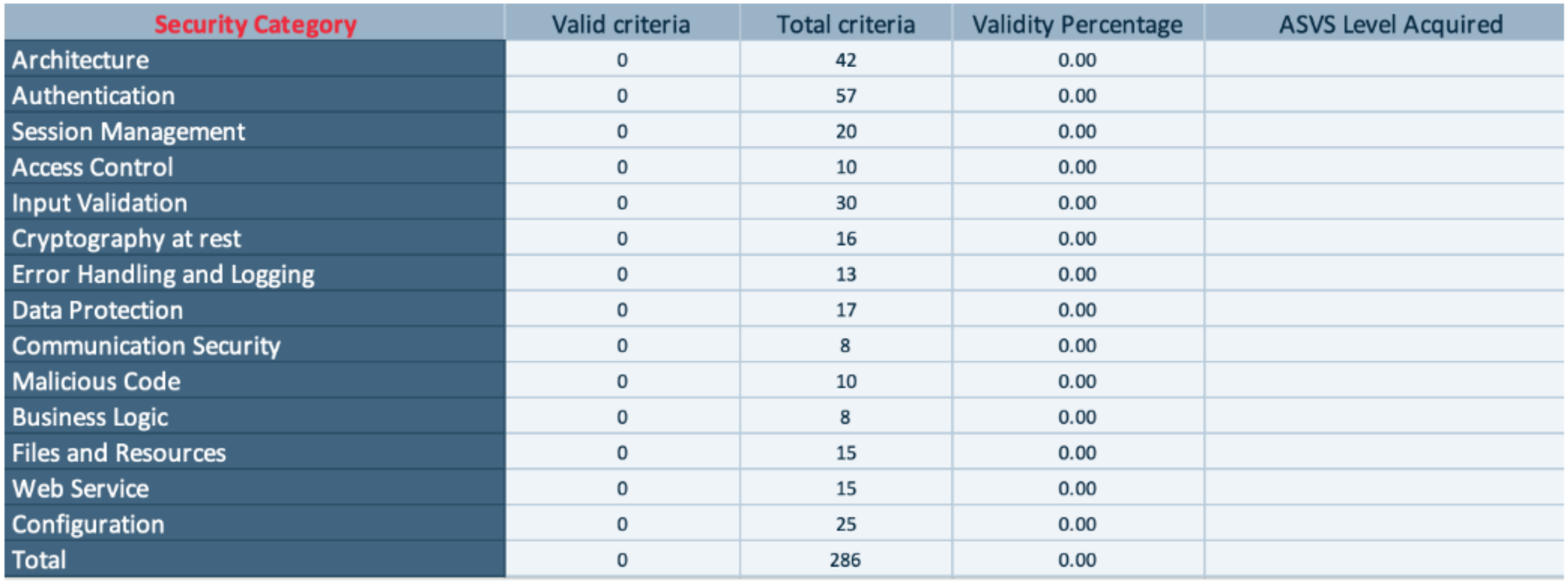
- Start with V1.1 Secure Software Development Lifecycle
  - Have security at all points: user stories? Unit tests? Quality Gates?
  - Model the attack surface
  - Have security restrictions on user stories
  - Document and justify confidence limits
  - Verify high-level security, including external services
  - Check for reusable and global controls
  - Check the existence of good practice standards for programmers

- None of these requirements are necessary for L1

OWASP ASVS Usage

- Result in a list of requirements and a checklist

- Various resources to calculate
  - https://github.com/shenril/owasp-asvs-checklist



Security Category	Valid criteria	Total criteria	Validity Percentage	ASVS Level Acquired
Architecture	0	42	0.00	
Authentication	0	57	0.00	
Session Management	0	20	0.00	
Access Control	0	10	0.00	
Input Validation	0	30	0.00	
Cryptography at rest	0	16	0.00	
Error Handling and Logging	0	13	0.00	
Data Protection	0	17	0.00	
Communication Security	0	8	0.00	
Malicious Code	0	10	0.00	
Business Logic	0	8	0.00	
Files and Resources	0	15	0.00	
Web Service	0	15	0.00	
Configuration	0	25	0.00	
Total	0	286	0.00	