

Secure Programming

A.A. 2022/2023

Corso di Laurea in Ingegneria delle Telecomunicazioni

I. Free Security Tools

Paolo Ottolino

Politecnico di Bari

Secure Programming Lab: Course Program

- A. **Intro Secure Programming: «Who-What-Why-When-Where-How»**
- B. **Building Security in: Buffer Overflow, UAF, Command Injection**
- C. **SwA: Weaknesses, Vulnerabilities, Attacks**
- D. **SwA (Software Assurance): Vulnerabilities and Weaknesses (CVE, OWASP, CWE)**
- E. **Security & Protection: Objectives (CIA), Risks (Likelihood, Impact), Rating Methodologies**
- F. **Security & Protection: Security Indicators, BIA, Protection Techniques (AAA, Listing, Duplication etc.)**
- G. **Architecture and Processes: App Infrastructure, Three-Tiers, Cloud, Containers, Orchestration**
- H. **Architecture and Processes 2: Ciclo di Vita del SW (SDLC), DevSecOps (OWASP DSOMM, NIST SSDF)**
- I. **Free Security Tools: OWASP (ZAP, ESAPI, etc), NIST (SAMATE, SARD etc.)**
- J. **Dynamic Security Test: VA, PT, DAST (cfr. VulnScanTools), WebApp Sec Scan Framework (Arachni, SCNR) :**
- K. **Operating Environment: Kali Linux on WSL**
- L. **Python: Powerful Language for easy creation of hacking tools**
- M. **Exercises: SecureFlag**



H. Free Security Tools

Agenda

I.1 Recap & Shatters

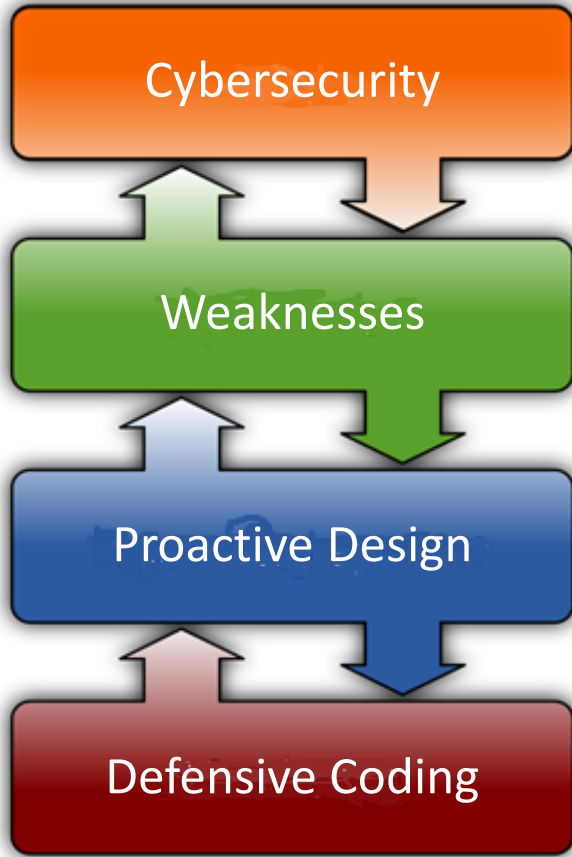
I.2 Shatters drill down

I.3 Tools



I.1 Free Security Tools: Recap

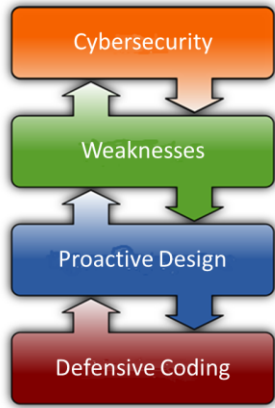
Secure Programming Arguments



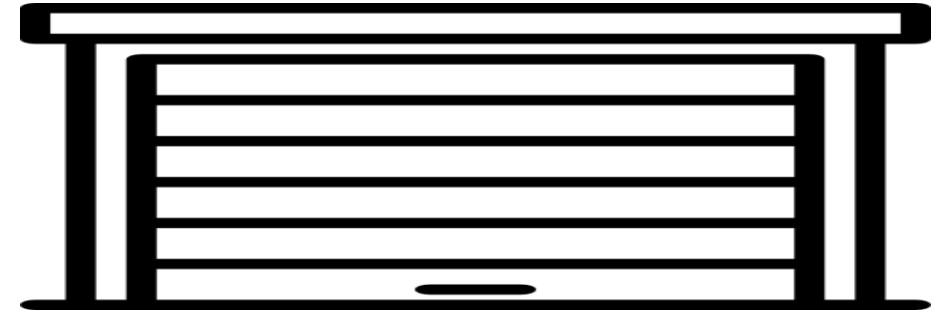
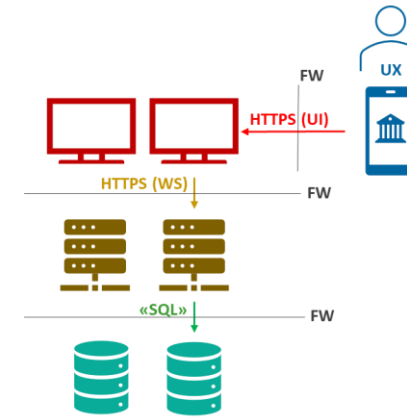
Goals	Techniques	Measures	Approaches	Abstractions
CIA (Attacker Profiles)	Protection Criteria: <ul style="list-style-type: none"> Filtering Hiding Logging 	Risk Rating <ul style="list-style-type: none"> SLExARO (likelihood) BIA Framework Checklist 	Risk Remediation <ul style="list-style-type: none"> Avoid Transfer Mitigate Accepts 	Indicators <ul style="list-style-type: none"> KPI KGI SLA etc.
Attack Lifecycle: <ul style="list-style-type: none"> Cyber Kill Chain MITRE ATT&CK 	Vulnerability <ul style="list-style-type: none"> Lifecycle Security Bulletins 	CVE / CVSS	OWASP Top10	CWE RFC 4949 (Glossary)
Containers (Orchestration)	CSMA ZTA Pillars	Responsibility Sharing <ul style="list-style-type: none"> IaaS PaaS SaaS 	DevOps SecDevOps	SDO Maturity Model
Secure Coding Practices	Code Bugs <ul style="list-style-type: none"> BOF UAF Uncontrolled Input 	Bugs & Exploits	Input Validation <ul style="list-style-type: none"> Checking Whitelist Sanitizing Escape Checking Blacklist Sanitizing Blacklist 	Shift Left

I.1a Free Security Tools: Recap

from Arguments to Shutters

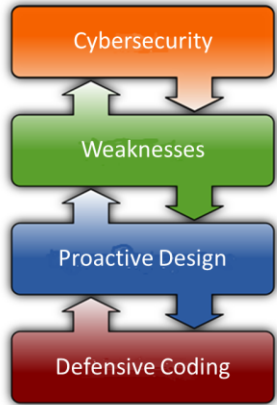


Goals	Techniques	Measures	Approaches	Abstractions
CIA (Attacker Profiles)	Protection Criteria: <ul style="list-style-type: none"> Filtering Hiding Logging 	Risk Rating <ul style="list-style-type: none"> SLExARO (likelihood) BIA Framework Checklist 	Risk Remediation <ul style="list-style-type: none"> Avoid Transfer Mitigate Accepts 	Indicators <ul style="list-style-type: none"> KPI KGI SLA etc.
Attack Lifecycle: <ul style="list-style-type: none"> Cyber Kill Chain MITRE ATT&CK 	Vulnerability <ul style="list-style-type: none"> Lifecycle Security Bulletins 	CVE / CVSS	OWASP Top10	CWE RFC 4949 (Glossary)
Containers (Orchestration)	CSMA ZTA Pillars	Responsibility Sharing <ul style="list-style-type: none"> IaaS PaaS SaaS 	DevOps SecDevOps	SDO Maturity Model
Secure Coding Practices	Code Bugs <ul style="list-style-type: none"> BOF UAF Uncontrolled Input 	Bugs & Exploits	Input Validation <ul style="list-style-type: none"> Checking Whitelist Sanitizing Escape Checking Blacklist Sanitizing Blacklist 	Shift Left

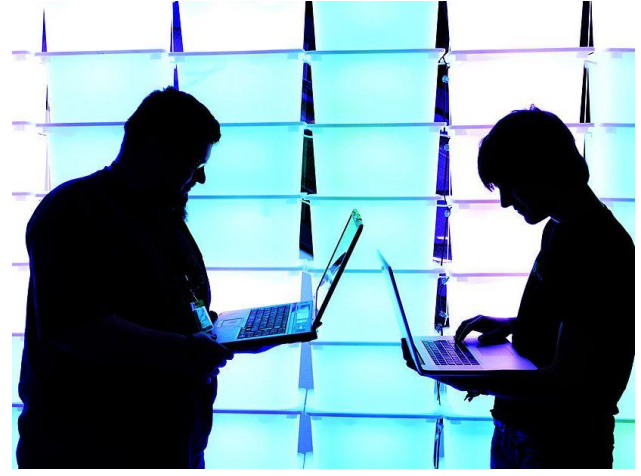
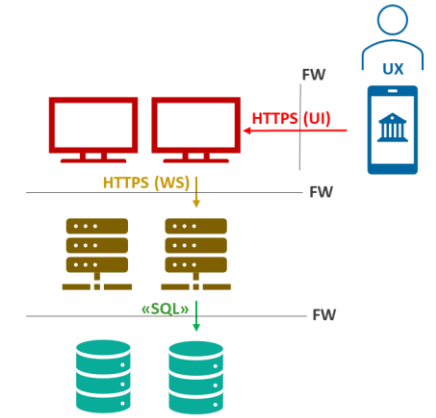


I.1b Free Security Tools: Recap

from Arguments to Shutters

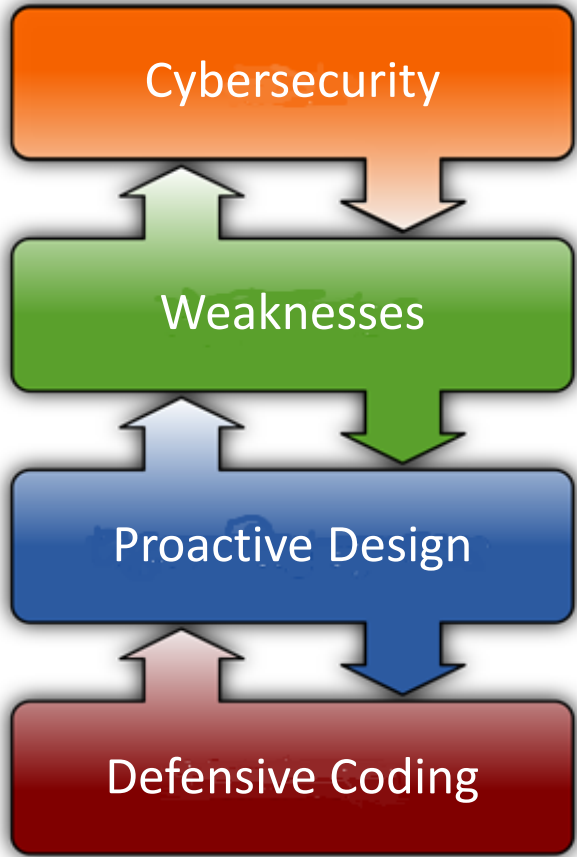


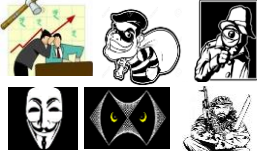



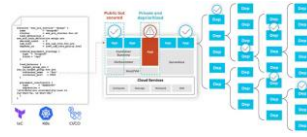
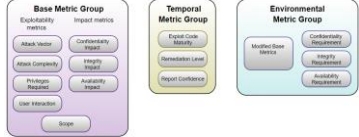


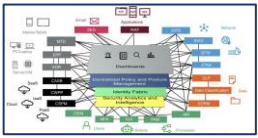


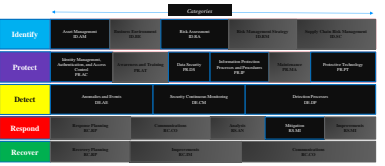

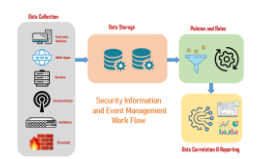
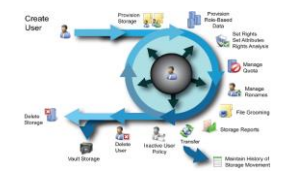
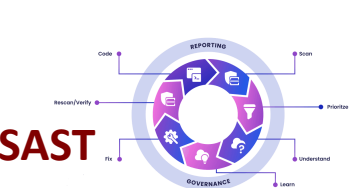




Goals	Techniques	Measures	Approaches	Abstractions
CIA (Attacker Profiles)	Protection Criteria: <ul style="list-style-type: none"> Filtering Hiding Logging 	Risk Rating <ul style="list-style-type: none"> SLExARO (likelihood) BIA Framework Checklist 	Risk Remediation <ul style="list-style-type: none"> Avoid Transfer Mitigate Accepts 	Indicators <ul style="list-style-type: none"> KPI KGI SLA etc.
Attack Lifecycle: <ul style="list-style-type: none"> Cyber Kill Chain MITRE ATT&CK 	Vulnerability <ul style="list-style-type: none"> Lifecycle Security Bulletins 	CVE / CVSS	OWASP Top10	CWE RFC 4949 (Glossary)
Containers (Orchestration)	CSMA ZTA Pillars	Responsibility Sharing <ul style="list-style-type: none"> IaaS PaaS SaaS 	DevOps SecDevOps	SDO Maturity Model
Secure Coding Practices	Code Bugs <ul style="list-style-type: none"> BOF UAF Uncontrolled Input 	Bugs & Exploits	Input Validation <ul style="list-style-type: none"> Checking Whitelist Sanitizing Escape Checking Blacklist Sanitizing Blacklist 	Shift Left



I.1c Free Security Tools: Shutters

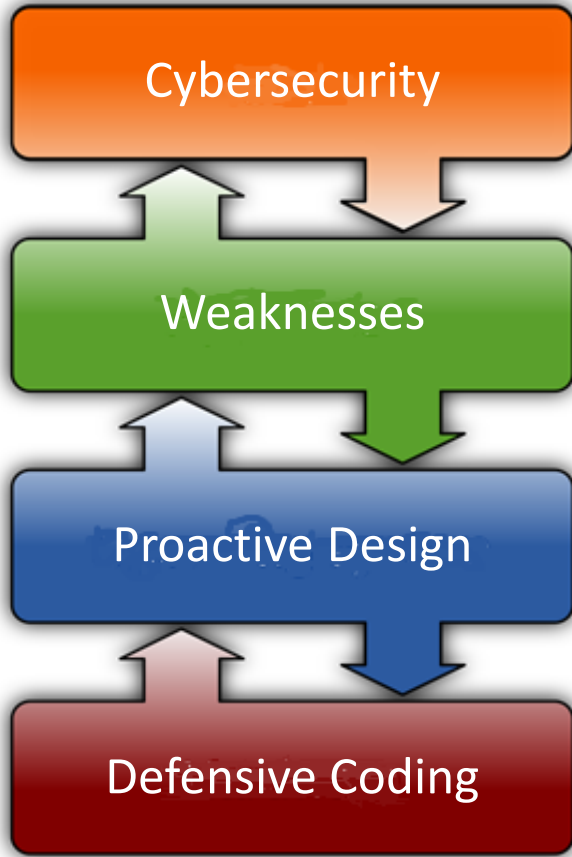
Secure Programming Shutters



Know	Reduce	Evaluate	Execute	Process																																					
 <p>Attackers</p>	 <p>Protection</p>	<table border="1"> <thead> <tr> <th colspan="4">Overall Risk Severity</th> </tr> <tr> <th rowspan="3">Impact</th> <th>HIGH</th> <th>Medium</th> <th>High</th> <th>Critical</th> </tr> </thead> <tbody> <tr> <td>MEDIUM</td> <td>Low</td> <td>Medium</td> <td>High</td> </tr> <tr> <td>LOW</td> <td>Note</td> <td>Low</td> <td>Medium</td> </tr> <tr> <td></td> <td>LOW</td> <td>MEDIUM</td> <td>HIGH</td> <td></td> </tr> <tr> <td></td> <td colspan="3">Likelihood</td> <td></td> </tr> </tbody> </table> <p>Risk Rating</p>	Overall Risk Severity				Impact	HIGH	Medium	High	Critical	MEDIUM	Low	Medium	High	LOW	Note	Low	Medium		LOW	MEDIUM	HIGH			Likelihood				<table border="1"> <thead> <tr> <th>Risk Option</th> <th>Default Option</th> </tr> </thead> <tbody> <tr> <td>Avoid avoid the activity that creates the risk</td> <td>Avoid The O.S. decides what resource request to accept, on the basis of the knowledge of which resources each process will use, when requested.</td> </tr> <tr> <td>Transfer transfer the risk you take to another party</td> <td>Detect and Recover detect the occurring conditions and acting for removing one or more of them.</td> </tr> <tr> <td>Reduce security actions for reducing the vulnerabilities</td> <td>Prevent Prevent all four necessary conditions from occurring. They describe how resources are to be requested.</td> </tr> <tr> <td>Accept no action at all (or reduced one)</td> <td>Ignore Justified by the low occurrence of the deadlock. Requires manual user intervention.</td> </tr> </tbody> </table> <p>Risk Mgmt</p>	Risk Option	Default Option	Avoid avoid the activity that creates the risk	Avoid The O.S. decides what resource request to accept, on the basis of the knowledge of which resources each process will use, when requested.	Transfer transfer the risk you take to another party	Detect and Recover detect the occurring conditions and acting for removing one or more of them.	Reduce security actions for reducing the vulnerabilities	Prevent Prevent all four necessary conditions from occurring. They describe how resources are to be requested.	Accept no action at all (or reduced one)	Ignore Justified by the low occurrence of the deadlock. Requires manual user intervention.	<p>Monitor Plan</p> 
Overall Risk Severity																																									
Impact	HIGH	Medium	High	Critical																																					
	MEDIUM	Low	Medium	High																																					
	LOW	Note	Low	Medium																																					
	LOW	MEDIUM	HIGH																																						
	Likelihood																																								
Risk Option	Default Option																																								
Avoid avoid the activity that creates the risk	Avoid The O.S. decides what resource request to accept, on the basis of the knowledge of which resources each process will use, when requested.																																								
Transfer transfer the risk you take to another party	Detect and Recover detect the occurring conditions and acting for removing one or more of them.																																								
Reduce security actions for reducing the vulnerabilities	Prevent Prevent all four necessary conditions from occurring. They describe how resources are to be requested.																																								
Accept no action at all (or reduced one)	Ignore Justified by the low occurrence of the deadlock. Requires manual user intervention.																																								
 <p>CVE</p>	 <p>SCA</p>	 <p>CVSS</p>	 <p>Vulnerability Mgmt</p>	<p>Test Release</p> 																																					
 <p>CSMA</p>	 <p>ZTA (Pillars)</p>	 <p>DAST</p>	 <p>Architecture Mgmt</p>	<p>Deploy Operate</p> 																																					
 <p>(Audit) Log</p>	 <p>Access Control</p>	 <p>SAST</p>	<table border="1"> <thead> <tr> <th>Option</th> <th></th> <th></th> </tr> </thead> <tbody> <tr> <td>Avoid avoid the activity that creates the risk</td> <td>Checking Whitelisting reject strings that seems invalid (safer than fix it).</td> <td></td> </tr> <tr> <td>Transfer transfer the risk you take to another party</td> <td>Sanitization Escaping Replace problematic characters with safe ones</td> <td></td> </tr> <tr> <td>Reduce security actions for reducing the vulnerabilities</td> <td>Checking Blacklisting Reject strings with possibly bad char.</td> <td></td> </tr> <tr> <td>Accept no action at all (or reduced one)</td> <td>Sanitization Blacklisting Delete the characters you don't want</td> <td></td> </tr> </tbody> </table> <p>Input Mgmt</p>	Option			Avoid avoid the activity that creates the risk	Checking Whitelisting reject strings that seems invalid (safer than fix it).		Transfer transfer the risk you take to another party	Sanitization Escaping Replace problematic characters with safe ones		Reduce security actions for reducing the vulnerabilities	Checking Blacklisting Reject strings with possibly bad char.		Accept no action at all (or reduced one)	Sanitization Blacklisting Delete the characters you don't want		<p>Code Build</p> 																						
Option																																									
Avoid avoid the activity that creates the risk	Checking Whitelisting reject strings that seems invalid (safer than fix it).																																								
Transfer transfer the risk you take to another party	Sanitization Escaping Replace problematic characters with safe ones																																								
Reduce security actions for reducing the vulnerabilities	Checking Blacklisting Reject strings with possibly bad char.																																								
Accept no action at all (or reduced one)	Sanitization Blacklisting Delete the characters you don't want																																								
<p>SOAR</p> 	<p>IGA</p> 	<p>KPI</p> 																																							

I.1d Free Security Tools: Shutters

Secure Programming Shutters

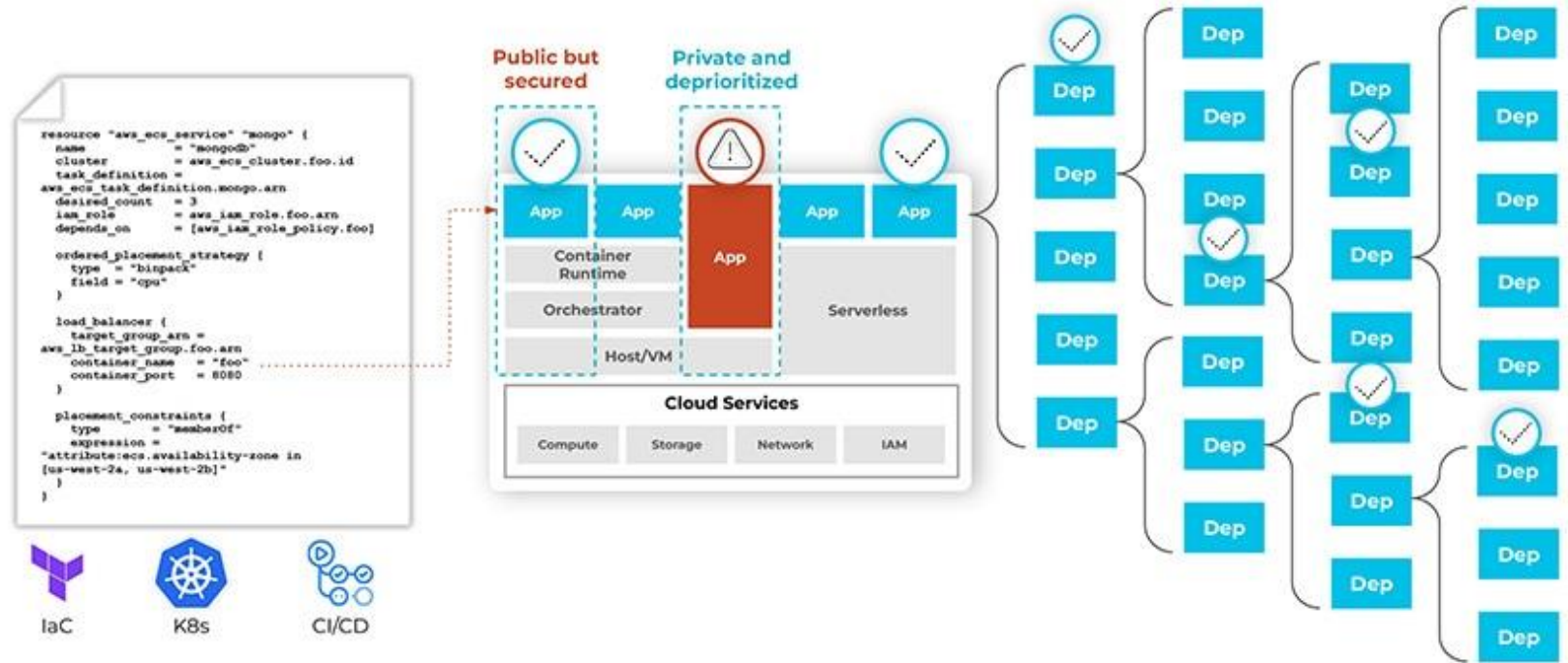


Know	Reduce	Evaluate	Execute	Process
Attackers <ul style="list-style-type: none"> • Profile • Trends • Motive • Opportunity • Means 	Protection <ul style="list-style-type: none"> • AAA • Duplicate • Filter • Log • Encode 	Risk Rating <ul style="list-style-type: none"> • Likelihood • Impact • Level 	Risk Mgmt <ul style="list-style-type: none"> • Avoid • Transfer • Mitigate • Accepts 	Monitor Plan
CVE <ul style="list-style-type: none"> • Description • Severity • References • Weaknesses • Configuration 	SCA <ul style="list-style-type: none"> • Identify • Dependences • Vulnerability • OSInt, CLOSInt • Speed 	CVSS <ul style="list-style-type: none"> • Exploitability • Impact • Scope 	Vulnerability Mgmt <ul style="list-style-type: none"> • Substitute • Virtual Patch • Patch • Ignore/Postpone 	Test Release
CSMA <ul style="list-style-type: none"> • Users • Cloud/On-premise • Network • Application • Data 	ZTA (Pillars) <ul style="list-style-type: none"> • Identity • Endpoint • Network • Workload • Data 	DAST <ul style="list-style-type: none"> • Explore • Test • Evaluate 	Architecture Mgmt <ul style="list-style-type: none"> • WAF • Supplier • Implement • Ignore/Postpone 	Deploy Operate
(Audit) Log <ul style="list-style-type: none"> • Date, Time • User, Device • Net Addr, Prot • Location • Event/Activity 	Access Control <ul style="list-style-type: none"> • Identify • AuthN • AuthZ • Govern (Certify) • Monitor 	SAST <ul style="list-style-type: none"> • Scan • Prioritize • Verify 	Input Mgmt <ul style="list-style-type: none"> • Checking Whitelist • Sanitizing Escape • Checking Blacklist • Sanitizing Blacklist 	Code Build
SOAR	IGA	KPI		

1.2b Drill Down: Software Composition Analysis

5 SCA challenges

1. Obscured visibility
2. Understanding the dependency logic
3. Drowning in vulnerabilities
4. vulnerability database
5. Speed



<https://snyk.io/series/open-source-security/software-composition-analysis-sca/>

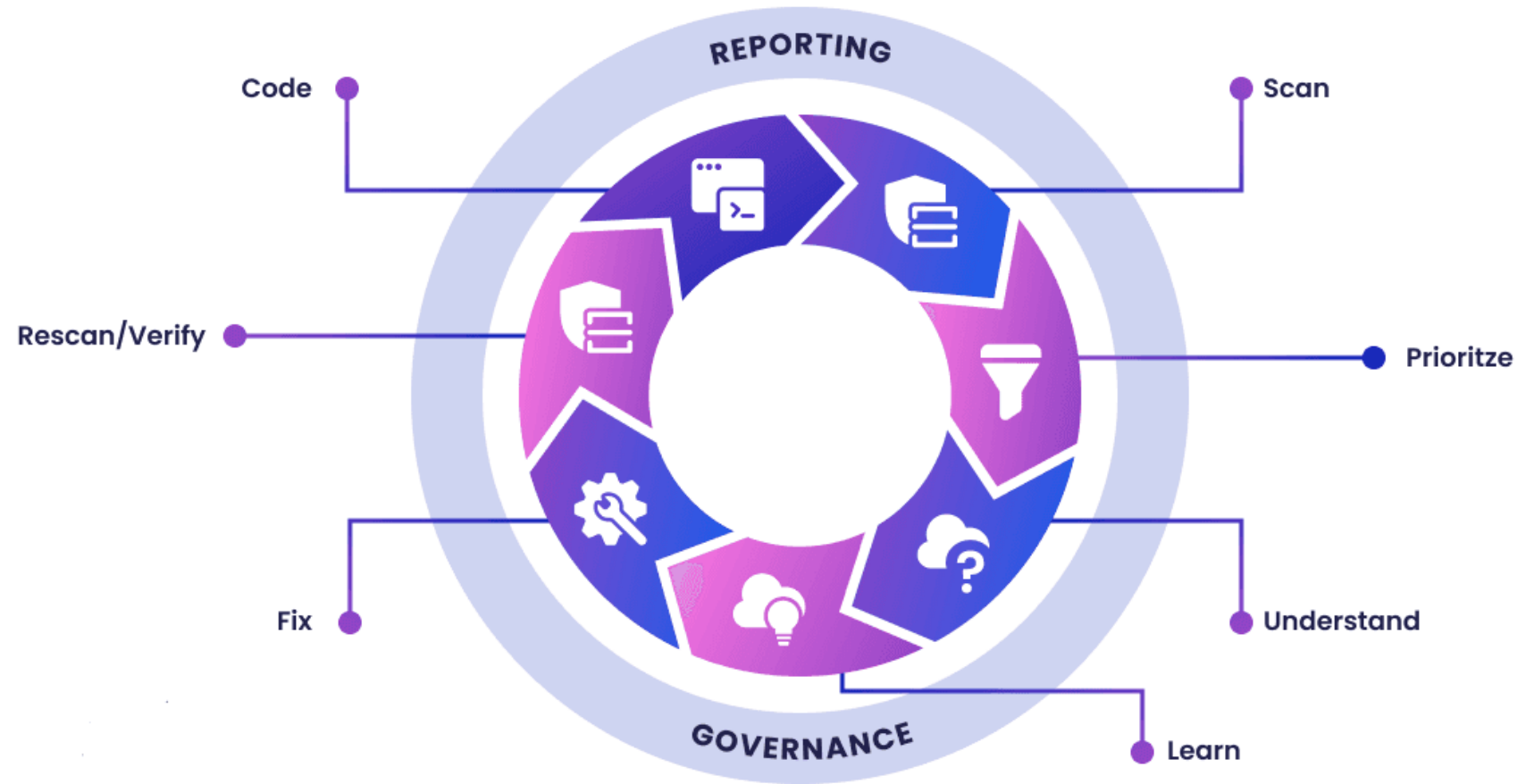
<https://www.paloaltonetworks.com/cyberpedia/what-is-sca>



1.2c Drill Down: Static Application Security Testing

7 Stages of SAST

1. (Code)
2. Scan
3. Prioritize
4. (Understand)
5. (Learn)
6. (Fix)
7. Rescan/Verify

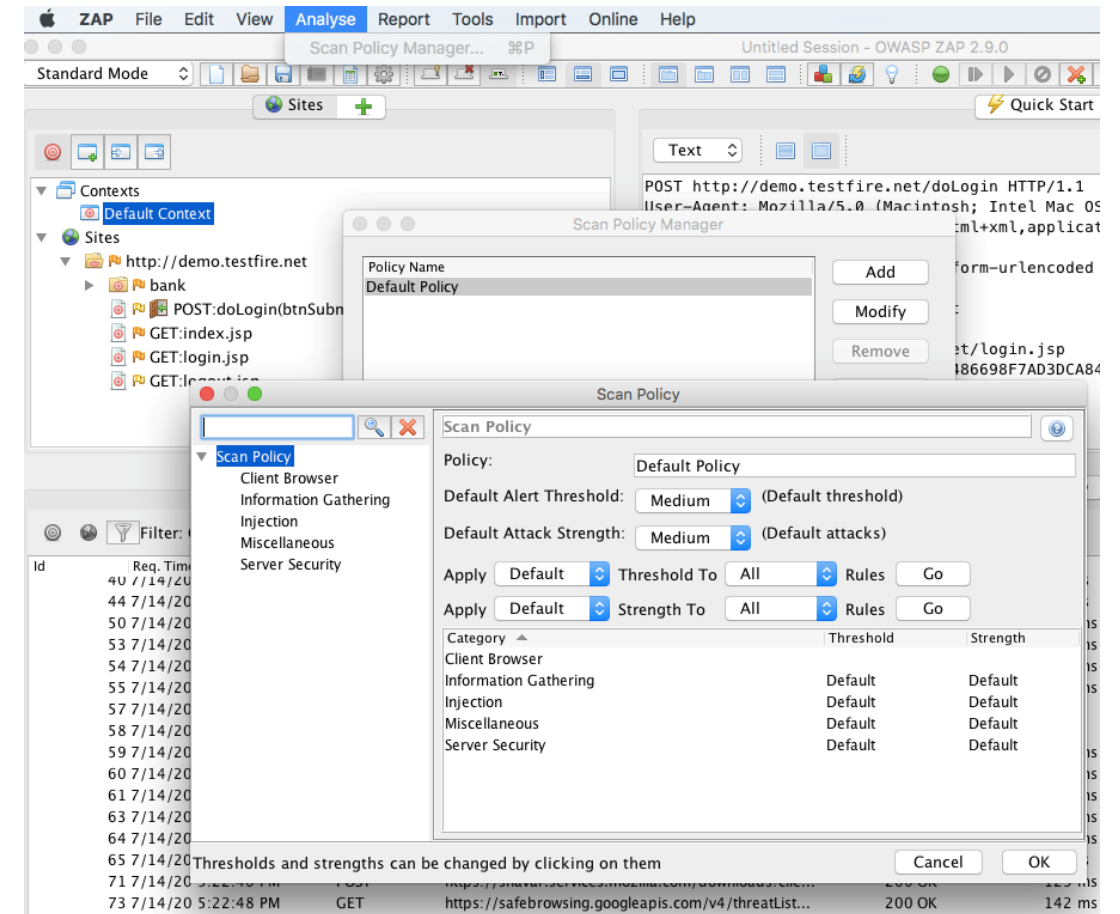


<https://snyk.io/learn/application-security/static-application-security-testing/>

1.2d Drill Down: Dynamic Application Security Testing

Using Zed Attack Proxy

- 1. Passive Scan:** the intended applications under assessment are being **intercepted** by the tool & those requests / responses are observed by the tool to flag security misconfigurations such as missing security headers or cookie settings. The tool doesn't send any new requests on its own in this phase, it just analyses the intercepted requests / responses.
- 2. Active Scan:** the intended applications under assessment are attacked by the tool by **sending new requests** with malicious payloads to discover security violations. The tool flags the violations based on its behaviour / received responses from the server after injecting malicious payloads. These payloads are introduced by the tool after we complete intercepting the application journeys that we want to test so they act as a baseline for the tool to start sending new requests with new payloads.

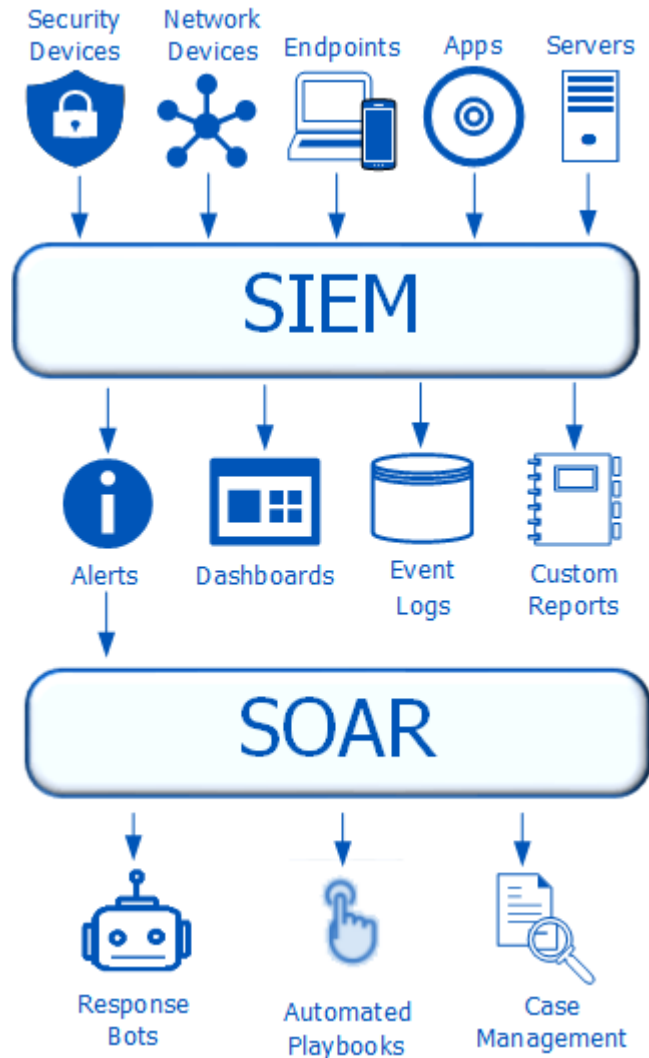


<https://gotowebsecurity.com/dynamic-application-security-testing-dast-using-owasp-zap-v2-9-0/>



I.2e Drill Down: Security Orchestration and Automation

SIEM and SOAR



SIEM and SOAR work together, enabling you to detect, investigate, and respond quickly and confidently to critical cybersecurity threats across your organization:

- Unifying threat & telemetry data across disparate sources
- Identifying event and alert trends
- Prioritizing alerts to minimize false positives
- Simplifying compliance and reporting obligations
- Building playbooks that orchestrate the critical tools you rely on
- Rapidly assessing scenarios and quantifying their impact on your organization
- Streamlining incident response through a single, customizable interface
- Automating routine and repeatable incident response tasks and workflows

I.3a OWASP

Tools

OWASP Tools

- OWASP [Amass](#) is a [penetration testing](#) tool for mapping the target application's attack surface.
- The OWASP [Zed Attack Proxy \(ZAP\)](#) is a useful tool for testing web applications, comparable to widely-used penetration testing proxies such as Burp or Fiddler.
- OWASP [WebGoat](#) (Java), [Security Shepherd](#) (Java/Android) and OWASP [Juice Shop](#) (Node.js) are intentionally vulnerable applications to help practice your application security skills.
- OWASP SKF Write-Ups: <https://owasp-skf.gitbook.io/asvs-write-ups/>
- [Dependency-Check](#) and [Dependency-Track](#) allow automated detection of vulnerable project dependencies in a number of programming languages and build systems, with CI/CD integration.

OWASP Code

- The OWASP [CSRFGuard](#) protects against [Cross-Site Request Forgery](#) attacks for Java web apps.
- The OWASP [ModSecurity Core Rule Set](#) is a set of generic attack detection rules to be used with web application firewalls to protect against many common attacks



1.3b OWASP

Labs

OWASP Security Shepherd: <https://owasp.org/www-project-security-shepherd/>

Beginner Guide to OWASP: <https://blog.gitguardian.com/a-beginners-guide-to-owasp/>

OWASP Vulnerable Flask App: <https://owasp.org/www-project-vulnerable-flask-app/>

OWASP VWAD (Vulnerable WebApp Directory, developed in Ruby on Rails):
<https://owasp.org/www-project-vulnerable-web-applications-directory/>

GitLab SecureFlag Integration: https://gitlab.com/gitlab-org/gitlab/-/merge_requests/111592

GitLab Partner Solution Integration – SecureFlag: <https://gitlab.com/alliances/alliances/-/issues/297>

web and mobile application security training platform

Easy introduction to the community making free security tools and resources

lab environment created for people who want to improve themselves in the field of web penetration testing

well maintained registry of known vulnerable web and mobile applications currently available, to be used by web developers, security auditors, and penetration testers to practice their knowledge and skills

GitLab – SecureFlag integration



I.3c OWASP

Documentation

OWASP Documentation

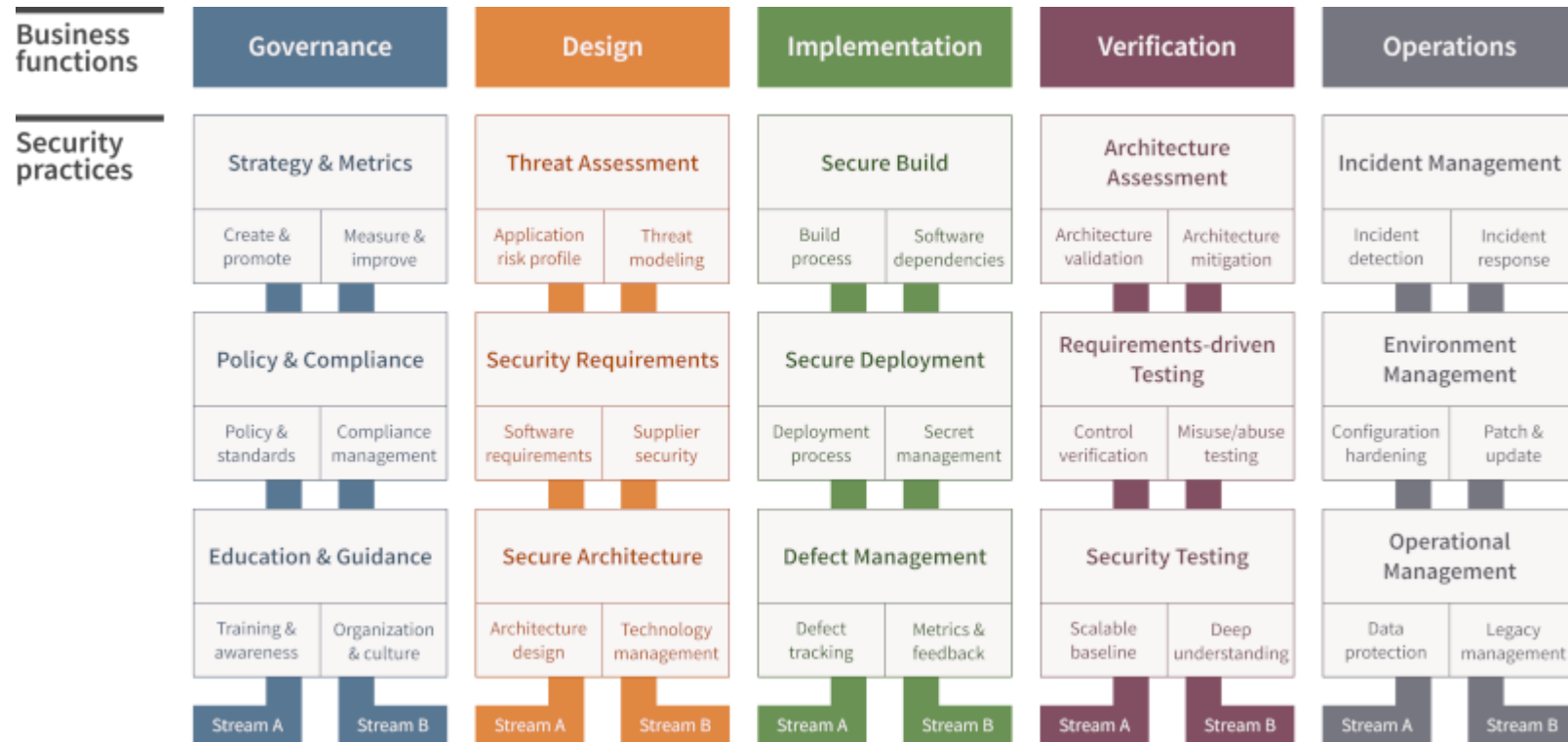
- The [Top Ten](#) is a very important document to learn more about the most critical web application security risks. Find the current version at owasp.org.
- The OWASP [Cheat Sheet Series](#) condenses the most important things to know about various vulnerabilities – as well as security features – into an easily-digestible format. It is also reasonably up-to-date.
- The OWASP [Security Knowledge Framework](#) provides guidance for designing secure web applications.
- For testers, the OWASP [Application Security Verification Standard](#) as well as the OWASP [Web Security Testing Guide](#) and the [Mobile Security Testing Guide](#) give guidance about what to target during a security test, and – more importantly – how to test for certain weaknesses.
- The OWASP [Software Assurance Maturity Model \(SAMM\)](#) is one of the commonly-used methodologies to build security into your software development process (alongside [BSIMM](#) and [Microsoft SDL](#)).
- DevSecOps Maturity Model: https://owaspsamm.org/presentations/SUD2021/SAMM_DevSecOps_Maturity_Model.pdf



I.3d OWASP

SAMM <https://owasp.org/www-project-samm/>

OWASP Software Assurance Maturity Model



I.3e NIST

SAMATE Tools <https://www.nist.gov/itl/ssd/software-quality-group/samate>

Software Assurance Metric And Tool Evaluation

NIST SOFTWARE ASSURANCE REFERENCE DATASET

A project driven by SAMATE

The Software Assurance Reference Dataset (SARD) is a growing collection of test programs with documented weaknesses. Test cases vary from small synthetic programs to large applications. The programs are in C, C++, Java, PHP, and C#, and cover over 150 classes of weaknesses. The [Acknowledgments and Test Case Descriptions](#) page describes the content. [The Manual](#) explains how to use the SARD website.

- Collection of more than 450,000 test cases
From piece of code to production software
- Various types of weaknesses
Covering more than 150 Common Weakness Enumeration classes (CWE)

•**SARD**: Software Assurance Reference Dataset (<https://samate.nist.gov/SARD/>) growing collection of test programs with documented weaknesses

NIST

Information Technology Laboratory / Software and Systems Division

SOFTWARE QUALITY GROUP

SAMATE

Static Analysis Tool Exposition (SATE)

•**SATE**: Static Analysis Tool Exposition (<https://www.nist.gov/itl/ssd/software-quality-group/samate/static-analysis-tool-exposition-sate>) recurring non-competitive study of static analysis tool effectiveness, aiming at improving tools and increasing public awareness and adoption



THE BUGS FRAMEWORK (BF) -
SOFTWARE DEVELOPERS' AND TESTERS' "BEST FRIEND"

•**BF**: the Bug Framework (<https://samate.nist.gov/BF/>) classifying software bugs and weaknesses to allow precise descriptions of vulnerabilities that exploit them

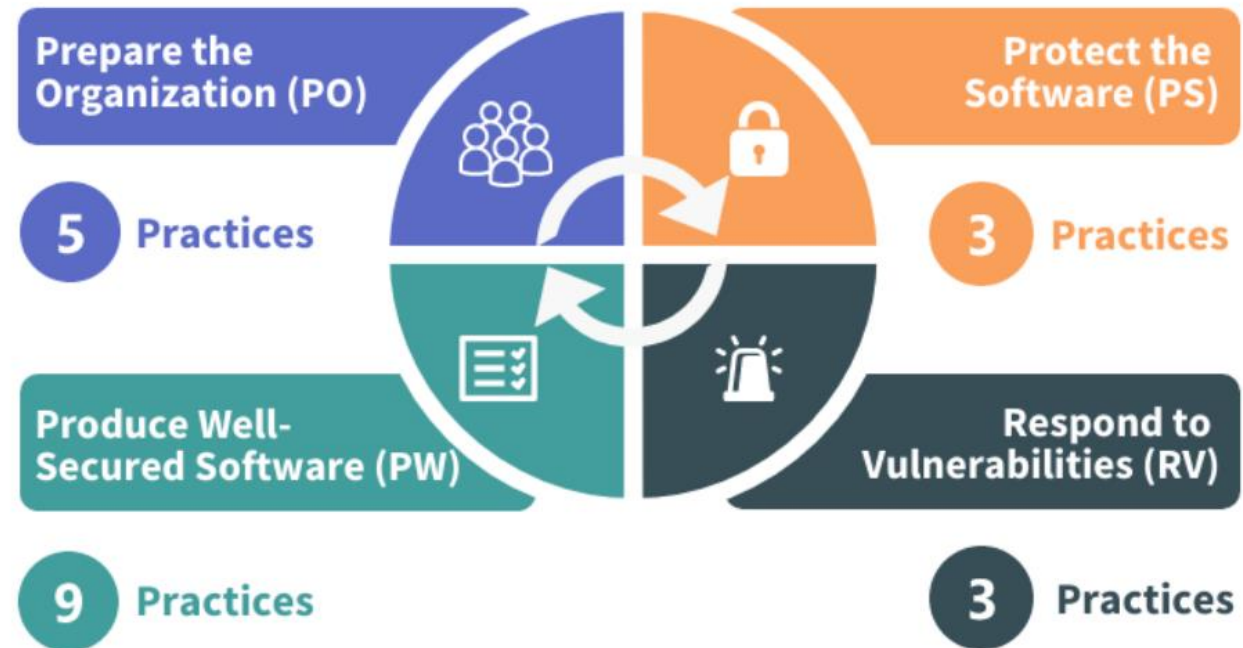


I.3f NIST

SSDF <https://csrc.nist.gov/Projects/ssdf>

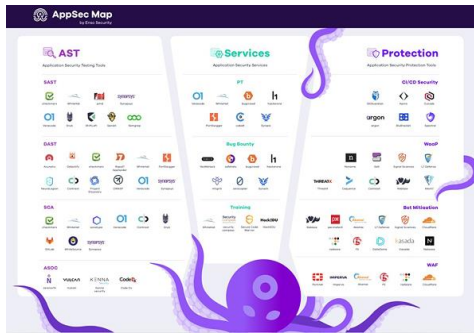
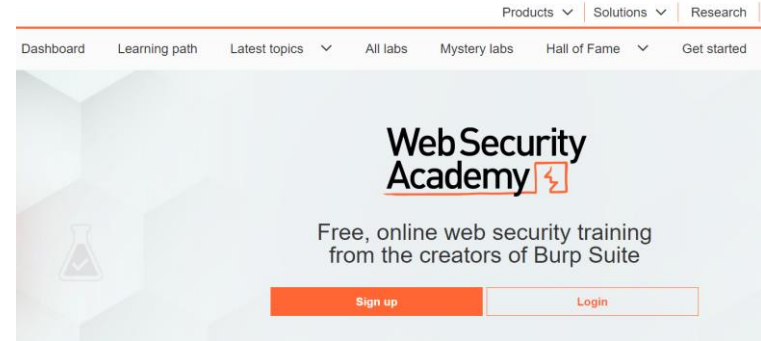
Secure Software Development Framework

set of fundamental, sound, and secure software development practices based on established secure software development practice documents from organizations such as BSA, OWASP, and SAFECode.



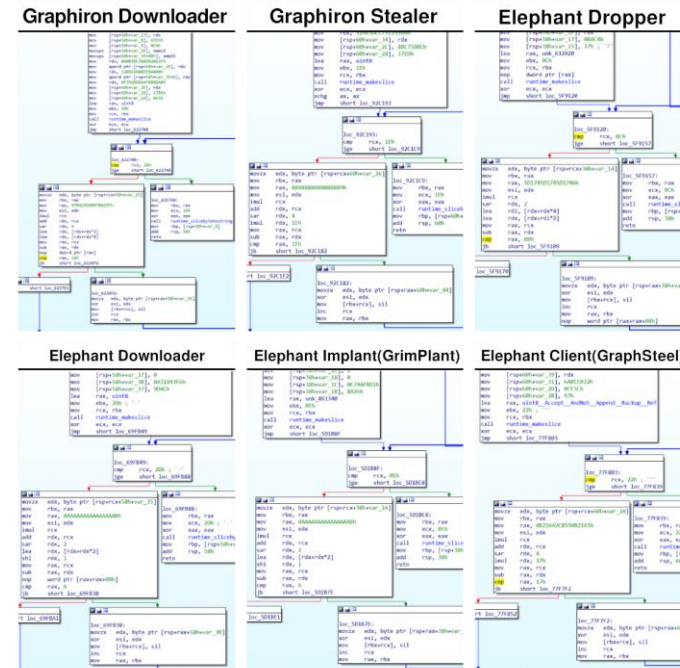
1.3g Other Tools

- PortSwigger (Creator of Burp suite) - Web Security Academy:
<https://portswigger.net/web-security>



- (Web) AppSecMap: <https://appsecmap.com/AppSecMap>

- TreathRay (Malware Code Reuse Analysis):
<https://threatray.com/blog/linking-and-tracking-uac-0056-tooling-through-code-reuse-analysis/>



I.3h Other

XSS Labs

- Google XSS Game - <https://xss-game.appspot.com/>

Excess XSS

A comprehensive tutorial on cross-site scripting

Created by [Jakob Kallin](#) and [Irene Lobo Valbuena](#)

[Overview](#) | [XSS Attacks](#) | [Preventing XSS](#) | [Summary](#)

- Reddit XSS: <https://www.reddit.com/r/xss/>

Warning: You are entering the XSS game area

Welcome, recruit!

Cross-site scripting (XSS) bugs are one of the most common and dangerous types of vulnerabilities in Web applications. These nasty buggers can allow your enemies to steal or modify user data in your apps and you must learn to dispatch them, pronto!

At Google, we know very well how important these bugs are. In fact, Google is so serious about finding and fixing XSS issues that we are paying mercenaries up to \$7,500 for dangerous XSS bugs discovered in our most sensitive products.

In this training program, you will learn to find and exploit XSS bugs. You'll use this knowledge to confuse and infuriate your adversaries by preventing such bugs from happening in your applications.

There will be cake at the end of the test.

- A comprehensive tutorial on cross-site scripting: <https://excess-xss.com/>

