Secure Programming

A.A. 2022/2023

Corso di Laurea in Ingegneria delle Telecomnicazioni

I. Free Security Tools

Paolo Ottolino

Politecnico di Bari

DIPARTIMENTO DI
INGEGNERIA ELETTRICA
E DELL'INFORMAZIONE

Secure Programming Lab: Course Program

A. Intro Secure Programming: «Who-What-Why-When-Where-How»
B. Building Security in: Buffer Overflow, UAF, Command Inection
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

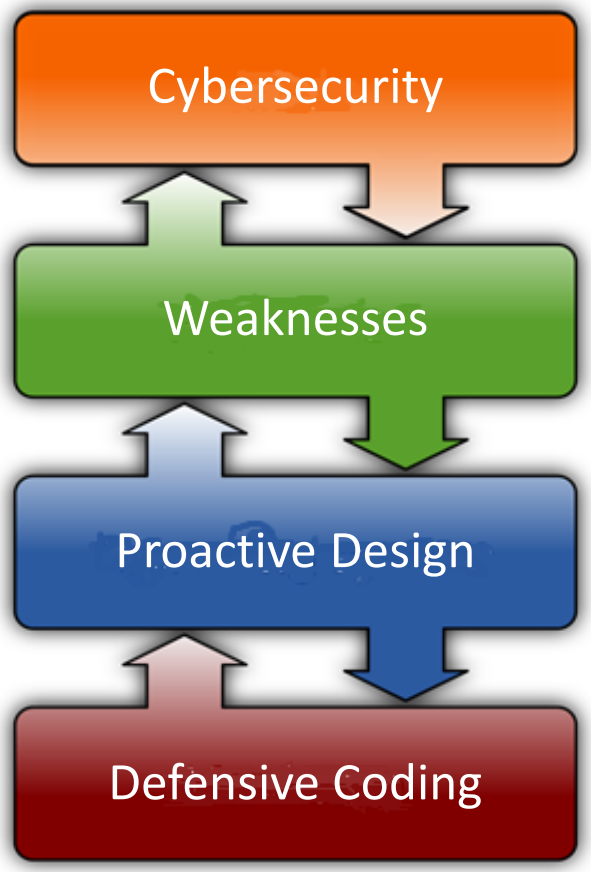
I.1 Recap & Shutters

I.2 Shutters drill down

I.3 Tools

I.1 Free Security Tools: Recap

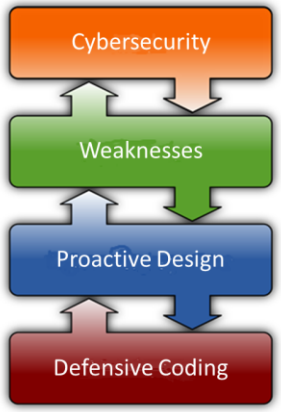
Secure Programming Arguments



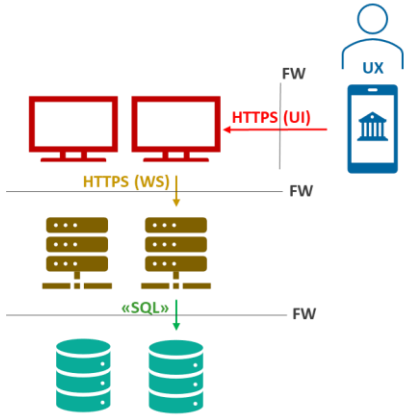
Goals	Techniques	Measures	Approaches	Abstractions
CIA (Attacker Profiles)	Protection Criteria: • Filtering • Hiding • Logging	Risk Rating • SLExARO (likelihood) • BIA • Framework Checklist	Risk Remediation • Avoid • Transfer • Mitigate • Accepts	Indicators • KPI • KGI • SLA • etc.
Attack Lifecycle: • Cyber Kill Chain • MITRE ATT&CK	Vulnerability • Lifecycle • Security Bulletins	CVE / CVSS	OWASP Top10	CWE RFC 4949 (Glossary)
Containers (Orchestration)	CSMA ZTA Pillars	Responsibility Sharing • IaaS • PaaS • SaaS	DevOps SecDevOps	SDO Maturity Model
Secure Coding Practices	Code Bugs • BOF • UAF • Uncontrolled Input	Bugs & Exploits	Input Validation • Checking Whitelist • Sanitizing Escape • Checking Blacklist • Sanitizing Blacklist	Shift Left

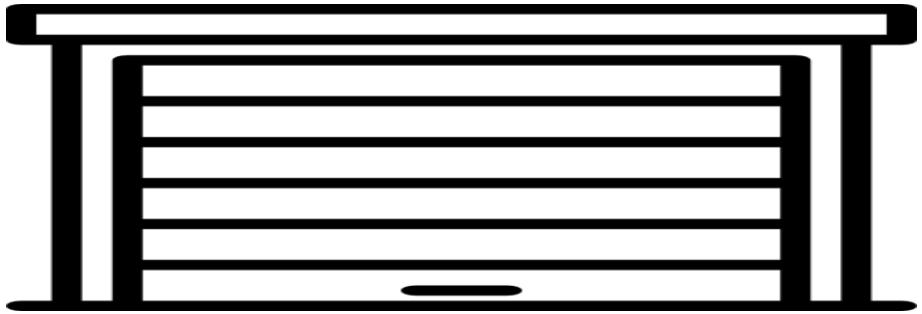
I.1a Free Security Tools: Recap

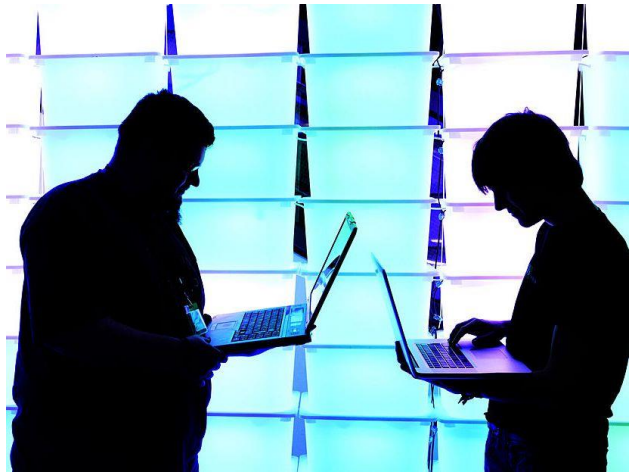
from Arguments to Shutters



Goals	Techniques	Measures	Approaches	Abstractions
CIA (Attacker Profiles)	Protection Criteria: • Filtering • Hiding • Logging	Risk Rating • SLExARO (likelihood) • BIA • Framework Checklist	Risk Remediation • Avoid • Transfer • Mitigate • Accepts	Indicators • KPI • KGI • SLA • etc.
Attack Lifecycle: • Cyber Kill Chain • MITRE ATT&CK	Vulnerability • Lifecycle • Security Bulletins	CVE / CVSS	OWASP Top10	CWE RFC 4949 (Glossary)
Containers (Orchestration)	CSMA ZTA Pillars	Responsibility Sharing • IaaS • PaaS • SaaS	DevOps SecDevOps	SDO Maturity Model
Secure Coding Practices	Code Bugs • BOF • UAF • Uncontrolled Input	Bugs & Exploits	Input Validation • Checking Whitelist • Sanitizing Escape • Checking Blacklist • Sanitizing Blacklist	Shift Left

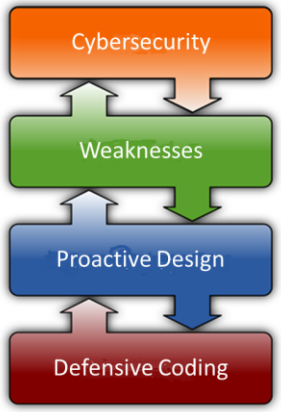




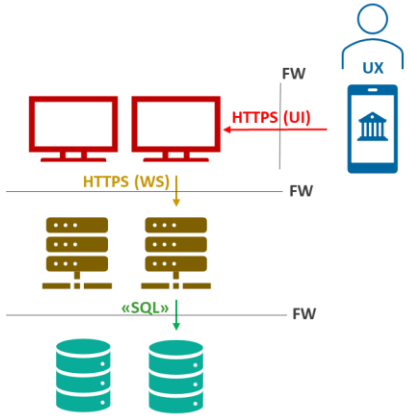


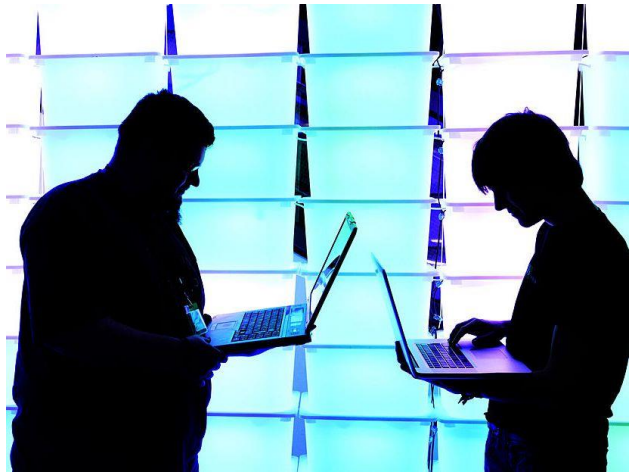
I.1b Free Security Tools: Recap

from Arguments to Shutters



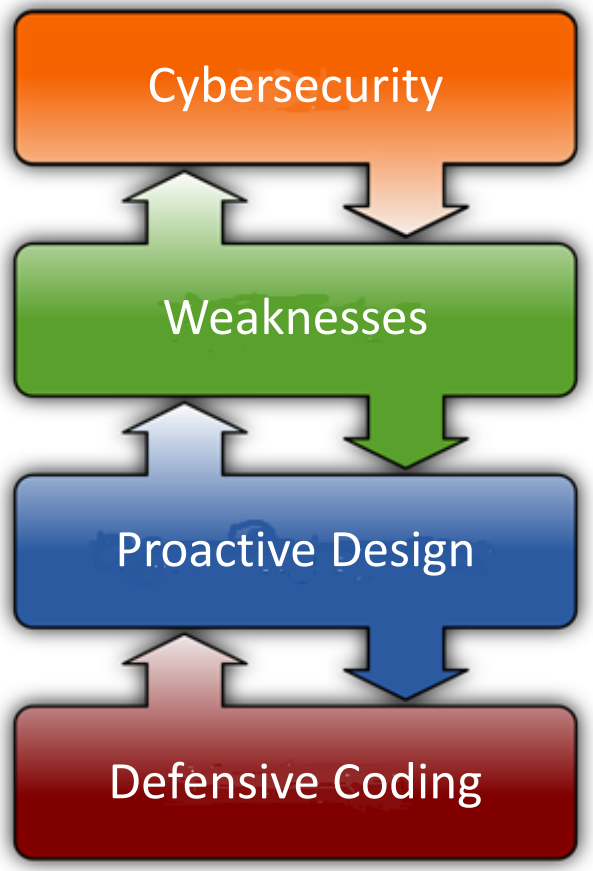
Goals	Techniques	Measures	Approaches	Abstractions
CIA (Attacker Profiles)	Protection Criteria: • Filtering • Hiding • Logging	Risk Rating • SLExARO (likelihood) • BIA • Framework Checklist	Risk Remediation • Avoid • Transfer • Mitigate • Accepts	Indicators • KPI • KGI • SLA • etc.
Attack Lifecycle: • Cyber Kill Chain • MITRE ATT&CK	Vulnerability • Lifecycle • Security Bulletins	CVE / CVSS	OWASP Top10	CWE RFC 4949 (Glossary)
Containers (Orchestration)	CSMA ZTA Pillars	Responsibility Sharing • IaaS • PaaS • SaaS	DevOps SecDevOps	SDO Maturity Model
Secure Coding Practices	Code Bugs • BOF • UAF • Uncontrolled Input	Bugs & Exploits	Input Validation • Checking Whitelist • Sanitizing Escape • Checking Blacklist • Sanitizing Blacklist	Shift Left

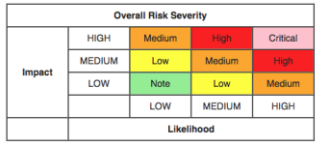
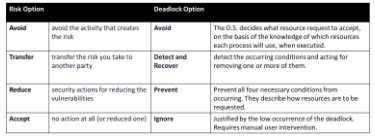
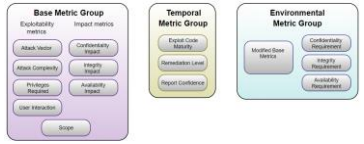
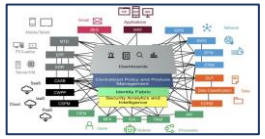
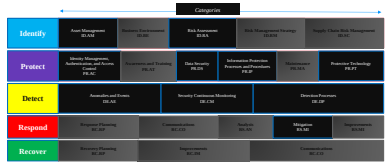
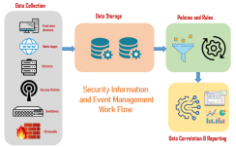
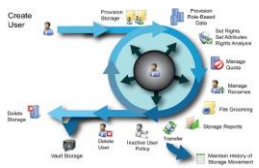
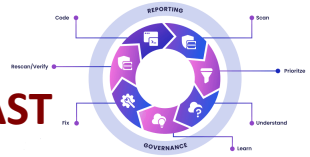




I.1c Free Security Tools: Shutters

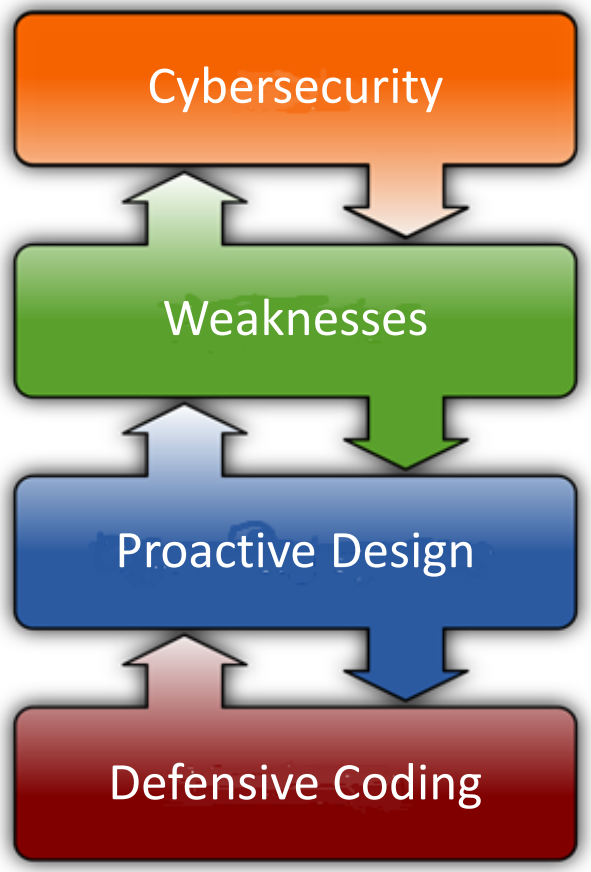
Secure Programming Shutters



Know	Reduce	Evaluate	Execute	Process
 Attackers	 Protection	 Risk Rating	 Risk Mgmt	Monitor Plan 
 CVE	 SCA	 CVSS	 Vulnerability Mgmt	Test Release 
 CSMA	 ZTA (Pillars)	DAST 	 Architecture Mgmt	Deploy Operate 
 (Audit) Log	 Access Control	SAST 	 Input Mgmt	Code Build 
SOAR 	IGA 	KPI 		

I.1d Free Security Tools: Shutters

Secure Programming Shutters

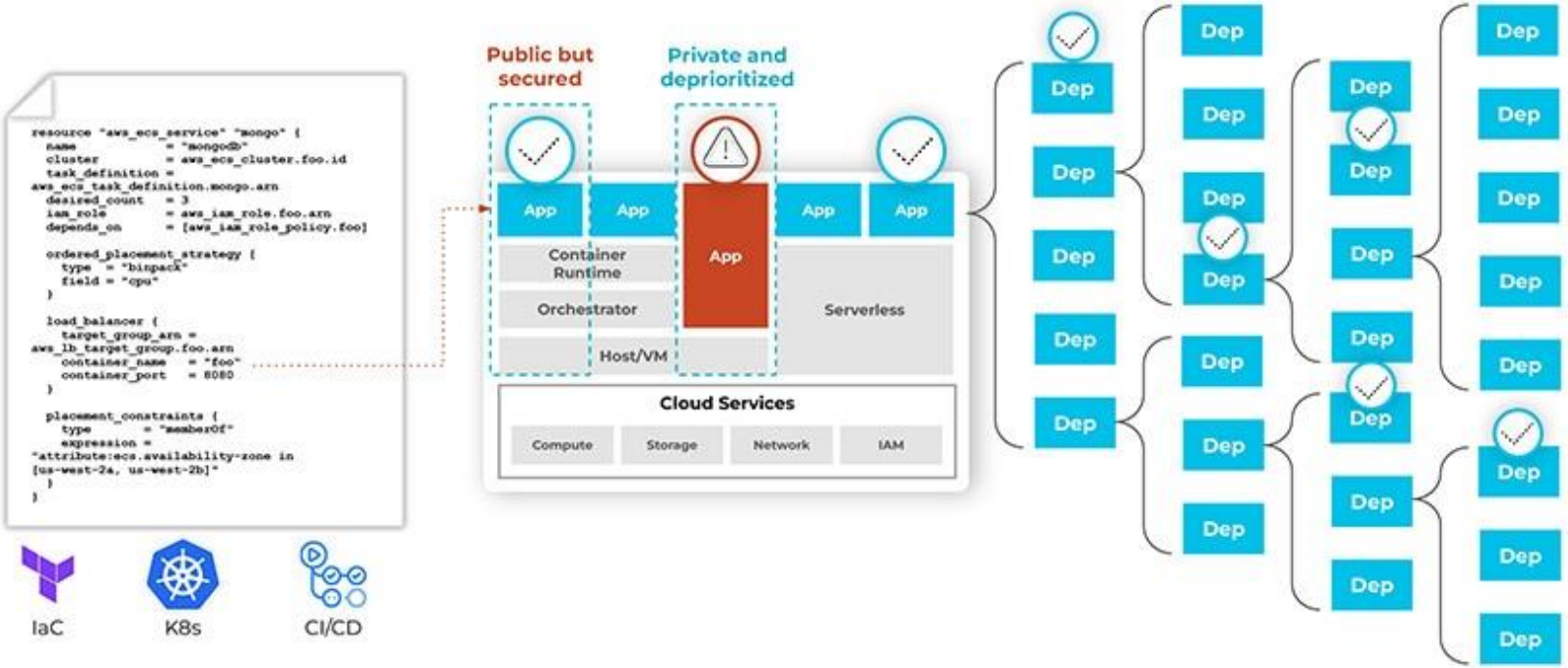


Know	Reduce	Evaluate	Execute	Process
Attackers • Profile • Trends • Motive • Opportunity • Means	**Protection** • AAA • Duplicate • Filter • Log • Encode	**Risk Rating** • Likelihood • Impact • Level	**Risk Mgmt** • Avoid • Transfer • Mitigate • Accepts	**Monitor** **Plan**
CVE • Description • Severity • References • Weaknesses • Configuration	**SCA** • Identify • Dependences • Vulnerability • OSInt, CLOSInt • Speed	**CVSS** • Exploitability • Impact • Scope	**Vulnerability Mgmt** • Substitute • Virtual Patch • Patch • Ignore/Postpone	**Test** **Release**
CSMA • Users • Cloud/On-premise • Network • Application • Data	**ZTA (Pillars)** • Identity • Endpoint • Network • Workload • Data	**DAST** • Explore • Test • Evaluate	**Architecture Mgmt** • WAF • Supplier • Implement • Ignore/Postpone	**Deploy** **Operate**
(Audit) Log • Date, Time • User, Device • Net Addr, Prot • Location • Event/Activity	**Access Control** • Identify • AuthN • AuthZ • Govern (Certify) • Monitor	**SAST** • Scan • Prioritize • Verify	**Input Mgmt** • Checking Whitelist • Sanitizing Escape • Checking Blacklist • Sanitizing Blacklist	**Code** **Build**
SOAR	**IGA**	**KPI**		

I.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**

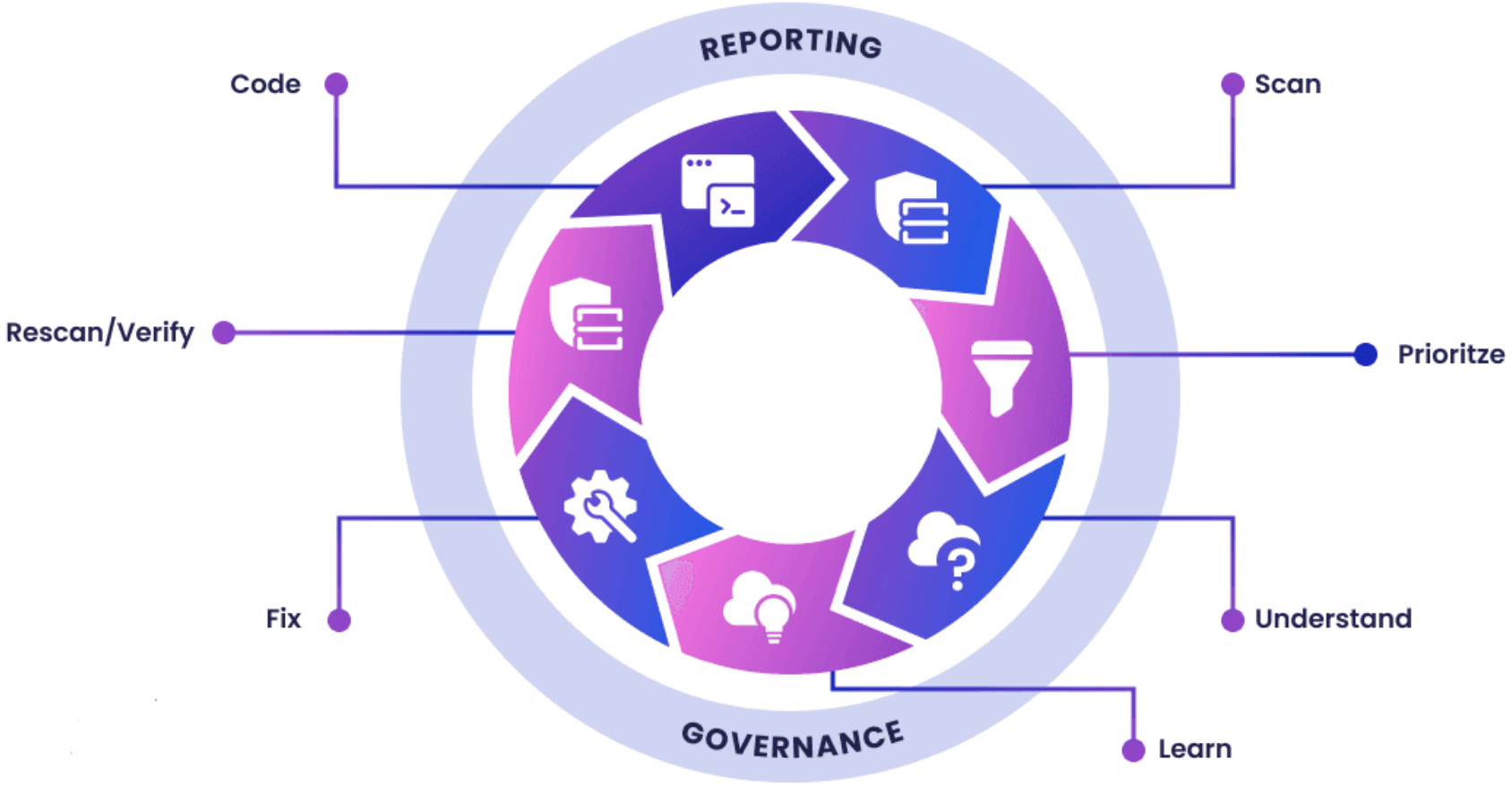


https://snyk.io/series/open-source-security/software-composition-analysis-sca/
https://www.paloaltonetworks.com/cyberpedia/what-is-sca

I.2c Drill Down: Statis Application Security Testing

7 Stages of SAST

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

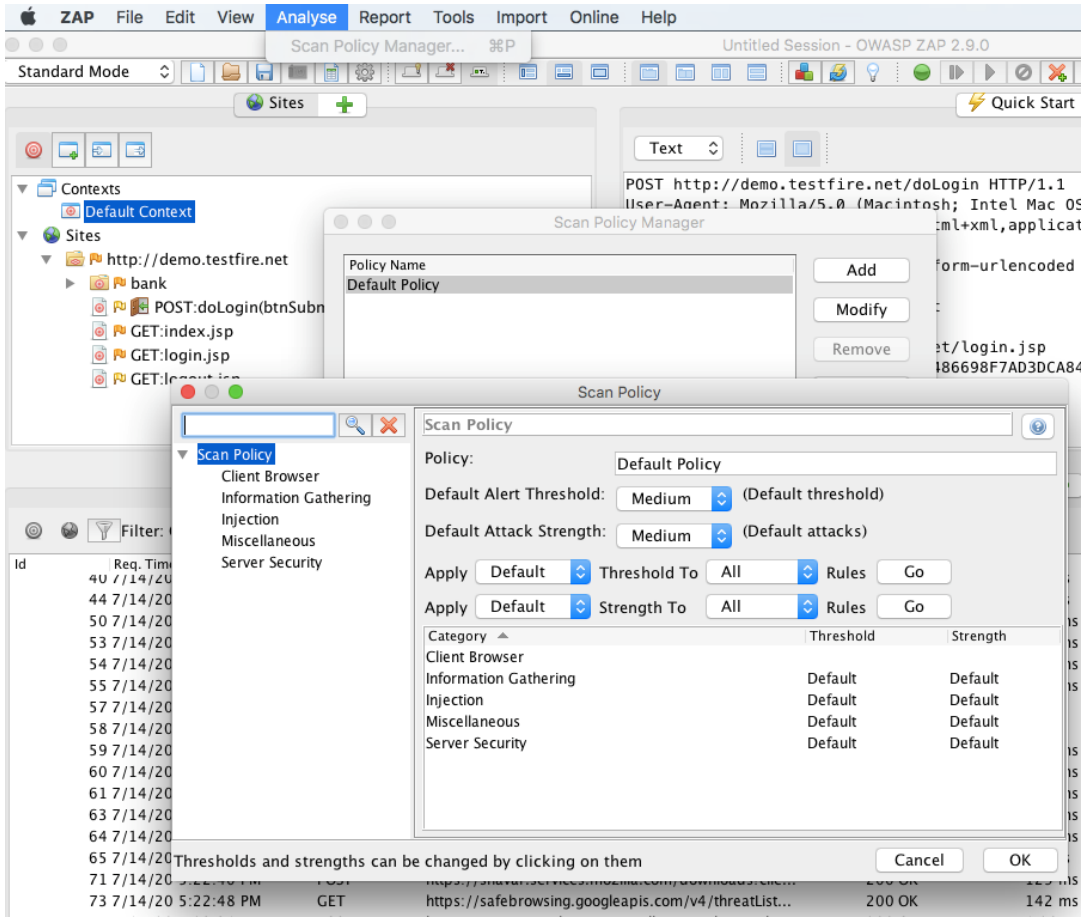


https://snyk.io/learn/application-security/static-application-security-testing/

I.2d Drill Down: Dynamic Application Security Testing

Using Zed Attack Proxy

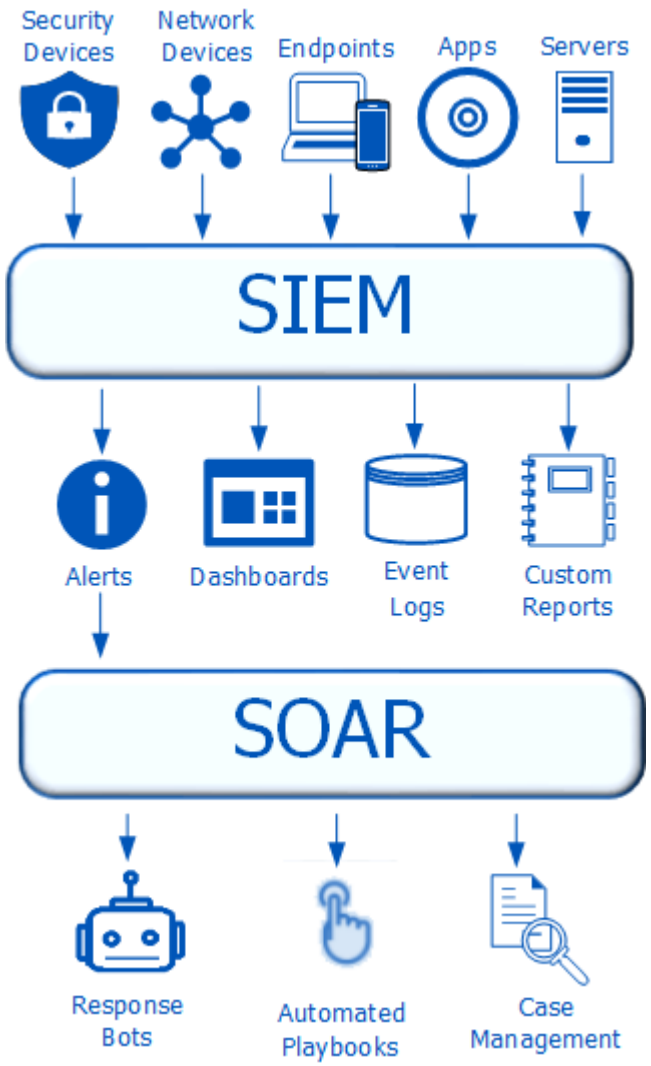
1. **Passive Scan**: the intended applications under assessment are being **intercepted** by the tool & those requests / responses are observed by the tool to flag security misconfigurations such as missing security headers or cookie settings. The tool doesn't send any new requests on its own in this 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

I.3b OWASP

Labs

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

web and mobile application security training platform

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

Easy introduction to the community making free security tools and resources

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

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

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

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: https://gitlab.com/gitlab-org/gitlab/-/merge_requests/111592

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

GitLab – SecureFlag integration

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.

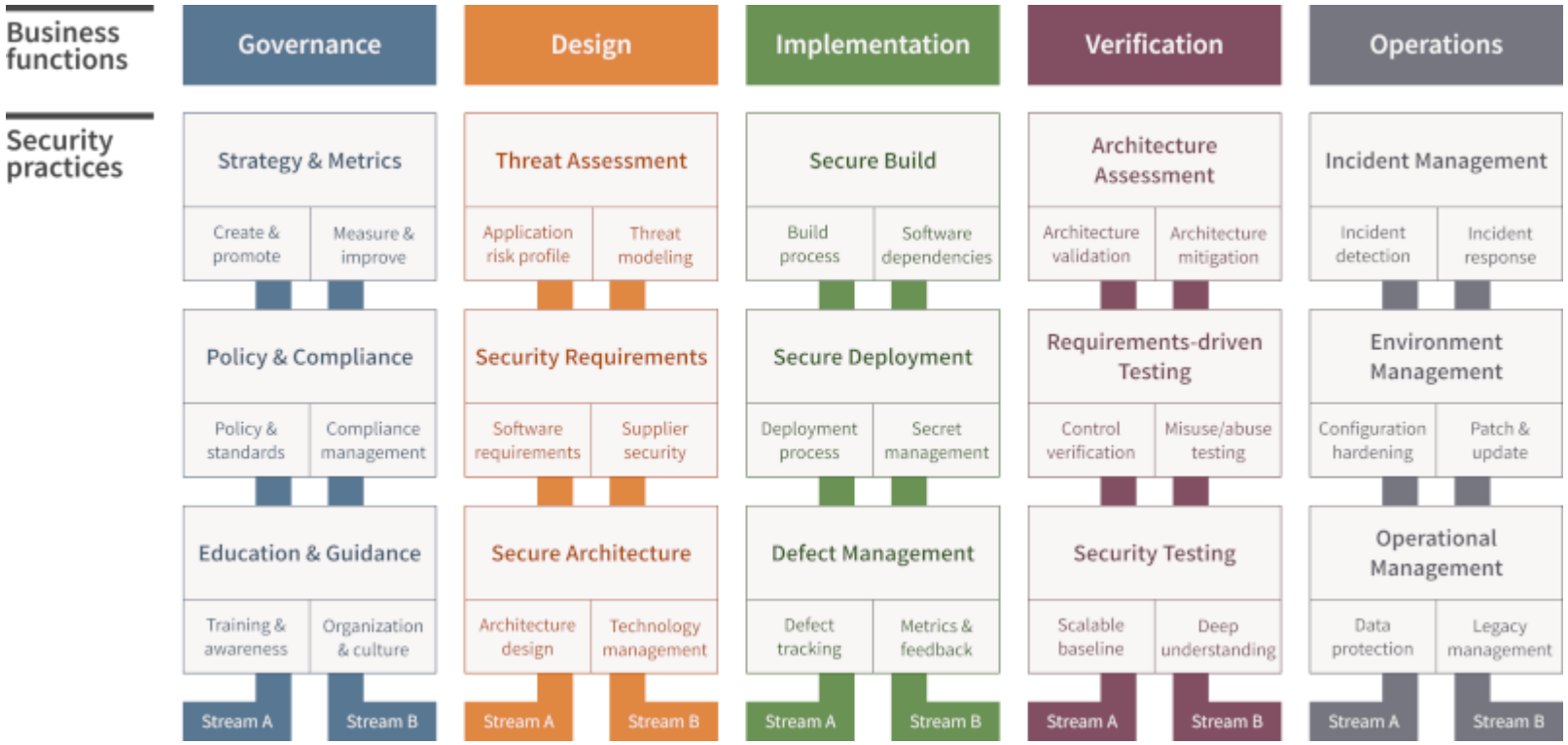
•The OWASP Software Assurance Maturity Model (SAMM) is one of the commonly-used methodologies to build security into your software development process (alongside BSIMM and Microsoft SDL).

•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

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

•**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

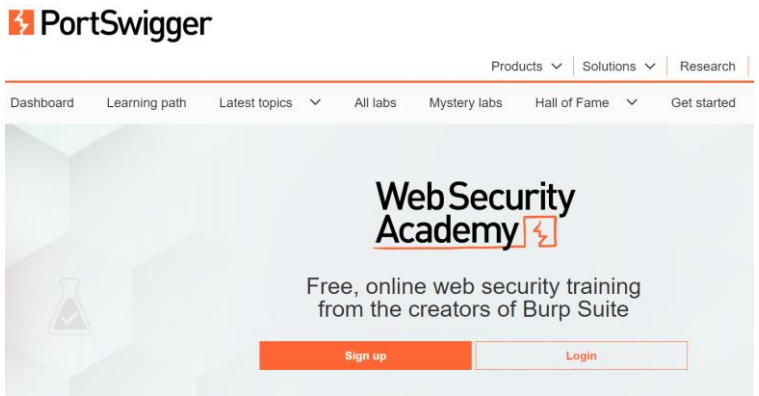
Secure Software Development Framework

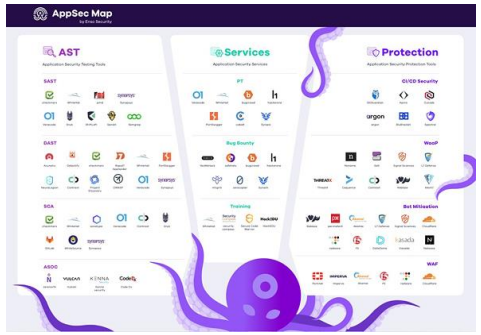
set of fundamental, sound, and secure software development practices based on established secure software development practice documents from organizations such as BSA, OWASP, and SAFECode.

I.3g Other

Tools

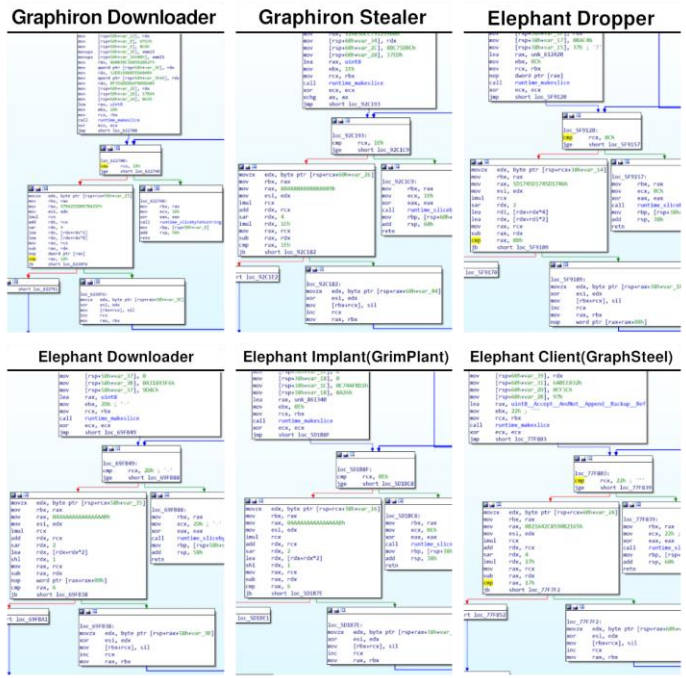
•PortSwigger (Creator of Burp suite) - Web Security Academy: https://portswigger.net/web-security





•(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/

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

•Reddit XSS: https://www.reddit.com/r/xss/