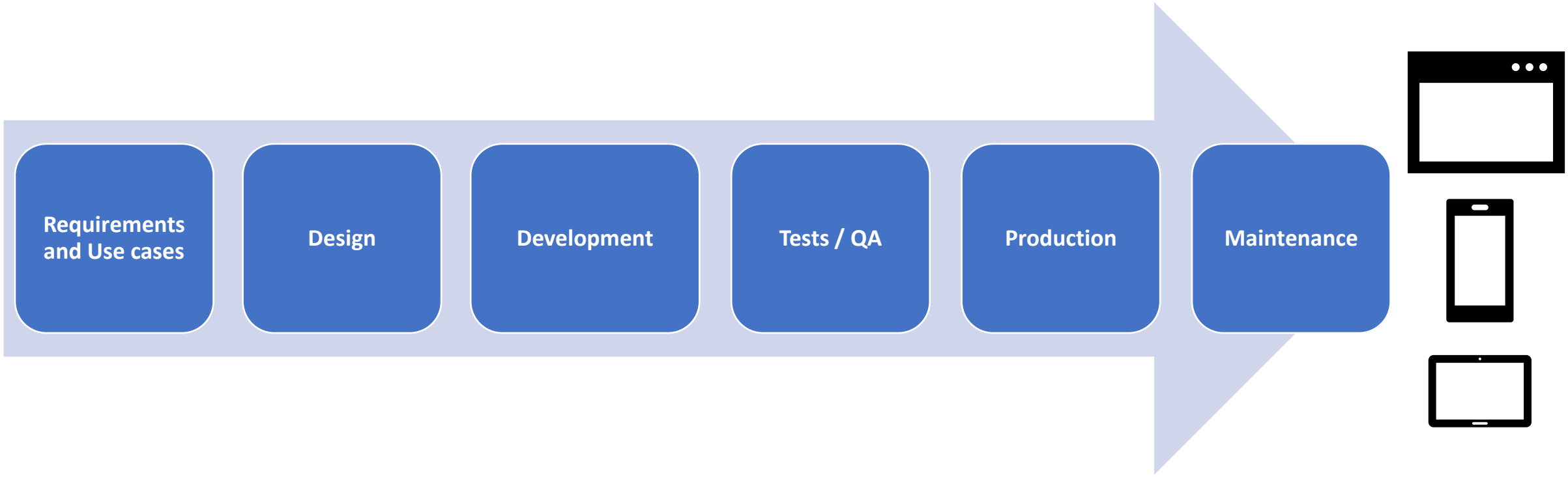
Secure Software Development

SIO

deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

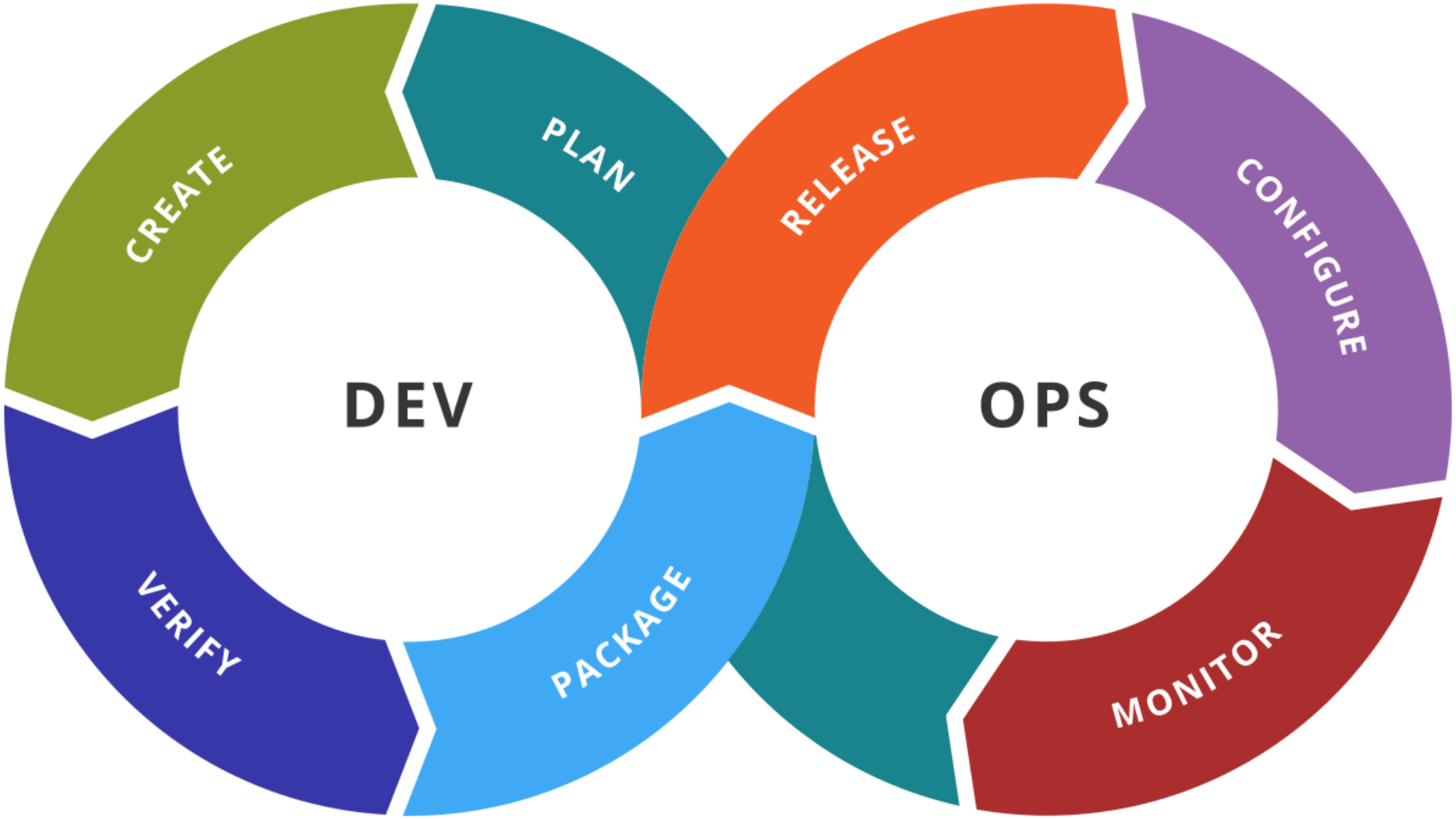
João Paulo Barraca

The Software Development Life Cycle (SDLC)



Implemented following popular models: Agile, Lean, Waterfall, Iteractive, Spiral, DevOps…..

The DevOps model



Secure Software Development

The Problem

- Software is developed towards a functional objective
  - Considering use cases, requirements and features (in an agile process)

- Security is frequently an afterthought
  - Unless security brings business value, use cases are typically feature oriented

As it is not effective to develop a random software and then add the desired features … **it is not effective to develop a software and then make it secure**

Secure Software Development

The Problem

- Software development is complex
  - Involves several collaborators/teams, several software packages and dependencies
  - Results in multiple artifacts, deployed over a potentially wide ecosystem
  - Easy for development to derail towards wrongly implemented features

As the software becomes more complex, the exposition and opportunities to attackers increase, **becoming hard remove leaked information or to ensure a secure development chain**

Secure Software Development

The Problem

- Software increasingly deals with **sensitive information**
  - Private information from users
  - Confidential information from users or businesses
  - Classified information from governments

Leaking or manipulation of information can result in high impact incidents for software users. Recovery, if possible, **may be highly expensive**

Secure Software Development

The Problem

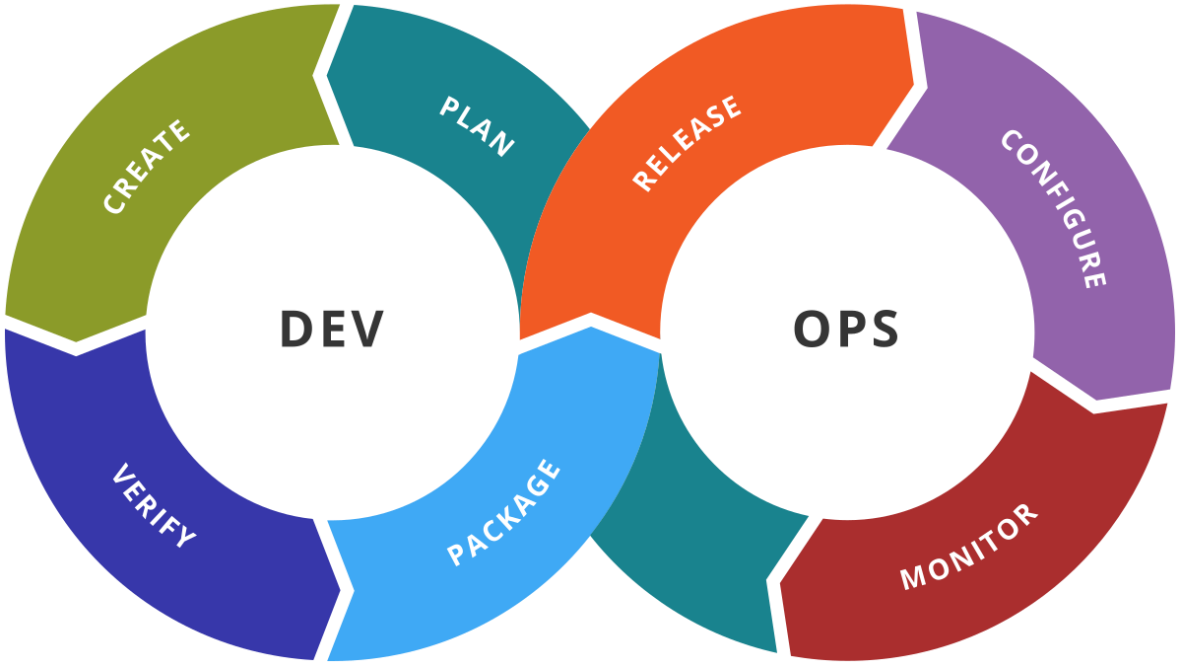
- Software may interact with the real world
  - IoT in homes and offices
  - OT in critical utilities
  - Automata in industries
  - Autonomous agents in business processes and systems (Cars)

Leaking or manipulation of information can result in high impact incidents for users. Recovery **may be highly expensive and may be impossible**

Why AppSec

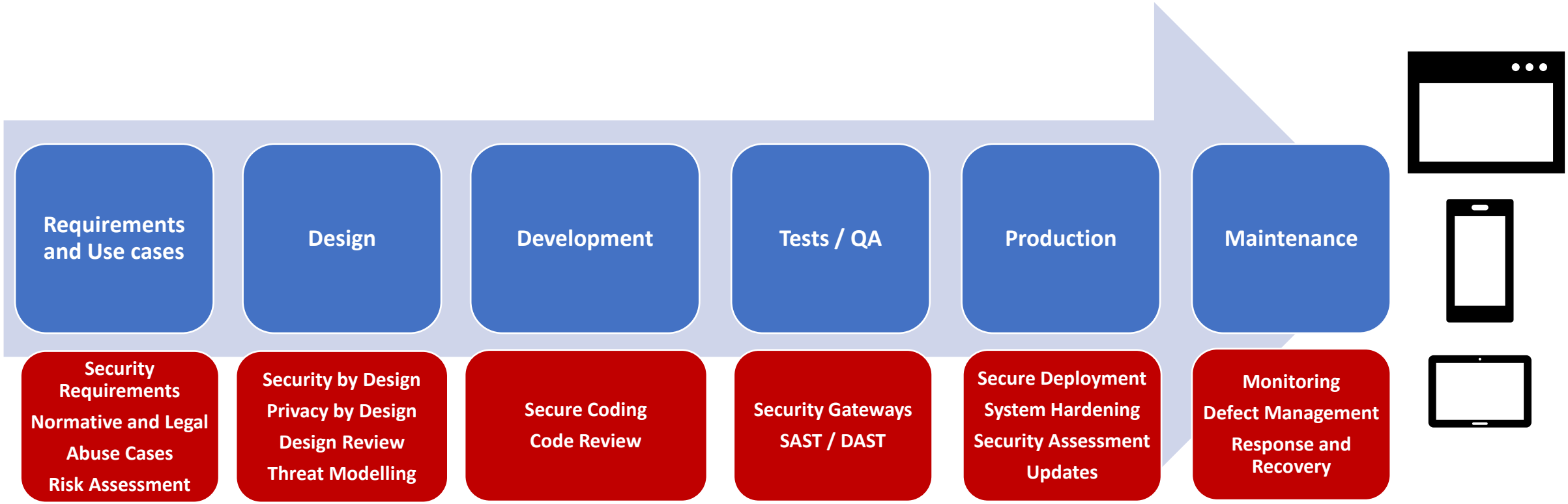
- Software is at the **center** of most activities
- Wide range of threats with more **complex techniques**
- Applications are actively explored for vulnerabilities and **Fraud**
- **Impact is broad**: brand, financial, reputation, clients
- **Regulatory** and **Legal** ramifications
- Increased demand by clients and **requirements to access market**

The Secure DevOps model in practice



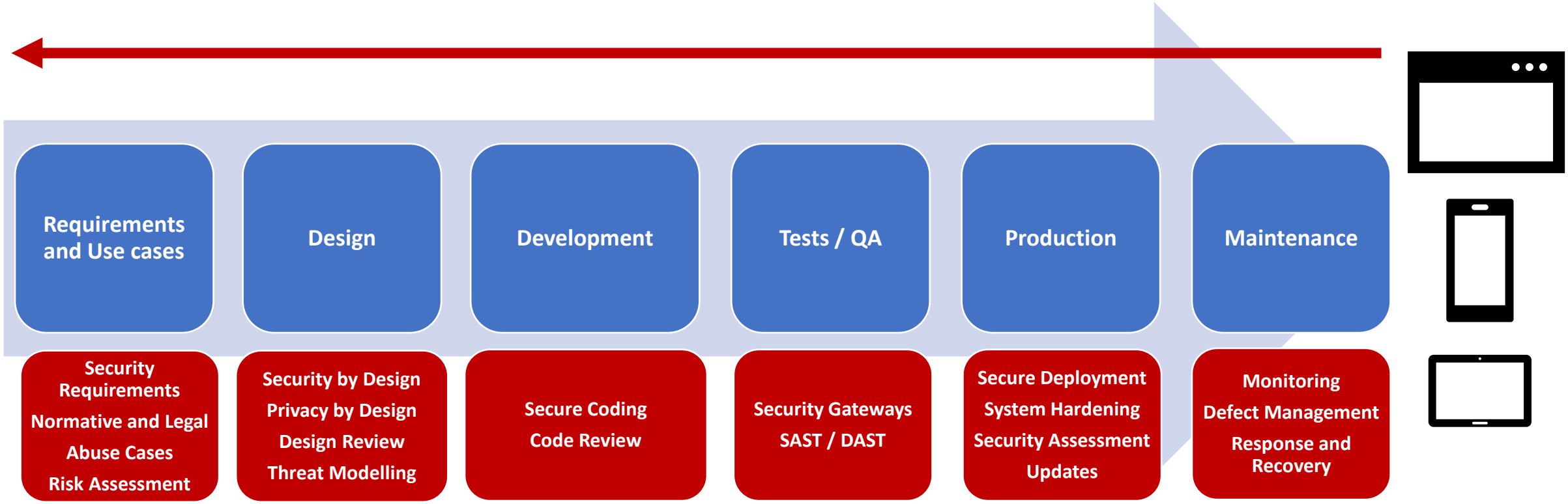
People	Processes	Technology
Collaborators Contractors Suppliers Operators Support	Regulatory compliance Development practices Client requirements	Development tools Organization tools Support tools Operational systems Dependencies

The Secure Software Development Life Cycle (SDLC)



The Secure Software Development Life Cycle (SDLC)

Shift Left: more effort towards secure requirements and design



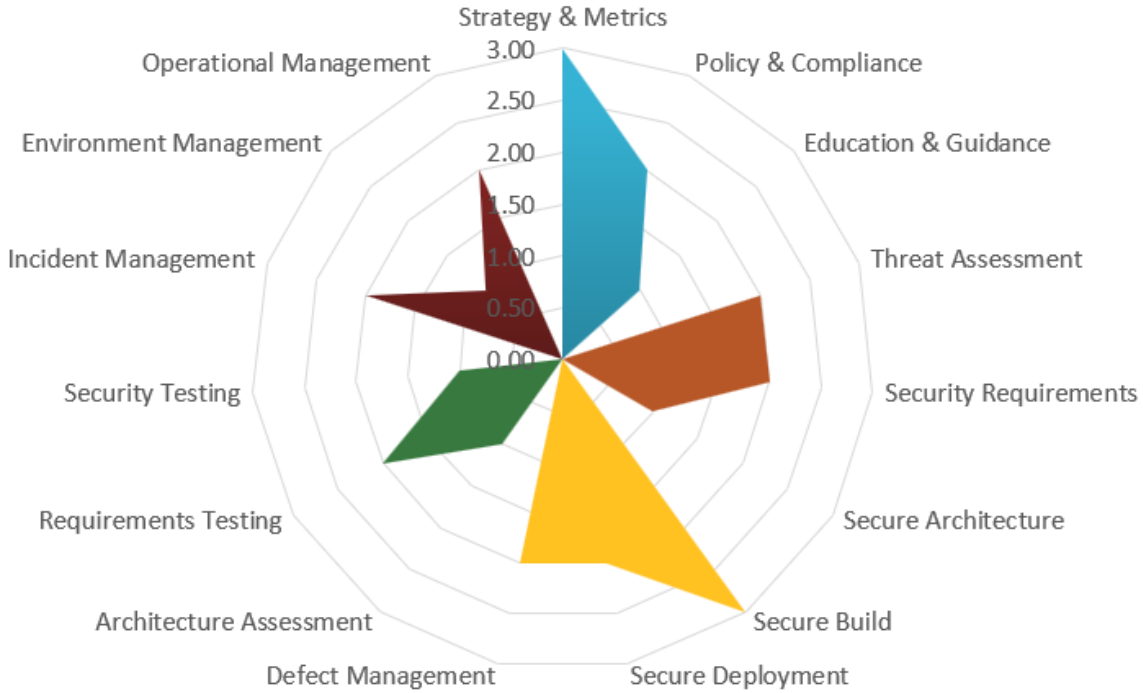
Overall SDLC Maturity

OWASP Software Assurance Maturity Model (SAMM)

- **Effective and measurable** way to analyze and improve the **secure development lifecycle**.
  - SAMM supports the complete software lifecycle and is **technology and process agnostic**.
  - SAMM is **evolutive and risk-driven** in nature, as there is no single recipe that works for all organizations.
  - Aligned with NIST CiberSecurity Framework 2.0 (NIST CSF2)
  - Ranges from Organization Strategy to Operations

- Companies can identify their location and build a path towards improving their posture

- https://owaspsamm.org/



Overall SDLC Maturity

OWASP Software Assurance Maturity Model (SAMM)

Compliance Management	**1**	**Do you have a complete picture of your external compliance obligations?**
		You have identified all sources of external compliance obligations You have captured and reconciled compliance obligations from all sources
	2	**Do you have a standard set of security requirements and verification procedures addressing the organization's external compliance obligations?**
		You map each external compliance obligation to a well-defined set of application requirements You define verification procedures, including automated tests, to verify compliance with compliance-related requirements
	3	**Do you regularly report on adherence to external compliance obligations and use that information to guide efforts to close compliance gaps?**
		You have established, well-defined compliance metrics You measure and report on applications' compliance metrics regularly Stakeholders use the reported compliance status information to identify compliance gaps and prioritize gap remediation efforts

Deployment Process	**1**	**Do you use repeatable deployment processes?**
		You have enough information to run the deployment processes Your deployment documentation up to date Your deployment documentation is accessible to relevant stakeholders You ensure that only defined qualified personnel can trigger a deployment You harden the tools that are used within the deployment process
	2	**Are deployment processes automated and employing security checks?**
		Deployment processes are automated on all stages Deployment includes automated security testing procedures You alert responsible staff to identified vulnerabilities You have logs available for your past deployments for a defined period of time
	3	**Do you consistently validate the integrity of deployed artifacts?**
		You prevent or roll back deployment if you detect an integrity breach The verification is done against signatures created during the build time If checking of signatures is not possible (e.g. externally build software), you introduce compensating measures

Planning and Requirements

Legal and Normative Compliance needs

- Specific operational areas have legal requirements
  - Processing of credit card information (a payment processor): PCI-DSS
  - Processing of user data (a shop): GDPR
  - Public sector: DL65/2021 and NIS2
  - Telecommunications: EECC, GDPR, NIS, Digital Services Act
  - Health Data (UK): HIPAA
  - Financial Sector: Digital Operational Resilience Act (DORA)
  - Others: ISO 27001, ISO 2000, Cyber Resilience Act

- Software planning must consider the legal requirements for the target market
  - Fulfilling legal framework may invalidate software for a given target security posture, functionality and value

Planning and Requirements

Legal and Normative Compliance needs

- Requirements are focused on product value, but must consider security aspects
  - Security Requirements
    - Multi-layer security protecting Confidentiality, Integrity and Availability
  - Avoid fraud and abuse
  - Allow updates and observability

- Requirements must include software and organization wide practices
  - Which tools are use
  - Who supplies them
  - How are system deployed/operated
  - How product support operates

Planning and Requirements

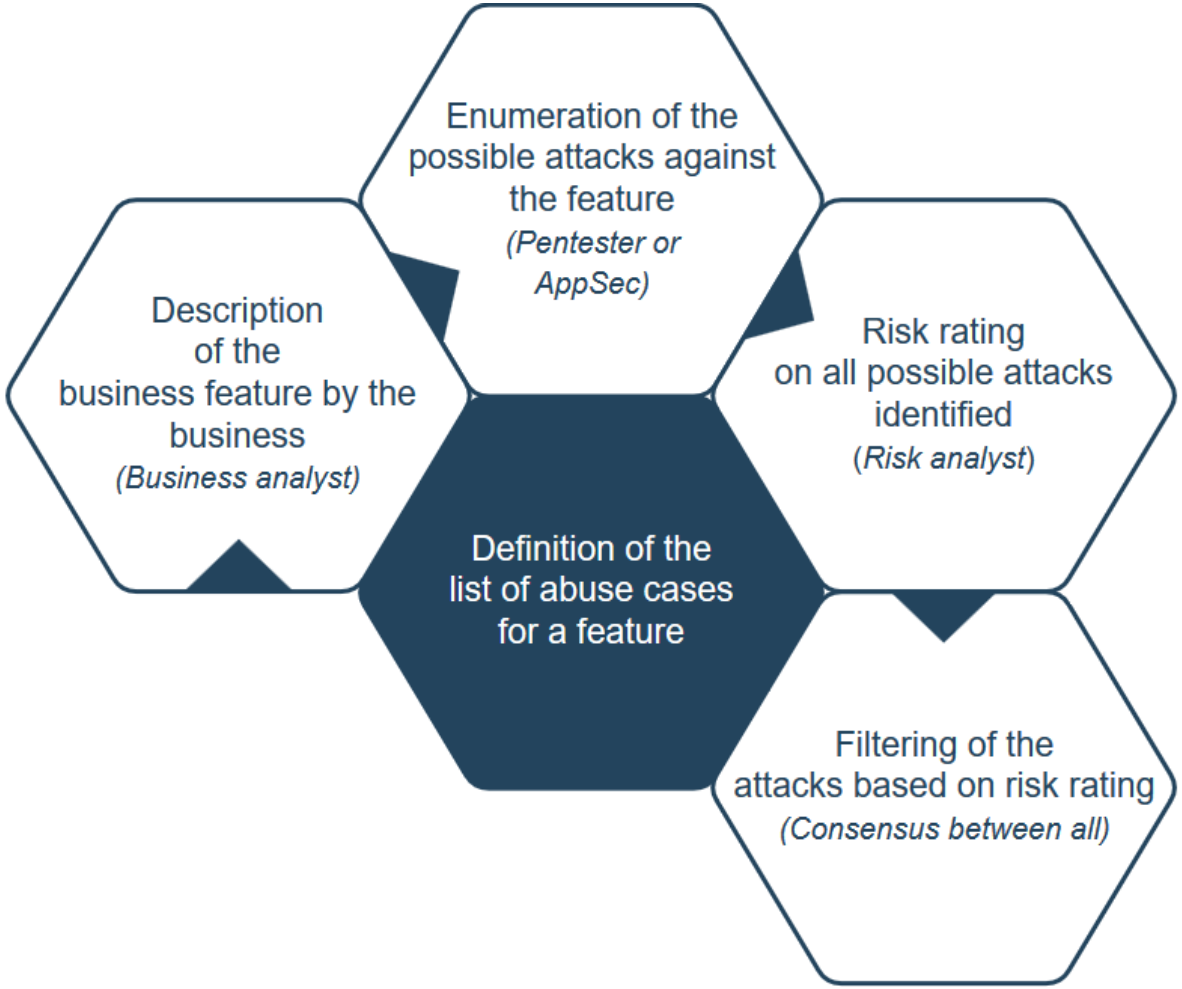
Use and Abuse Case

- **Abuse Case**: A way to use a feature that was not expected by the implementer, allowing an attacker to influence the feature or outcome of use of the feature based on the attacker action (or input).

- A Business Feature may result in a list of Abuse Cases
  - Each Define:
    - The Abusive Action
    - Source Reference
    - Risk Score (CVSS)
    - Applicable Countermeasure
    - Action to be taken
  - Abuse cases are registered but may be **accepted**
    - **Accepting risk consists in operating with its existence**

Planning and Requirements

Use and Abuse Case



Planning and Requirements

Use and Abuse Case

- **Feature**: Allow user to upload a compressed document along a message

  - **Abuse-01**: Upload Office file with malicious macro in charge of dropping a malware
    - **CVSS**: 6.3
    - **Countermeasure**: Parse the document for Macros
    - **Handling Decision**: Risk accepted

  - **Abuse-02**: Upload a Zip Bomb
    - **CVSS**: 9
    - **Countermeasure**: Scan file with a tool before decompression
    - **Handling Decision**: To be addressed

Planning and Requirements

Security Requirements

- Specific measures to be met to protect data, resources, and users.
  - Derived from applicable laws, industry standards, and the organization's security policies.
  - Security requirements are essential for ensuring the confidentiality, integrity, and availability of information.

- Which security requirements can we set for a service?
  - Will they be **enough**?
  - Will they be **aligned** with current risks?
  - Will they align with the **business requirements** of the application?
  - Will they be aligned with the **quality of competing solutions**?
  - Are they suitable for the **legal/regulatory** environment?
  - Can they be used to **secure the supply chain**?

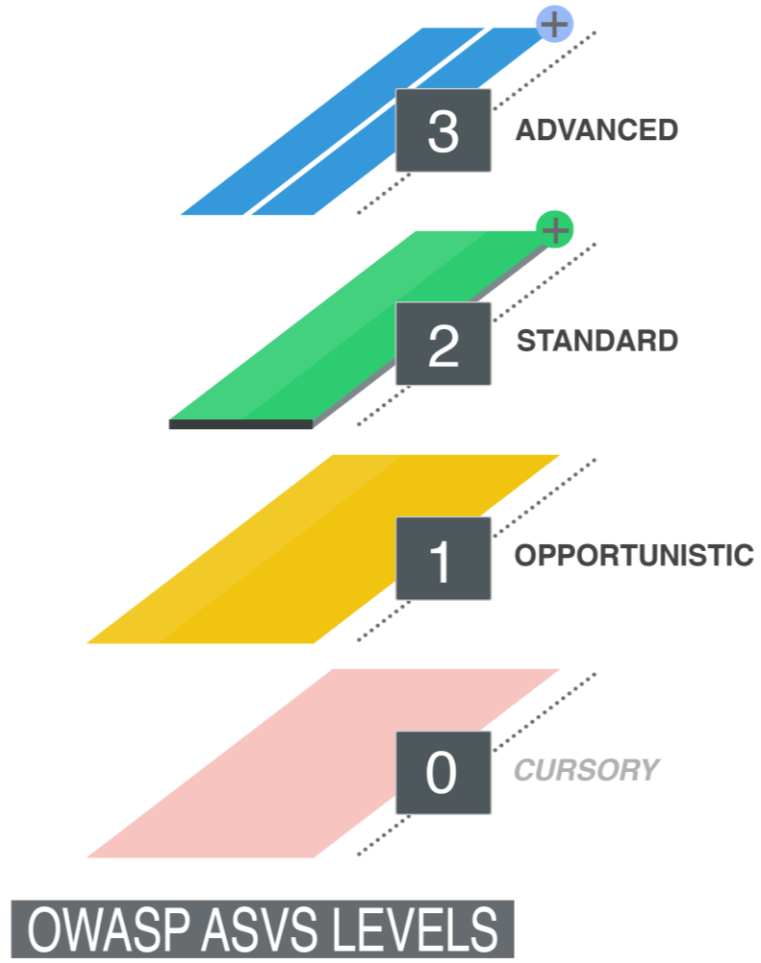
Planning and Requirements

OWASP Application Security Verification Standard

- Level 1: The minimum for any application
  - Completely testable from the outside without documentation
  - Partially testable by SAST and DAST applications
  - Considers the most common vulnerabilities and attacks

- Level 2: The right fit for any application
  - Defined for data-sensitive applications
  - Areas such as B2B transactions, Commerce, Gaming
  - Want to protect application from expert attackers

- Level 3: What is needed for critical applications
  - Defined for applications with very sensitive data
  - Areas such as military environments, healthcare, critical infrastructure



Planning and Requirements

ASVS

V3 Session Management

Identification

Control Objective

One of the core components of any web-based application or stateful API is the mechanism t
controls and maintains the state for a user or device interacting with it. Session management
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 requ

- 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 inact

As previously noted, these requirements have been adapted to be a compliant subset of sele)-
63b controls, focused around common threats and commonly exploited authentication weak ous
verification requirements have been retired, de-duped, or in most cases adapted to be stron th
the intent of mandatory NIST 800-63b requirements.

Description

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	

Section

Requirement

Applicable Levels

References to other sources

Design

Security By Design

- Products are built in a way that reasonably protects against malicious cyber actors successfully gaining access to devices, data, and connected infrastructure

- Some core principles (from CISA):
  - **Take Ownership** of Customer Security Outcomes
  - Embrace Radical **Transparency** and **Accountability**
  - Build Organizational Structure and **Leadership to Support Security Goals**

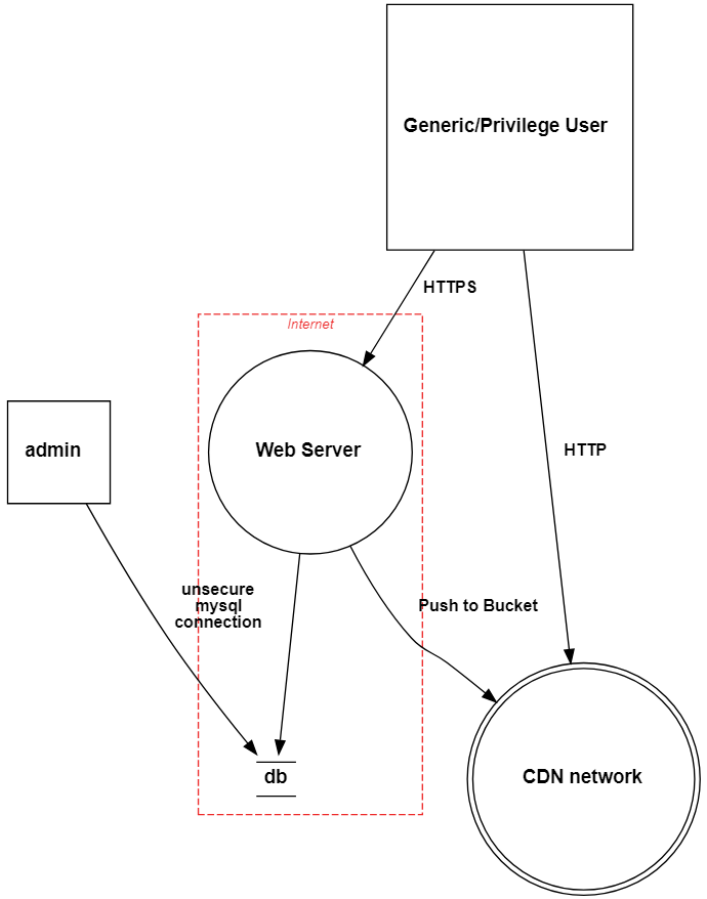
- Some methods:
  - Document Conformance to Secure SDLC Frameworks
  - Implement Vulnerability Management
  - Utilize Open Source Software **Responsibly**
  - Provide Secure Defaults for Developers
  - Foster a Security-Conscious Developer Workforce

Design

Threat Modelling

- To model and analyse technology systems and services to better understand how that system or service might be attacked or otherwise fail.
  - Identify boundaries to sub-domains
  - Identify Interactions
  - Identify potential locations for controls

- Steps:
  - Identify Assets and their relations (Scope)
  - Identify Attacks/Vulnerabilities/What could go wrong
  - Identify counter measures and mitigations
  - Evaluate

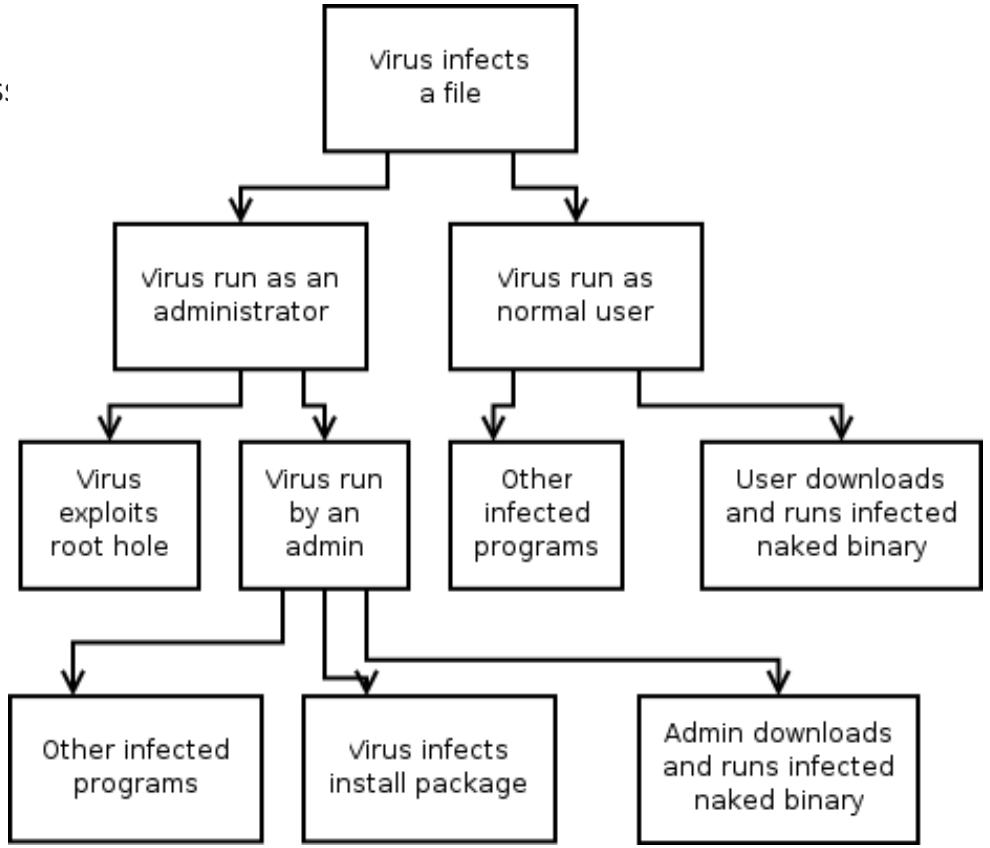


Design

Threat Modelling

- STRIDE model facilitates finding security threats
  - **Spoofing**: Pretending to be something or someone other than yourself
  - **Tampering**: Modifying something on disk, network, memory, or elsewhere
  - **Repudiation**: Claiming that you didn't do something or were not responsible;
  - **Information disclosure**: Someone obtaining information they are not authorized to acces
  - **Denial of service**: Exhausting resources needed to provide service
  - **Elevation of privilege**: Allowing someone to do something they are not authorized to do

- Potential damage of a threat is analyzed using an **Attack Tree**
  - Explores the possible chain of actions exploring threats
  - System Designers can build mitigations
  - Mitigation prevent further exploration along the attack tree



Source: Wikimedia foundation

https://github.com/OWASP/threat-model-cookbook

Design

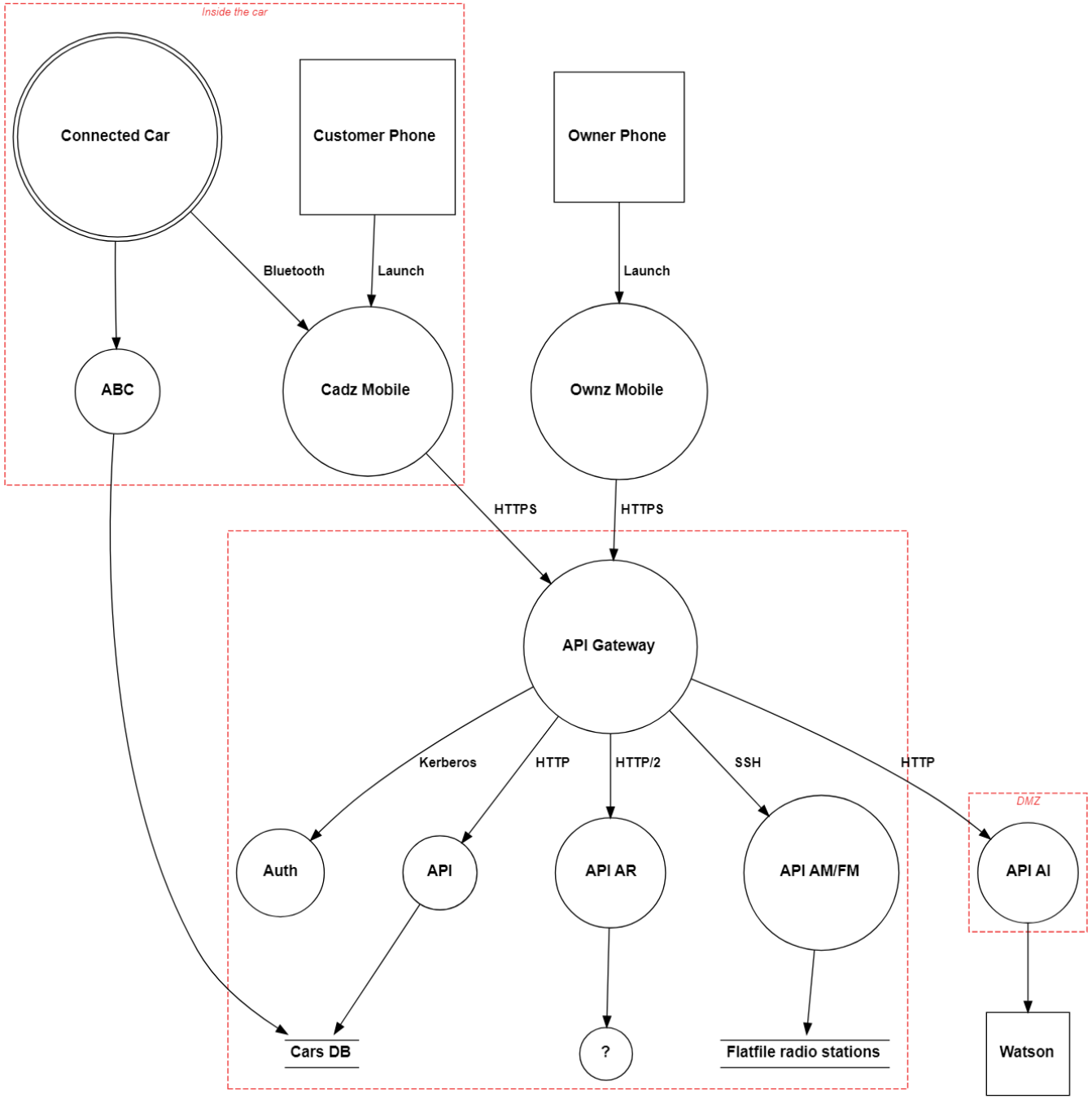
Threat Modelling

A startup ecosystem based on mobile applications and APIs that manage peer to peer car rentals.

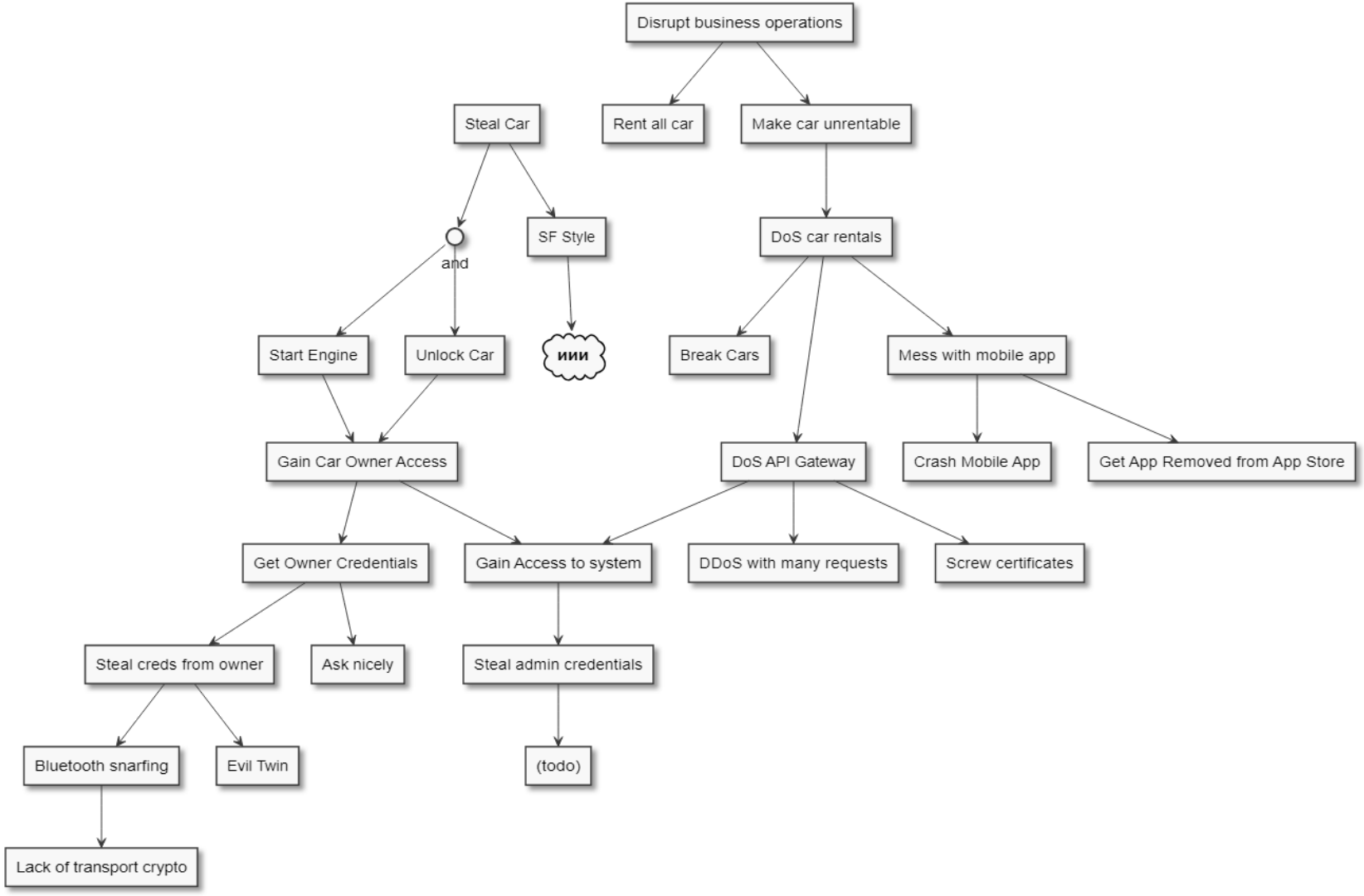
A customer can use a mobile app to unlock and start the car. The owner of the car has its own mobile app to manage rentals.

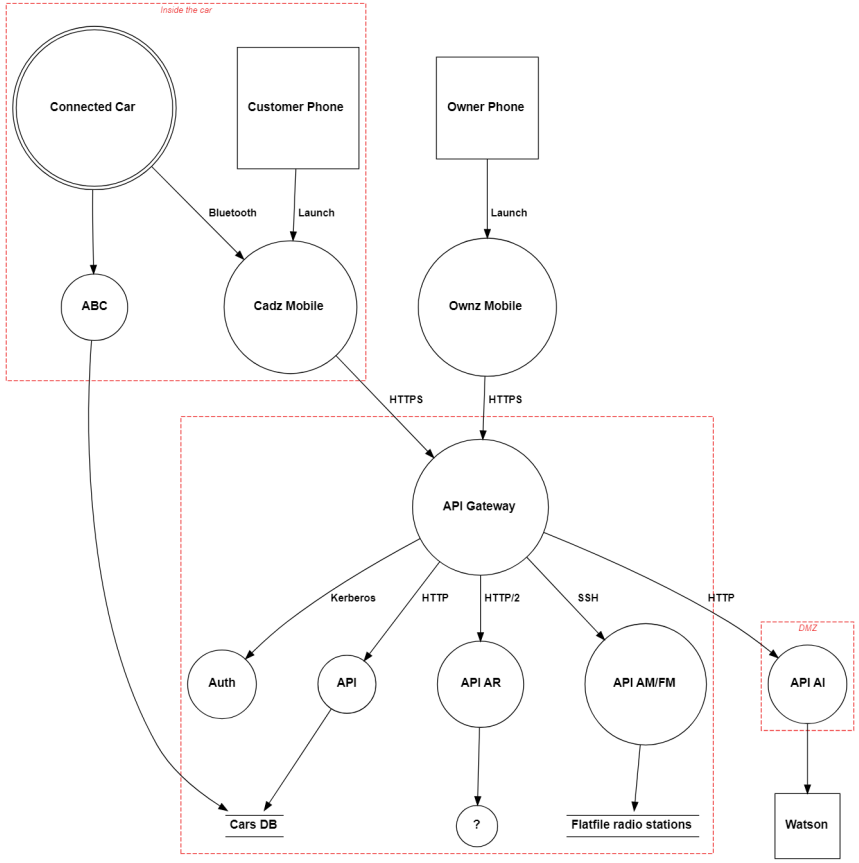
It has AI linked to its APIs and supports augmented reality features.

The APIs also allows to change radio stations which are stored in the cloud on a flat file for legacy reasons.



Design

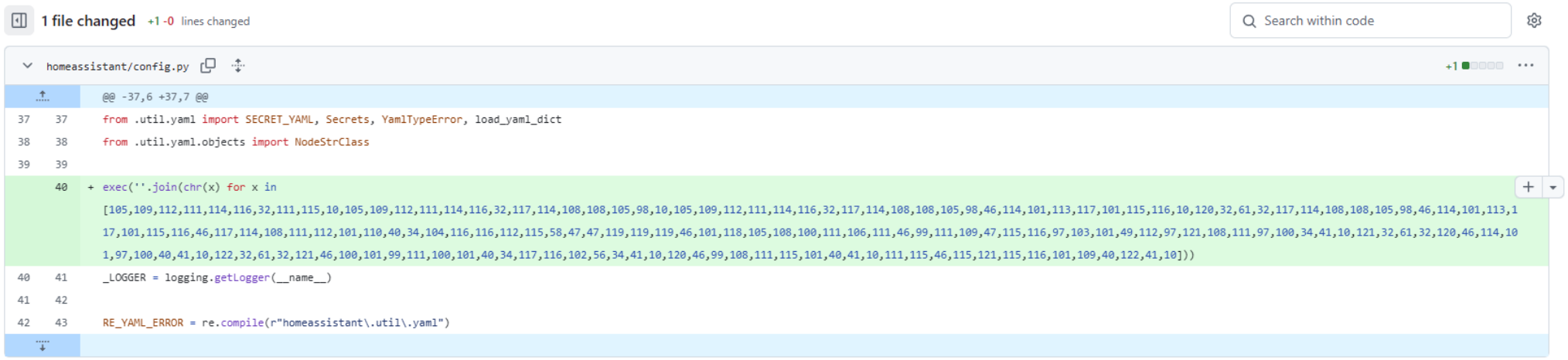

Disrupt business operations
Steal Car
Rent all car
Make car unrentable
and
SF Style
DoS car rentals
Start Engine
Unlock Car
ииии
Break Cars
Mess with mobile app
Gain Car Owner Access
DoS API Gateway
Crash Mobile App
Get App Removed from App Store
Get Owner Credentials
Gain Access to system
DDoS with many requests
Screw certificates
Steal creds from owner
Ask nicely
Steal admin credentials
Bluetooth snarfing
Evil Twin
(todo)
Lack of transport crypto


Inside the car
Connected Car
Customer Phone
Owner Phone
Bluetooth
Launch
Launch
ABC
Cadz Mobile
Ownz Mobile
HTTPS
HTTPS
API Gateway
Kerberos
HTTP
HTTP/2
SSH
HTTP
DMZ
Auth
API
API AR
API AM/FM
API AI
Cars DB
?
Flatfile radio stations
Watson

Development

- Development is made through Versioned systems (e.g. GIT)
  - With strong Access Control in place and Signed Commits
    - Prevent injection of malicious code by a third party
  - Artifacts are extracted from Repository and built automatically
    - Commit hashes can be used to replicate the build process, detecting anomalies
  - Allows facilitated Code Review and Attribution

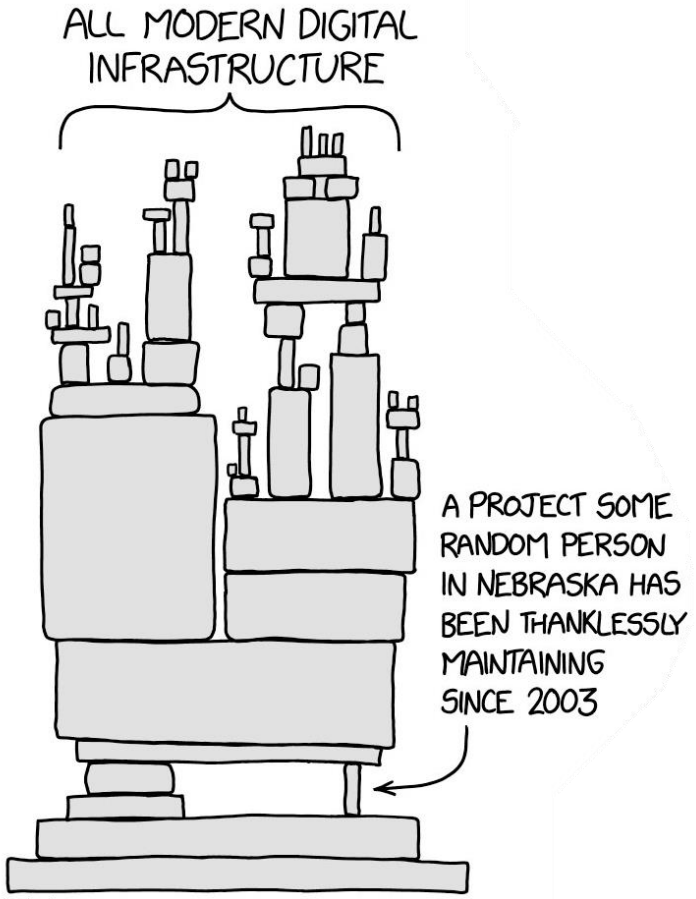
- As collaborative environments, repositories shall not have secrets
  - Passwords and API keys
  - Custom configurations from each developer



Development

Dependency Management

- Constitutes a major issue for a secure SDLC
  - Compromising a dependency is a proven method to subvert software
- Dependencies are easily injected
  - Each bringing both value and risk
  - Frameworks can rapidly inject Tens of libraries
- Dependency tracking and verification is vital
  - Includes applying tests and following the dependency development
  - Software and systems, while not dependencies should also be analyzed
  - Open Source model is especially vulnerable as dependencies are developed by small number of developers



"Further, 94% of projects **had fewer than ten developers** accounting for more than 90% of the LOC added. These findings are counter to the typically held belief that thousands or millions of developers are responsible for developing and maintaining FOSS projects. At a higher level, it was found that **136 developers were responsible for more than 80% of the LOC** added to these 50 FOSS projects", Census II of Free and Open Source Software,

Development

Attacks to dependencies

- **Typo squatting**: Deploys dependencies with names similar to original packages
- **Dependency confusion**: Deploys dependencies with same names as private dependencies
- **Dependency takeover**: Getting ownership of dependency and/or its domain
- **Dependency compromise**: Compromising dependency library or software

Development

SUPPLY CHAIN ATTACK
Attackers insert malicious code into a DLL component of legitimate software. The compromised DLL is distributed to organizations that use the related software.

EXECUTION, PERSISTENCE
When the software starts, the compromised DLL loads, and the inserted malicious code calls the function that contains the backdoor capabilities.

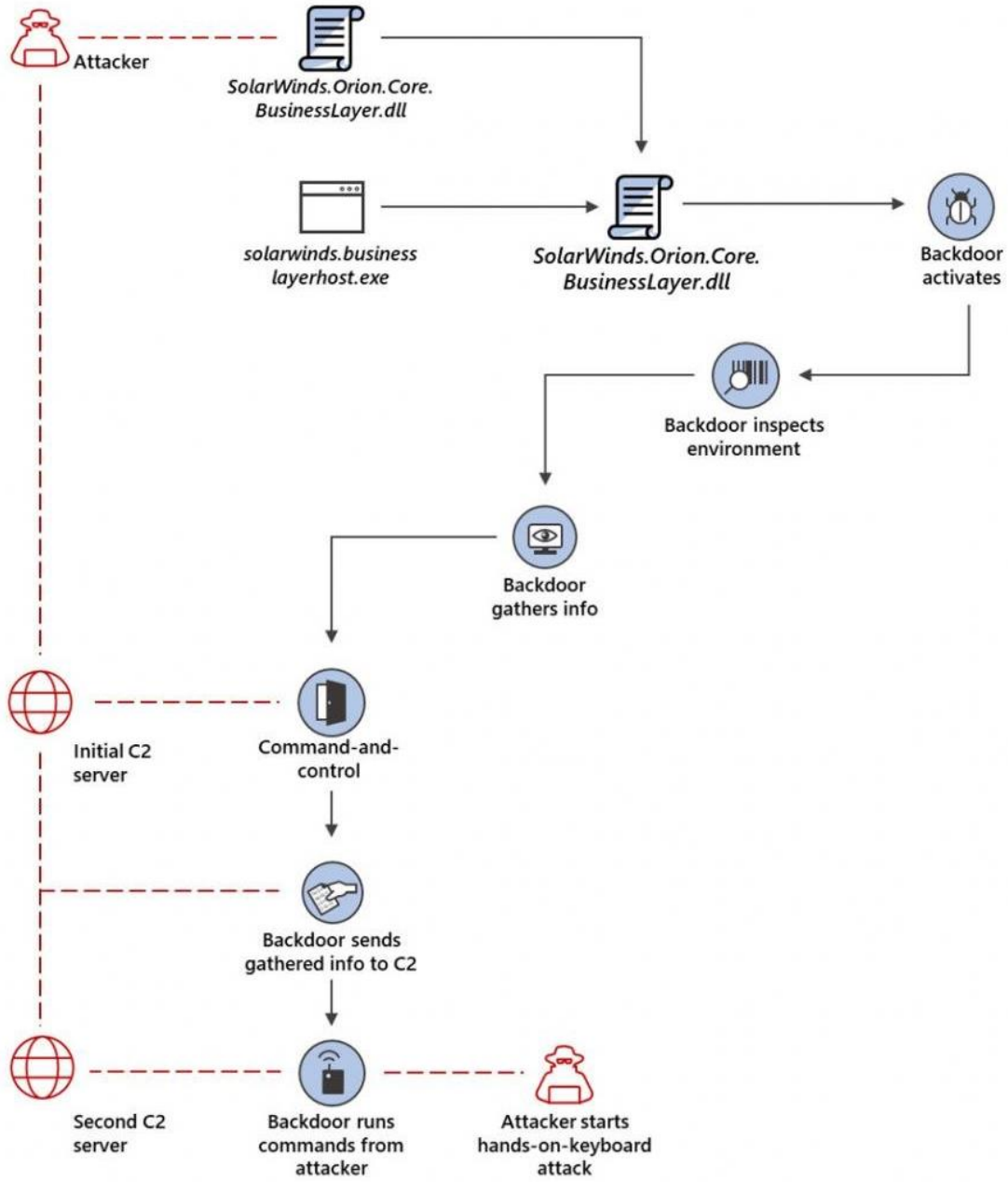
DEFENSE EVASION
The backdoor has a lengthy list of checks to make sure it's running in an actual compromised network.

RECON
The backdoor gathers system info

INITIAL C2
The backdoor connects to a command-and-control server. The domain it connects to is partly based on info gathered from system, making each subdomain unique. The backdoor may receive an additional C2 address to connect to.

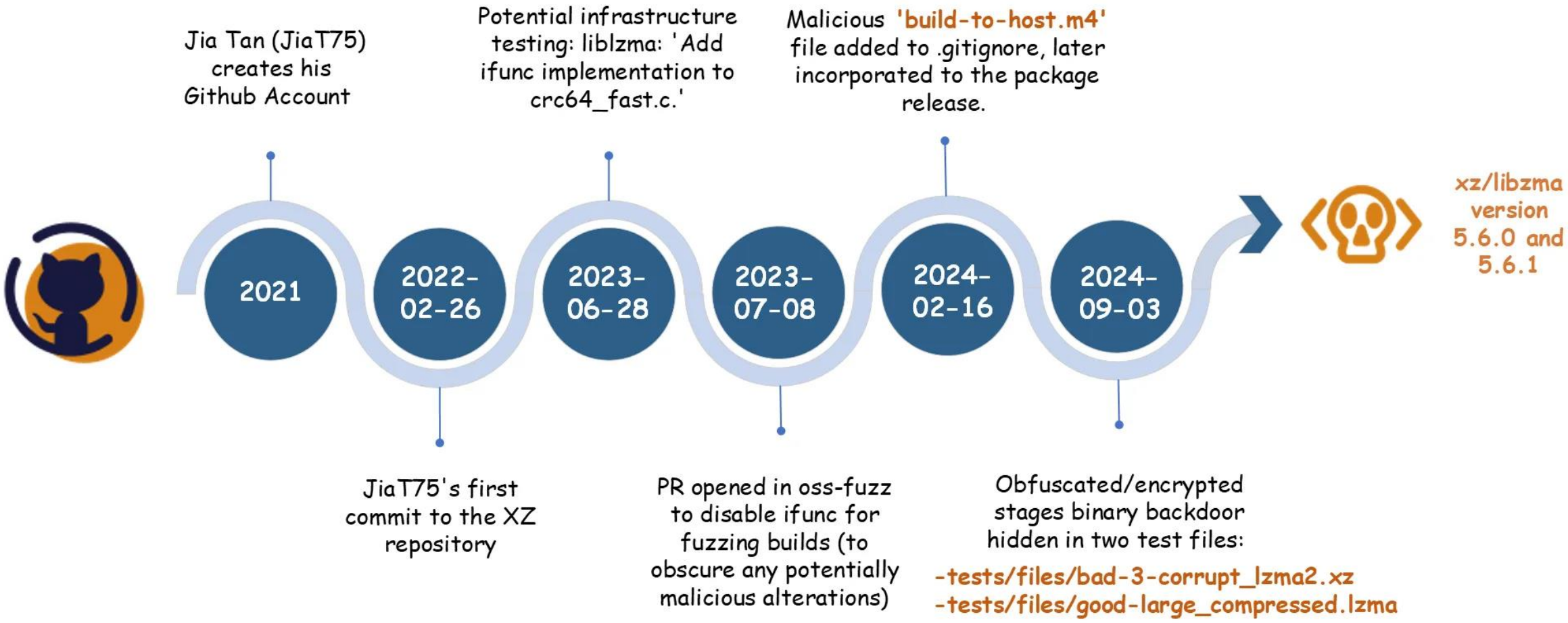
EXFILTRATION
The backdoor sends gathered information to the attacker.

HANDS-ON-KEYBOARD ATTACK
The backdoor runs commands it receives from attackers. The wide range of backdoor capabilities allow attackers to perform additional activities, such as credential theft, progressive privilege escalation, and lateral movement.



Development

XZ Supply Chain Attack



Development

The Polyfill attack

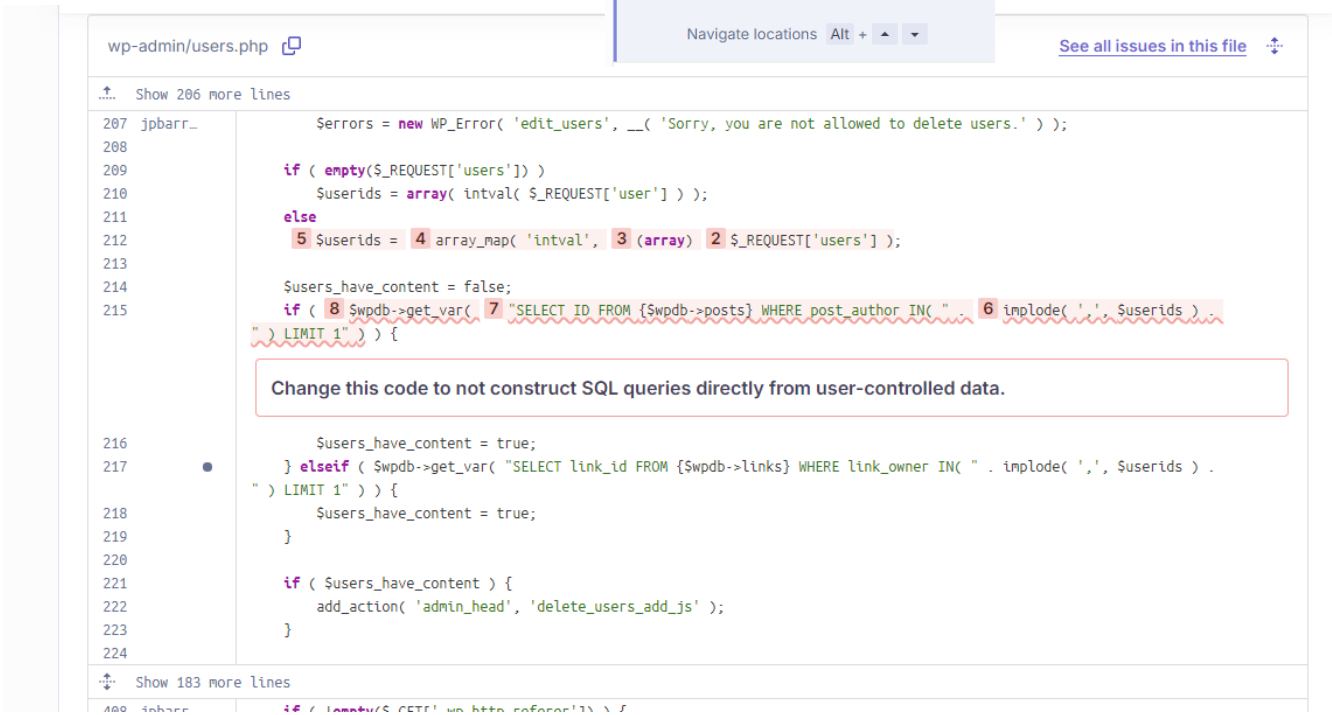
- On June 25, 2024, the Sansec forensics team published a report revealing a **supply chain attack** targeting the widely-used **Polyfill.io** JavaScript library. The attack originated in February 2024 when Funnull, a Chinese company, acquired the previously legitimate Polyfill.io domain and GitHub account. Shortly thereafter, the service began redirecting users to malicious sites and deploying sophisticated malware with advanced evasion techniques.
- On June 27, 2024, Namecheap **suspended the malicious polyfill.io** domain, mitigating the immediate threat for now. However, Censys still detects **384,773 hosts** embedding a polyfill JS script linking to the malicious domain as of July 2,2024, primarily located in Germany.
    - These hosts include websites associated with major platforms like Hulu, Mercedes-Benz, and WarnerBros. Security experts **strongly advise website owners to remove any references to polyfill.io and its associated domains** from their codebase as a precautionary measure. Cloudflare and Fastly have offered alternative, secure endpoints for polyfill services as a workaround.
- **Further investigation has uncovered a more extensive network of potentially compromised domains.** Researchers identified **four additional active domains** linked to the same account that owned the polyfill.io domain. Censys detected **1,637,160 hosts** referencing one or more of these endpoints. At least one of these domains has been observed engaging in malicious activities dating back to June 2023, but the nature of the other associated domains is currently unknown.

Testing

SAST

- Static application security testing
  - Analyses source code, identifying potential anti-patterns
  - Strongly linked with CWEs
  - Frequently included in CI/CD pipelines or IDEs
  - Typically, tools are language specific

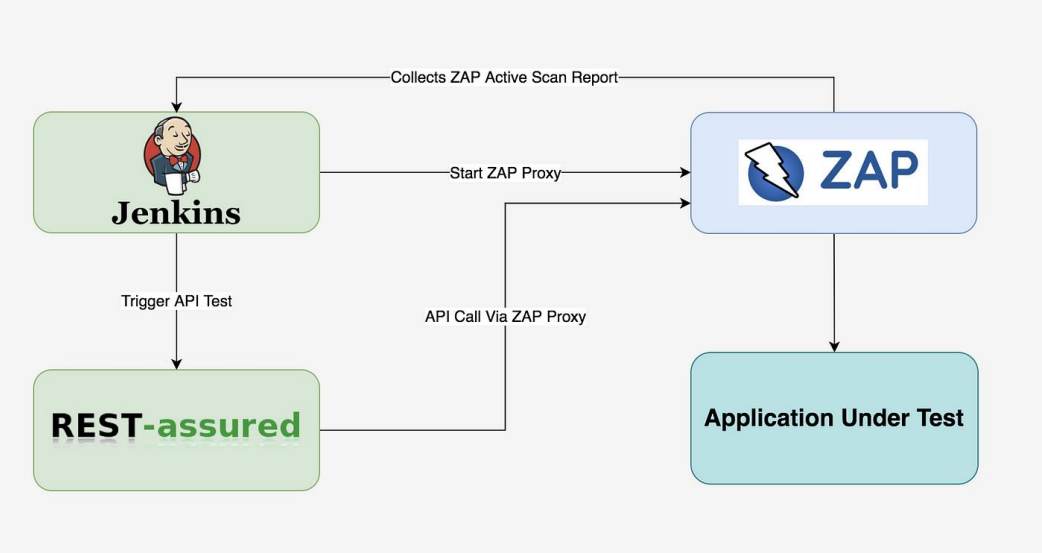
- Other uses:
  - Dependency tracking
  - Secret Detection



Testing

DAST

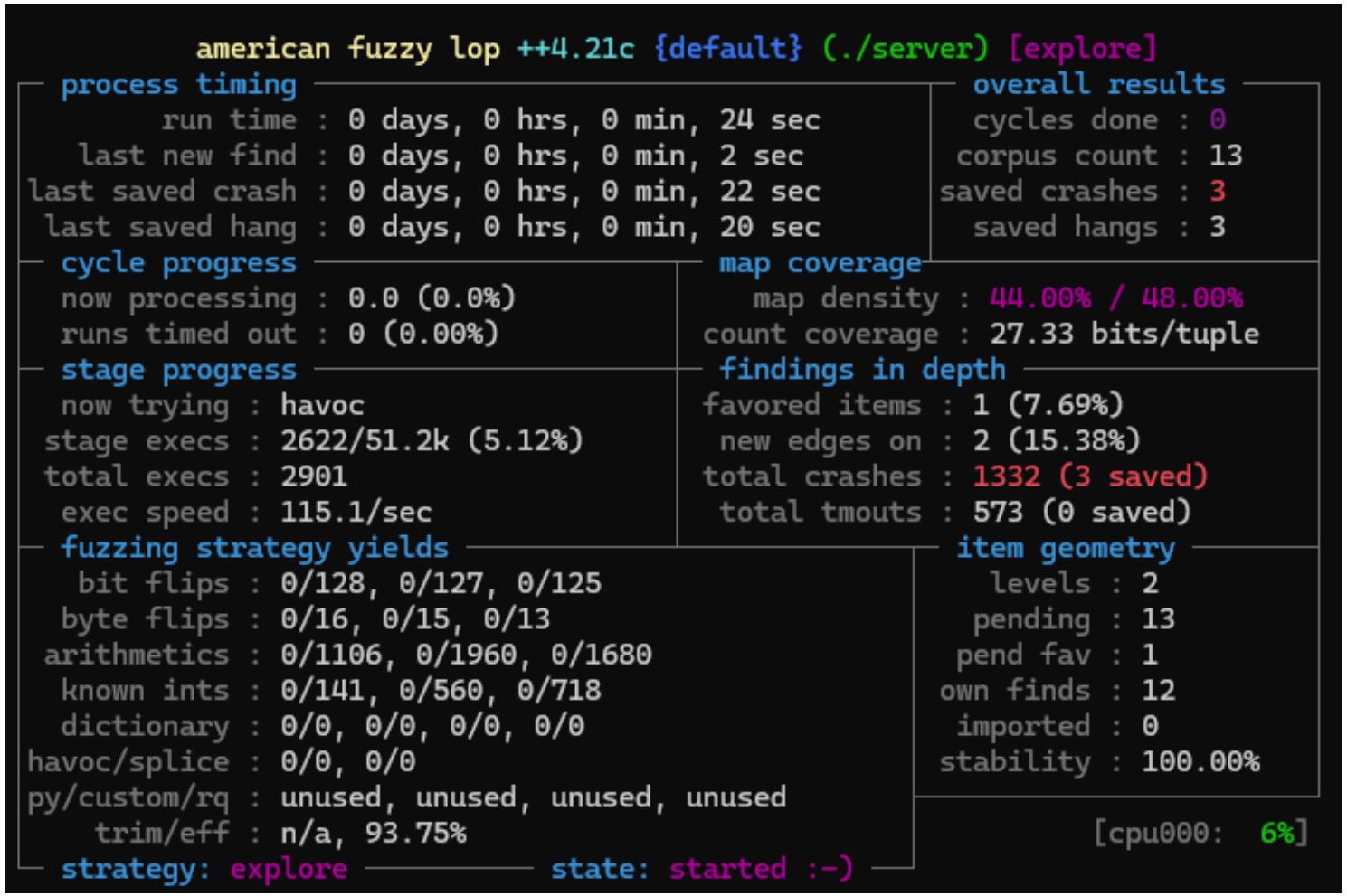
- Dynamic Application Security Testing
  - Analyses app behavior, identifying potential anomalies
  - Strongly linked with behavior analysis and error handling



- Involves active testing with application running
  - With test vectors for known typical vulnerabilities
    - XSS, XXE, SQLI…
  - With fuzzing: **automated software testing technique** that involves providing invalid, unexpected, or random data as inputs
  - Used to software in QA or production
  - Humans and AIs can enhance DAST conducting specialized attacks
    - Replicate Attack Chains or typical exploits

Testing

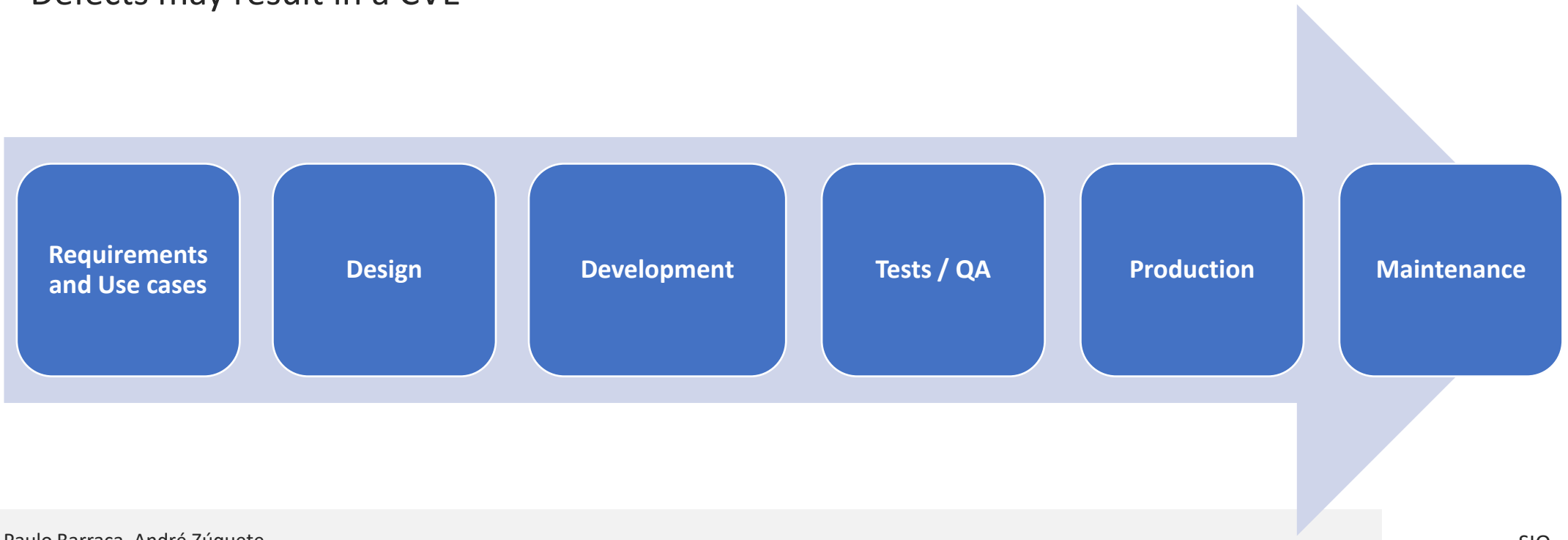
DAST



- AFL++
  - Coverage-based fuzzing
    - Keeps track of what areas of the binary are executing or coverage
    - Applies variation with a genetic algorithm
  - Enables figuring out which inputs lead to which parts of the code executing

Production

- Production considers providing the service from a client facing environment
  - May be internet facing
  - May consider systems outside organization (e.g. public clouds)
  - Constitutes a product, whose actions have relevant impact
  - Defects may result in a CVE



Production

Relevant security mechanisms

- Inventory and asset tracking
  - Enumerate and track assets relevant for service provisioning
  - Includes OS version, update level, **support contracts**, location and ownership

- Configuration hardening
  - Impose a set of configuration guidelines to increase the security
    - Use of passwords vs keys, user permissions, installed packages….
  - Hardening should follow internal policies plus best practices
    - CIS Benchmarks / CIS Controls
      Defense Information Systems Agency (DISA) Security Technical Implementation Guides (STIGs)
    - NIST SP 800-53
    - Payment Card Industry Data Security Standard

Production

Relevant security mechanisms

- Configuration assessment against expected policies
  - OVAL - Open Vulnerability And Assessment Language
  - XCCDF - eXtensible Configuration Checklist Description Format

- Artifact Integrity validation
  - Software signing of binaries produced
  - Validation of artifacts to specific source code repository releases
  - Reverse Engineer to check for additional injects
  - Common Criteria Assessment

```
<definition id="oval:mil.disa.stig.windows:def:177" version="2" class="compliance">
  <metadata>
    <title>BitLocker must be enabled on all fixed drives.</title>
    <affected family="windows">
      <platform>Microsoft Windows 10</platform>
    </affected>
    <description>BitLocker must be enabled on all fixed drives.</description>
  </metadata>
  <criteria operator="AND">
    <criterion test_ref="oval:mil.disa.stig.windows:tst:17700" comment="BitLocker must be enabled on all fixed drives." />
  </criteria>
</definition>
```

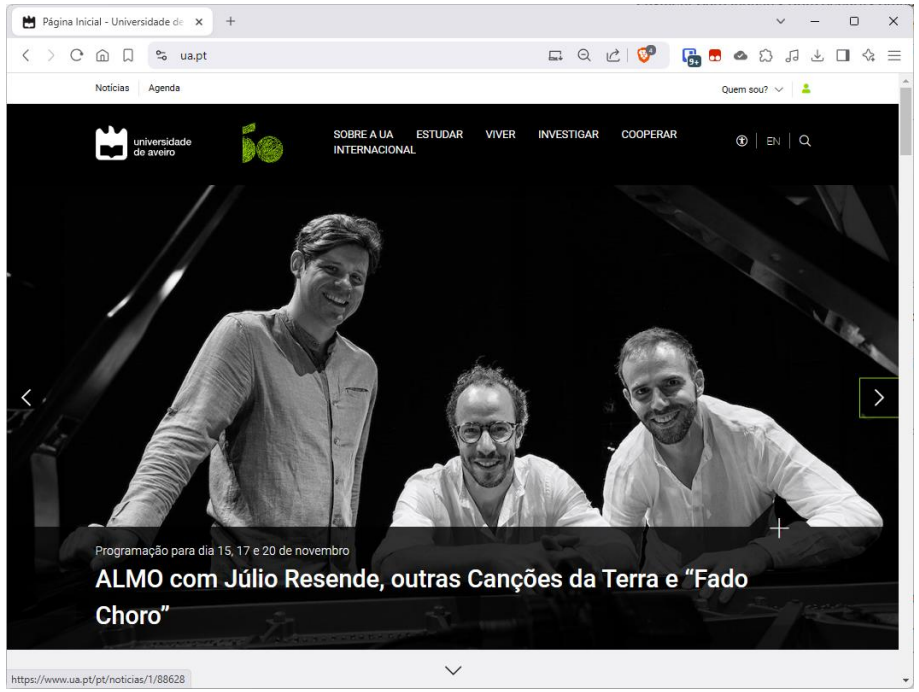
Maintenance

- Process of changing, modifying, and updating software to keep up with customer needs

- Includes
  - Monitoring exposition to internet
  - Deployment of observability capabilities to analyze operation
  - Monitoring features, use cases and abuse cases
  - Product support
  - Incident Response of issues found

- At this stage, issues can result in **security defects**

- Security Issues may have legal and brand impact



Maintenance

- Defects are handled according to a risk based approach
  - E.g. CVSS Based considering Temporal and Environmental factors
  - Scoring allows defining a Service Level Agreement for defects to be handled

	Internal Assets	Interface Assets	External Facing Assets
Emergency	30	10	10
Critical	60	30	10
High	120	60	30
Medium	240	120	60
Low	Not Considered		