Clusit Associazione Italiana per la Sicurezza Informatica

SECURITY TECHNOLOGIES MANAGEMENT COMPLIANCE SUMMIT

Streaming Edition 2021

ASTREA Advanced Security, Training Research, Events Agency

sessione

Cosa stavi dicendo?
I pericoli delle inconsistenze nelle applicazioni web

Stefano Calzavara, Università Ca' Foscari & OWASP Italy Chair

Matteo Meucci, CEO IMQ Minded Security & OWASP Italy Chair

17 Marzo 2021 orario 16:30-17:30 - StreamingEdition

#securitysummit #streamingedition

Agenda

- Introduction to OWASP
- OWASP for Secure Software Roadmap
- How can you use these resources to implement your roadmap
- What are the most common issues?
- What are the keys to implement a successful roadmap?

Who am I?

Informatics Engineer (since 2001)

Research

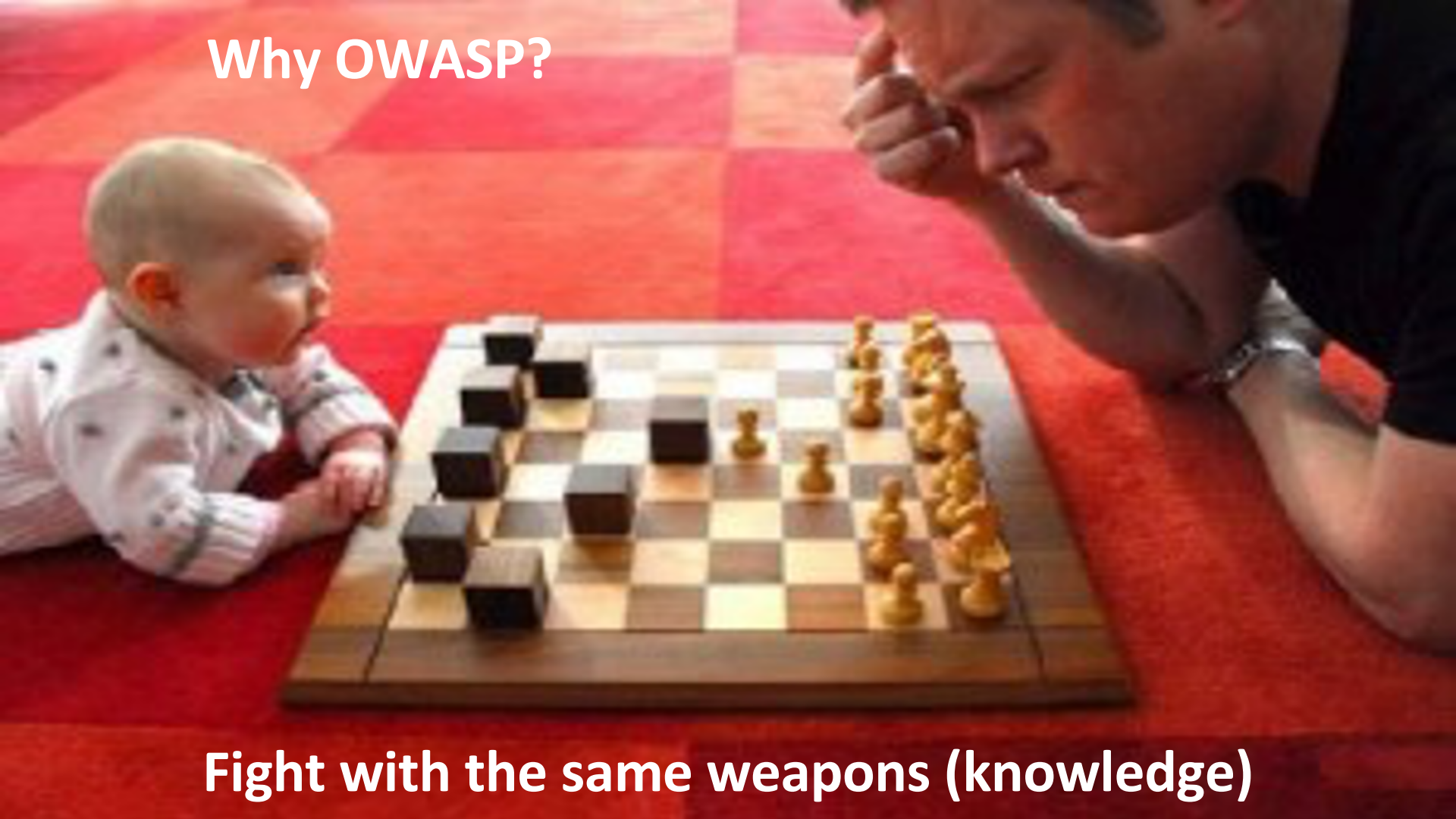
- OWASP contributor (since 2002)
- OWASP-Italy Chair (since 2005)
- OWASP Testing Guide Lead (since 2006)

Work

- 19+ years on Information Security focusing on Software Security
- CEO @ IMQ Minded Security an IMQ Group Company – The Software Security Company (since 2007)

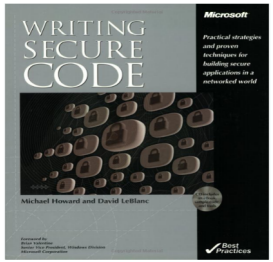
1. Introduction to OWASP

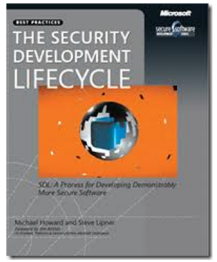
Why OWASP?



Fight with the same weapons (knowledge)

When?

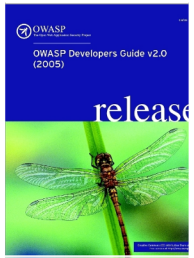


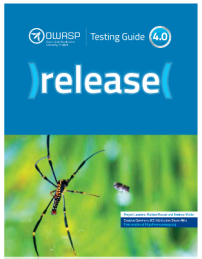


OWASP
Open Web Application Security Project

From: Bill Gates
Sent: Tuesday, January 15, 2002 5:22 PM
To: to every full-time employee at Microsoft
Subject: Trustworthy computing

...new capabilities is the fact that it is designed from the ground up to deliver **Trustworthy Computing.**



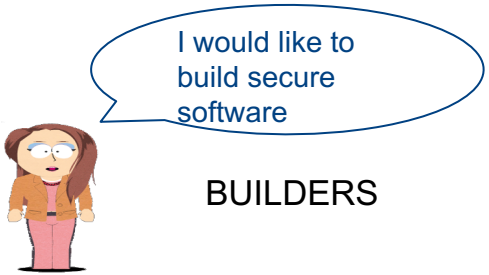


2001 **2002** **2004** **2005** **2006**

OWASP: The Open Web Application Security Project

- PROTECT - These are tools and documents that can be used to guard against security-related design and implementation flaws.
- DETECT - These are tools and documents that can be used to find security-related design and implementation flaws.
- LIFE CYCLE - These are tools and documents that can be used to add security-related activities into the Software Development Life Cycle (SDLC).

OWASP has ~140 Projects



BUILDERS

OWASP Pro Active CONTROLS

OWASP CHEAT SHEETS

OWASP has ~140 Projects

I would like to build secure software

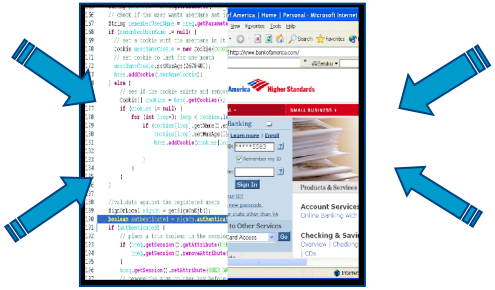
BUILDERS

I would like to find all the security bugs in my software

BREAKERS

OWASP Pro Active CONTROLS

OWASP CHEAT SHEETS



OWASP has ~140 Projects

I would like to build secure software

BUILDERS

I would like to find all the security bugs in my software

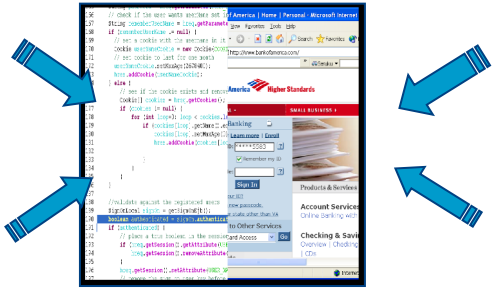
BREAKERS

I would like to implement a Roadmap for Software Security

MANAGERS

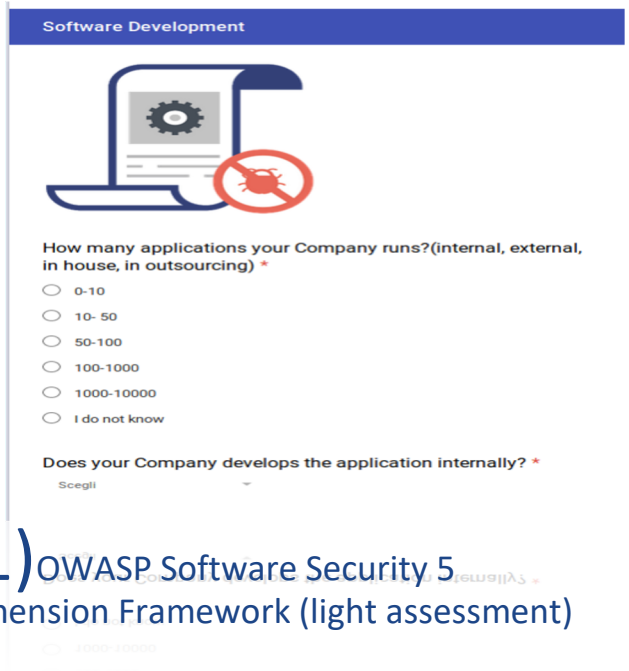
OWASP Pro Active CONTROLS

OWASP CHEAT SHEETS



2. OWASP for Secure Software Roadmap

Software Security Assessment



(1) OWASP Software Security 5 Dimension Framework (light assessment)

(2) OWASP Open SAMM is an OWASP Standard

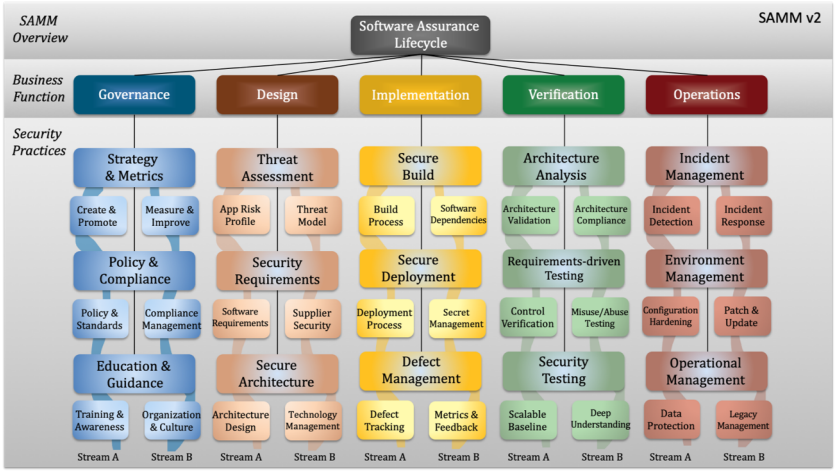
OWASP SAMM v2

OWASP SAMM (Software Assurance Maturity Model) is the OWASP framework to help organizations assess, formulate, and implement, a strategy for software security they can be integrated into their existing Software Development Lifecycle (SDLC).

OWASP SAMM has becoming the standard de facto to conduct the assessment.

The three main characteristics of SAMM are:

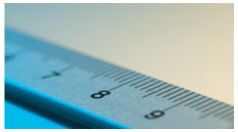
- **Measurable:** Defined maturity levels across security practices
- **Actionable:** Clear pathways for improving maturity levels
- **Versatile:** Technology, process, and organization agnostic



The goal is to evaluate the current state of the maturity of the organization in conducting software security activities within the SDLC and derive a roadmap that the organization can follow to improve his capabilities in software security.

OWASP SAMM: objectives

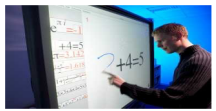
The SAMM's goals are:



Define and measure security-related activities throughout the organization

Evaluate an organization's existing software security practices

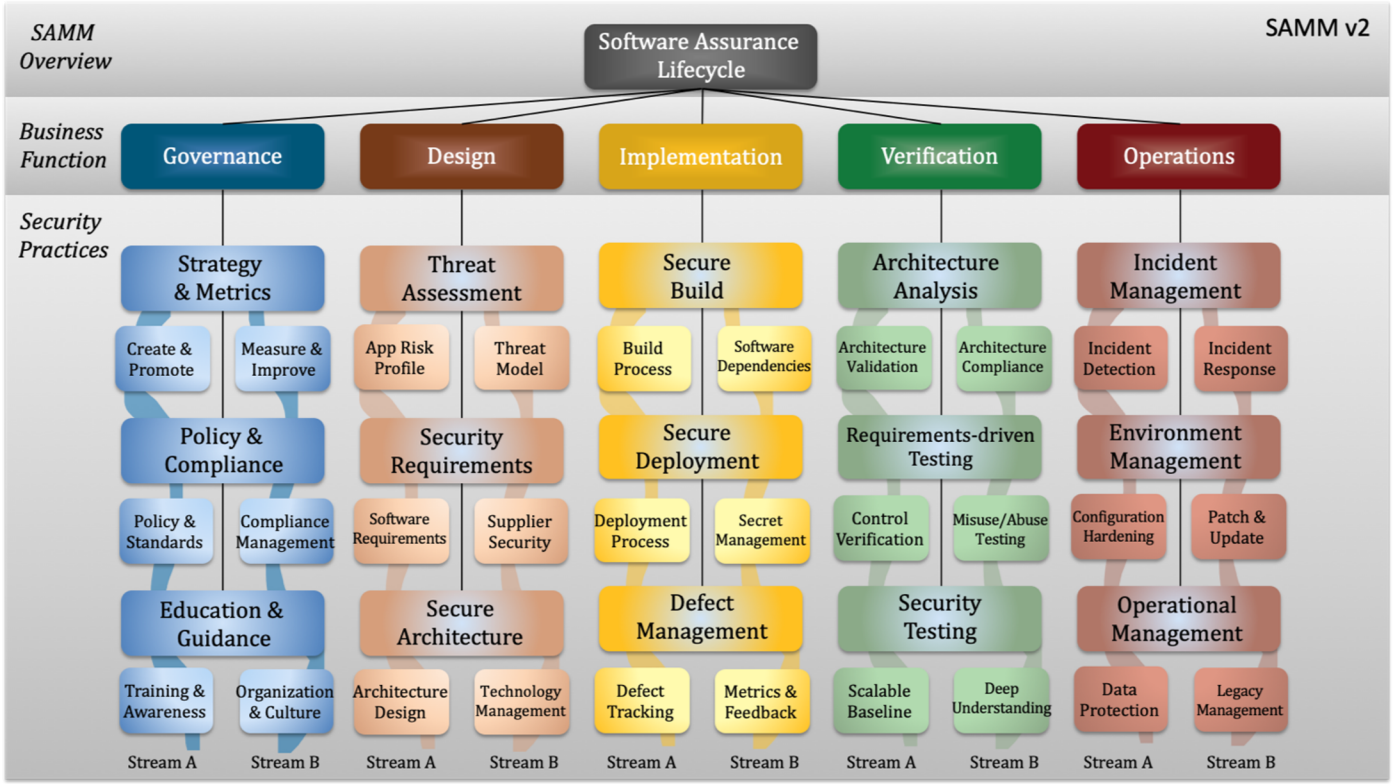
Build a balanced software security assurance program in well-defined iterations



Demonstrate concrete improvements to a security assurance program

3. How can you use these resources to implement your roadmap

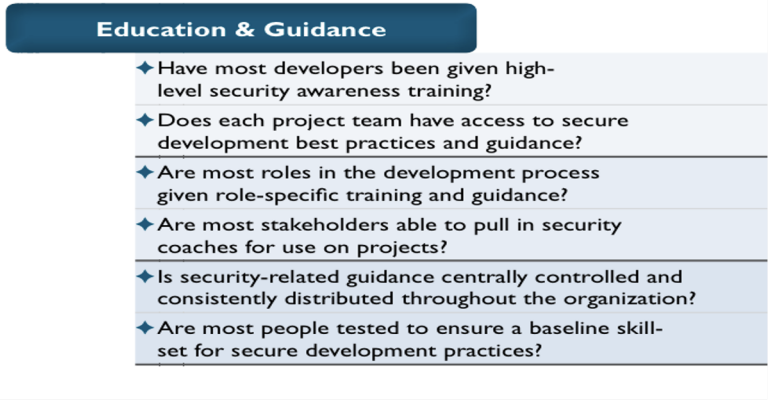
OWASP SAMM v2 Framework

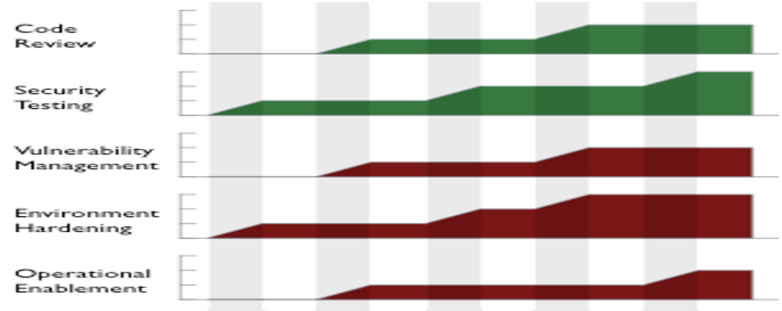


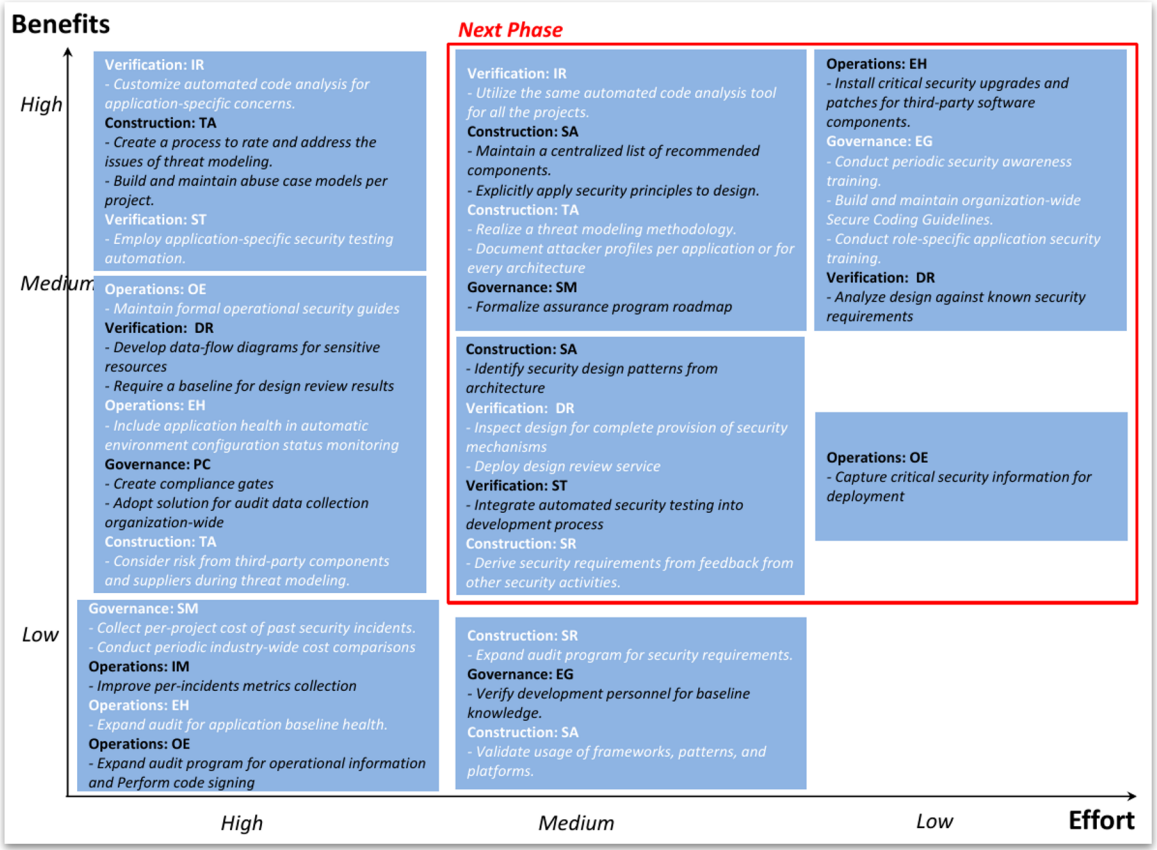
SAMM activities

0. Collect the names and functions of the people involved in the assessment with the SAMM sponsor (Roles and responsibility)

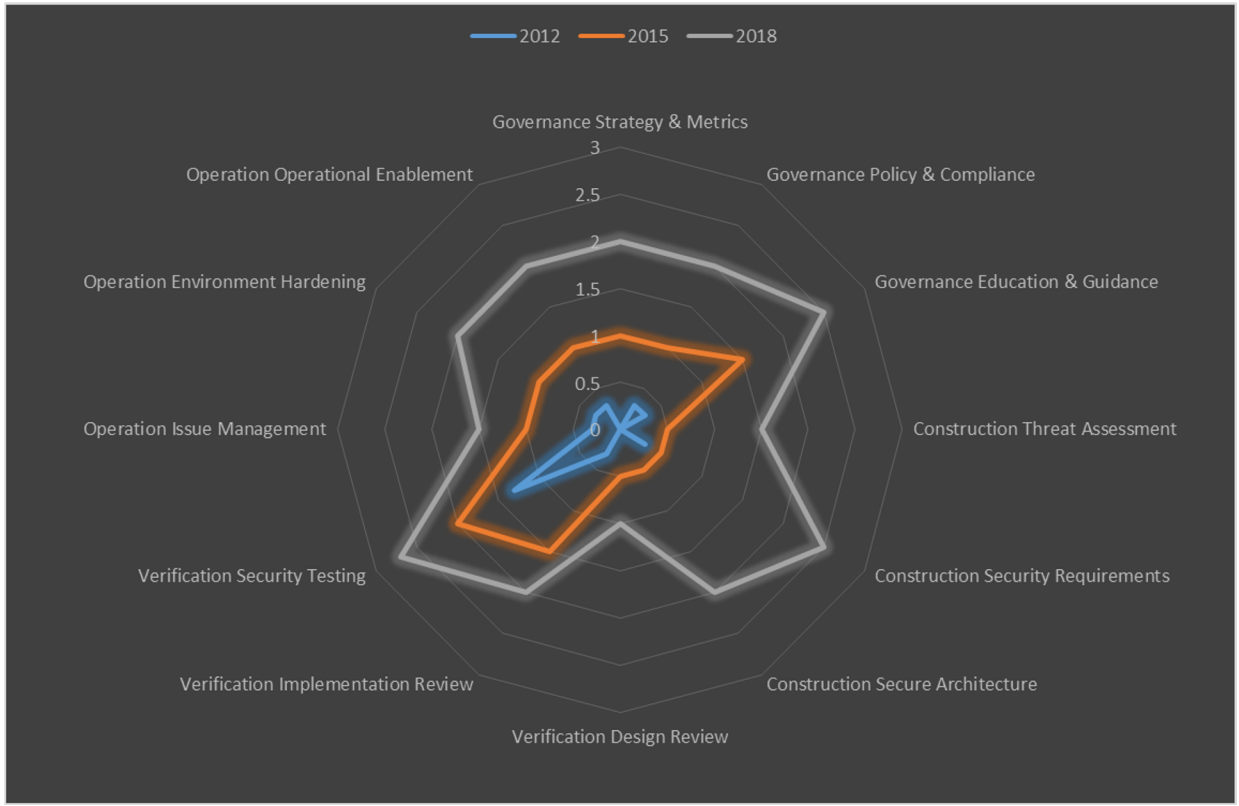
1. **Conduct the first assessment**
2. **Create a score card**
3. **Create a Software Security Program**
    1. Metrics
    2. Road map
4. **Implement the objectives of the roadmap and conduct a new assessment**



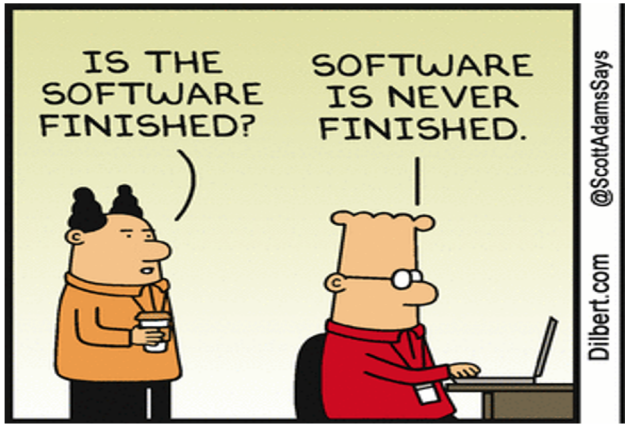



Benefits
Next Phase
High
Medium
Low
Verification: IR
- Customize automated code analysis for application-specific concerns.
Construction: TA
- Create a process to rate and address the issues of threat modeling.
- Build and maintain abuse case models per project.
Verification: ST
- Employ application-specific security testing automation.
Operations: OE
- Maintain formal operational security guides
Verification: DR
- Develop data-flow diagrams for sensitive resources
- Require a baseline for design review results
Operations: EH
- Include application health in automatic environment configuration status monitoring
Governance: PC
- Create compliance gates
- Adopt solution for audit data collection organization-wide
Construction: TA
- Consider risk from third-party components and suppliers during threat modeling.
Governance: SM
- Collect per-project cost of past security incidents.
- Conduct periodic industry-wide cost comparisons
Operations: IM
- Improve per-incidents metrics collection
Operations: EH
- Expand audit for application baseline health.
Operations: OE
- Expand audit program for operational information and Perform code signing
Verification: IR
- Utilize the same automated code analysis tool for all the projects.
Construction: SA
- Maintain a centralized list of recommended components.
- Explicitly apply security principles to design.
Construction: TA
- Realize a threat modeling methodology.
- Document attacker profiles per application or for every architecture
Governance: SM
- Formalize assurance program roadmap
Operations: EH
- Install critical security upgrades and patches for third-party software components.
Governance: EG
- Conduct periodic security awareness training.
- Build and maintain organization-wide Secure Coding Guidelines.
- Conduct role-specific application security training.
Verification: DR
- Analyze design against known security requirements
Construction: SA
- Identify security design patterns from architecture
Verification: DR
- Inspect design for complete provision of security mechanisms
- Deploy design review service
Verification: ST
- Integrate automated security testing into development process
Construction: SR
- Derive security requirements from feedback from other security activities.
Operations: OE
- Capture critical security information for deployment
Construction: SR
- Expand audit program for security requirements.
Governance: EG
- Verify development personnel for baseline knowledge.
Construction: SA
- Validate usage of frameworks, patterns, and platforms.
High
Medium
Low
Effort

Case Studies: 2012-2015-2018 (SAMM v1.5)

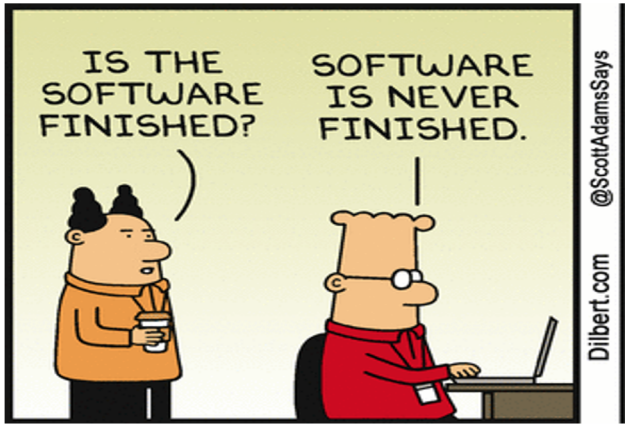


4. What are the most common issues?



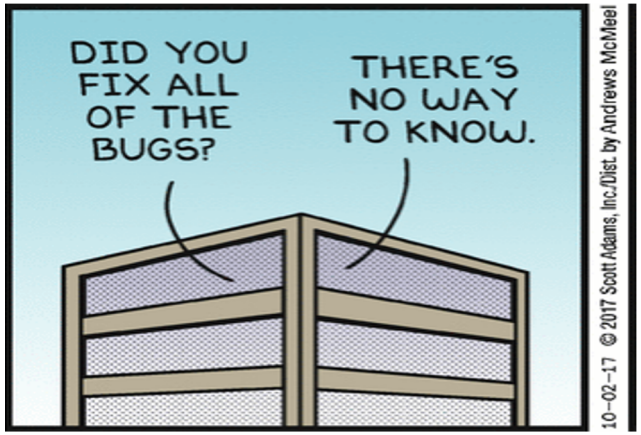
Testing is just one of AppSec dimensions

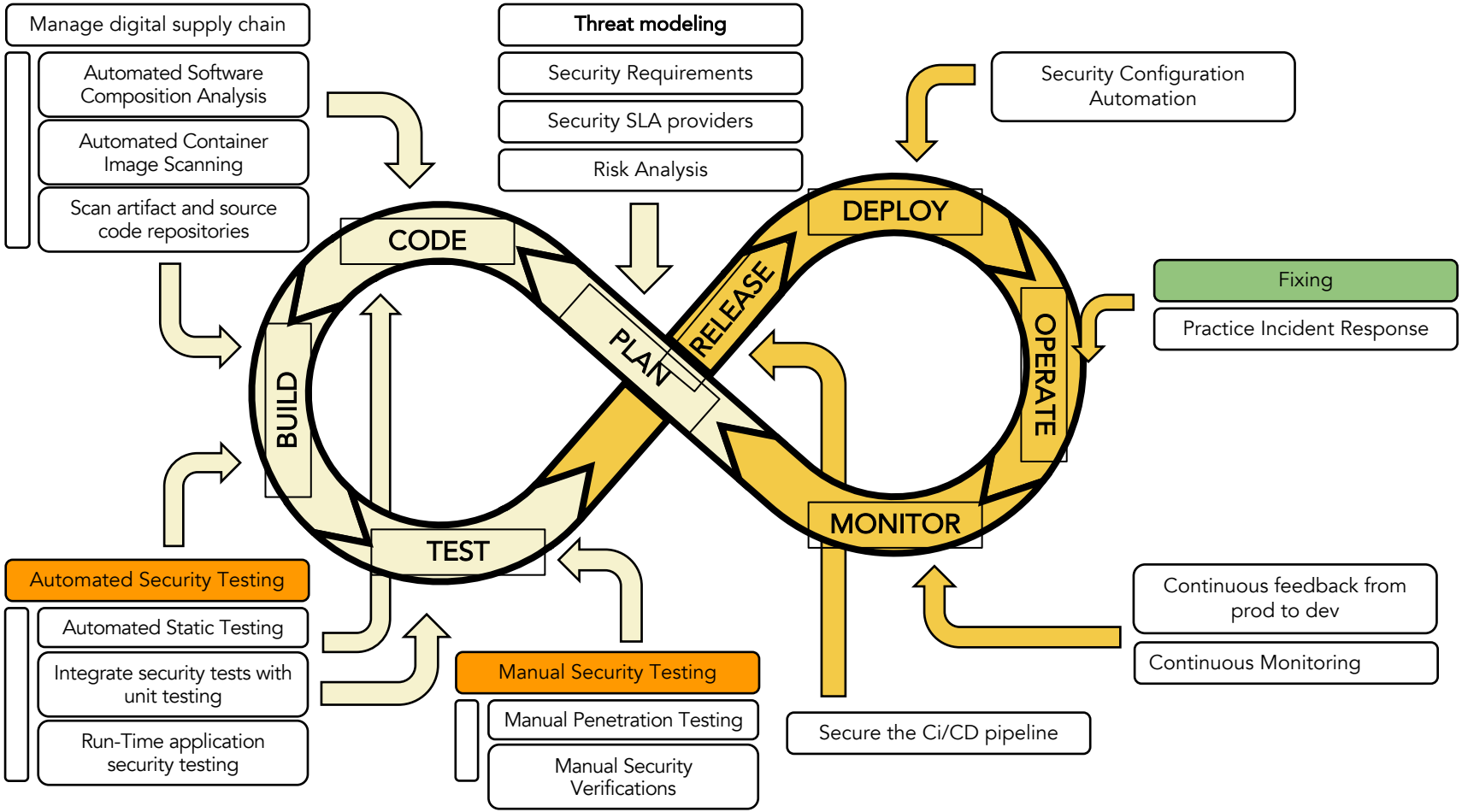
Testing is the solution to SwSec?



Testing is just one of AppSec dimensions

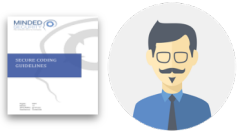
Fixing ASAP is the most important aspect of AppSec




Manage digital supply chain
Automated Software Composition Analysis
Automated Container Image Scanning
Scan artifact and source code repositories
Threat modeling
Security Requirements
Security SLA providers
Risk Analysis
Security Configuration Automation
CODE
BUILD
PLAN
RELEASE
DEPLOY
OPERATE
MONITOR
TEST
Fixing
Practice Incident Response
Automated Security Testing
Automated Static Testing
Integrate security tests with unit testing
Run-Time application security testing
Manual Security Testing
Manual Penetration Testing
Manual Security Verifications
Secure the Ci/CD pipeline
Continuous feedback from prod to dev
Continuous Monitoring


12:34

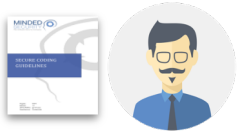
Yesterday: too much time to remediate



Manager

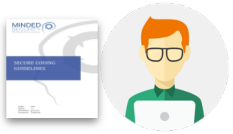
Time

Yesterday: too much time to remediate



Manager

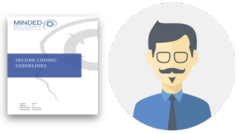
1w to 1 month



Dev team

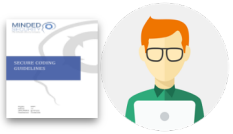
Time

Yesterday: too much time to remediate



Manager

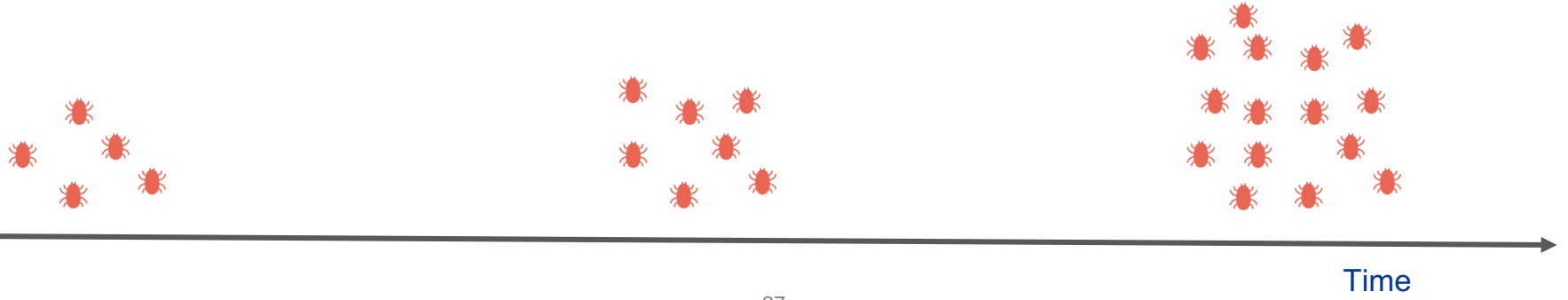
1w to 1 month



Dev team

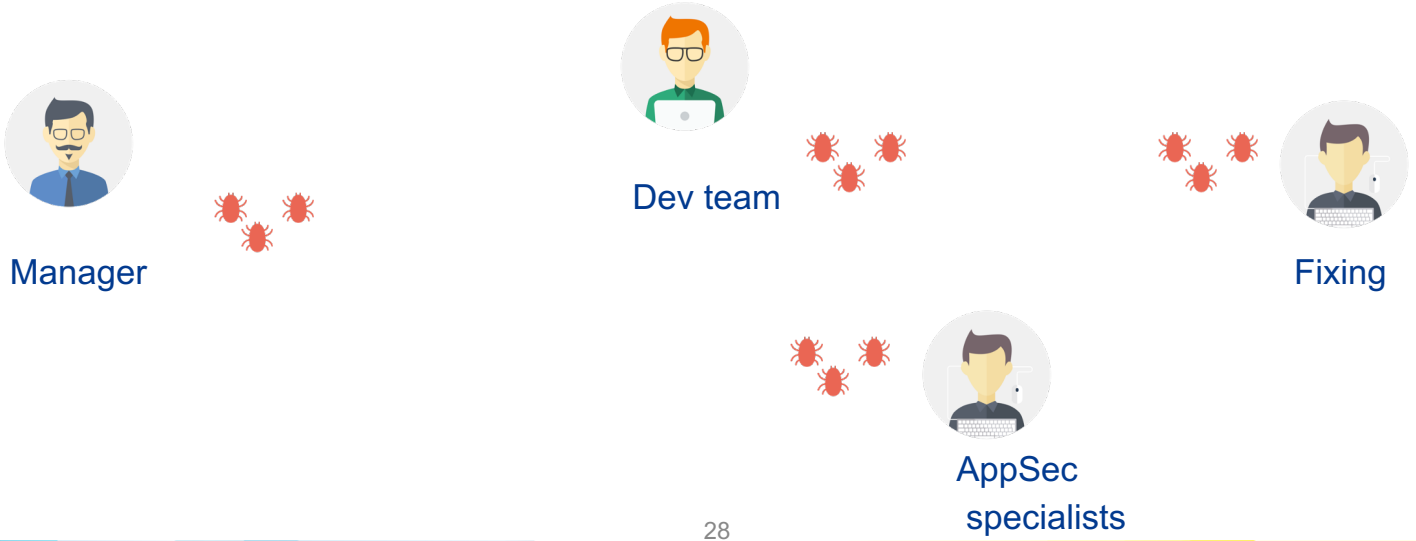
1w to 1 month

Fixing



What we need today?

1. **We need instant security feedback**
2. **Security must be shared**



A successful Software Security roadmap

- Stop thinking swsec is only testing!
- Stop thinking swsec is a developers responsibilities

- **Start sharing security bugs and fix it asap**
- **Start thinking that everyone is responsible for security**

5. What are the keys to implement a successful roadmap

A successful Software Security roadmap

- **Stop thinking swsec is only testing!**
- **Stop thinking swsec is a developers responsibilities**

A successful Software Security roadmap

- Stop thinking swsec is only testing!
- Stop thinking swsec is a developers responsibilities

- Start sharing security bugs and fix it asap
- Start thinking that everyone is responsible for security

- **OWASP SAMM Assessment and 5D Framework are standards that allows you to create a Software Security program**

A successful Software Security roadmap

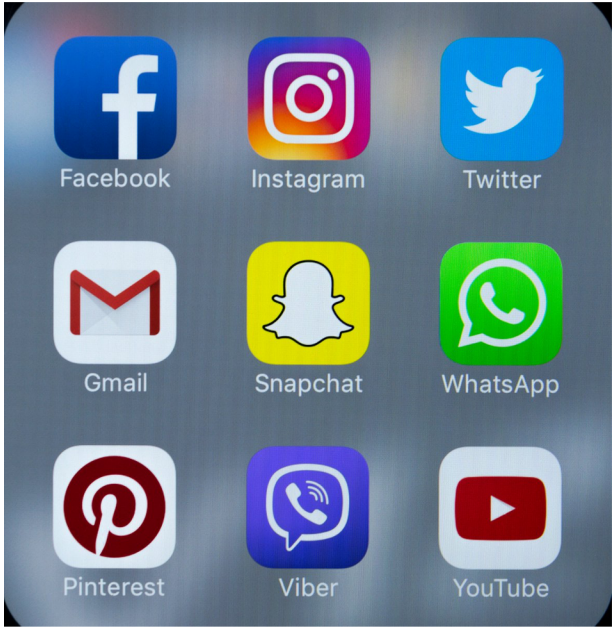
- Stop thinking swsec is only testing!
- Stop thinking swsec is a developers responsibilities

- Start sharing security bugs and fix it asap
- Start thinking that everyone is responsible for security

- OWASP SAMM Assessment and 5D Framework are standards that allows you to create a Software Security program

- **Culture! is the key to measure your Maturity Level for developing secure Software.**

Everyone loves web applications!



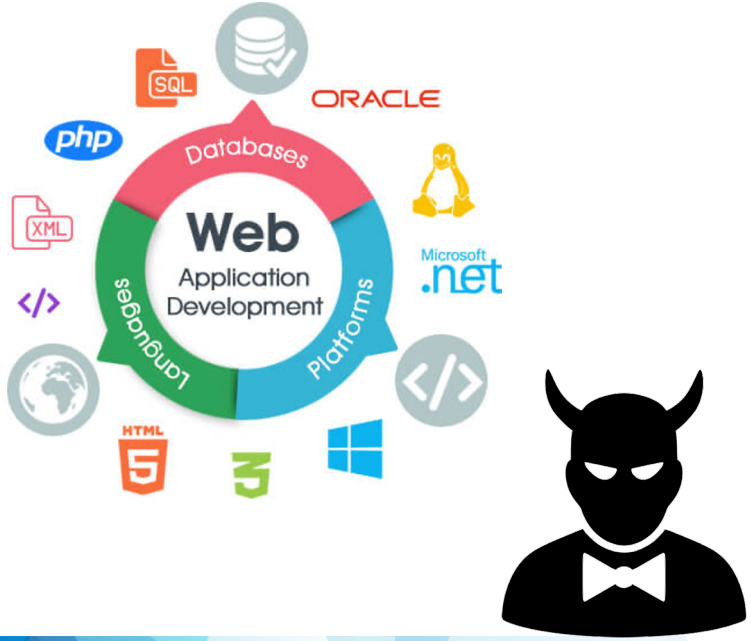
I know all of you use some of them :-)

Fantastic tools, offering personalized access to data and functionality

Primary targets for attackers!

- Link between digital and real life
- Access to private data
- Use of paid / financial services

Web application (in)security



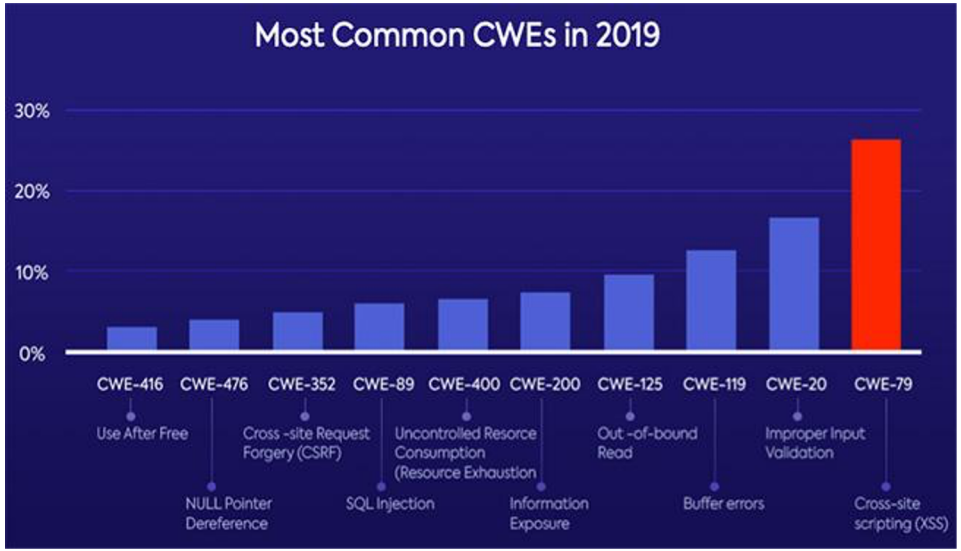
Web applications are **hard to secure**

Complex architecture, leading to a large attack surface

Insecure practices, coming from the pervasive use of scripting languages

Tons of different technologies

Web application (in)security



Client-side security mechanisms

Browsers offer **integrated security mechanisms** for web applications

- Great idea! Everyone uses a web browser!
- Independent from specific web technologies
- Often declarative in nature

Popular examples are based on **security headers**:

- Cookie security attributes
- HTTP Strict Transport Security (HSTS)
- Content Security Policy (CSP)

Client-side security in theory

You are the l33t developer of https://www.super-secure.com

- You are aware that the precious auth cookie must not be stolen, so you mark it as **HttpOnly** and **Secure**
- You know the dangers of XSS, so you set **CSP** to a very restrictive policy like script-src ‘self’; object-src ‘none’; default-src https:
- You deploy **HSTS** with max-age=31536000; includeSubDomains to force the use of HTTPS on the whole site
- You set **X-Frame-Options** to DENY to prevent framing on untrusted sites

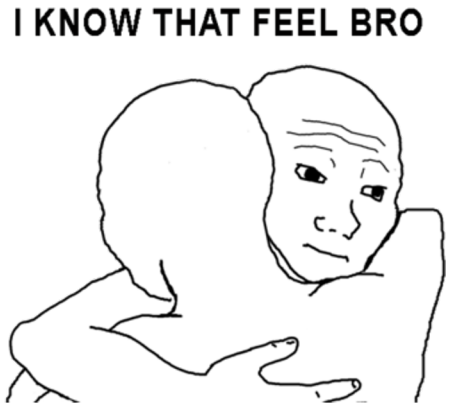
Client-side security in practice

Unfortunately, not so many l33t developers around! No offense meant :-)

- Security attributes unset on session cookies
- CSP configured to allow the execution of inline scripts
- Limited deployment of HSTS in the wild

What about sites which **apparently** enforce strong security?

- Are they really secure in practice?
- A new source of insecurity: **inconsistencies**



Inconsistencies

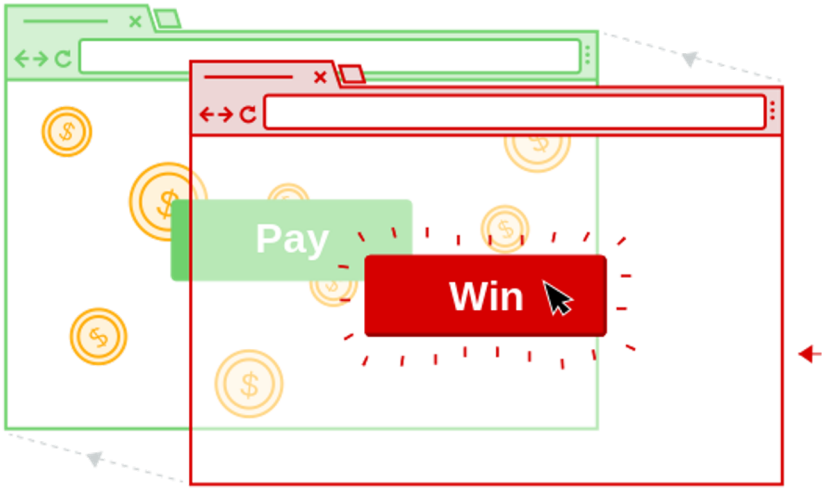
Inconsistencies arise when a **secure** use of a protection mechanism may be voided by an **insecure** use of the same protection mechanism (or a related one)

Inconsistencies might give a **false sense of security!**

- Hey, these guys are using an impressively strict CSP!
- Wow, someone is really using the Secure attribute in the wild!
- … uh oh ... ehr, better to take a second look…

In this talk I discuss different flavors of inconsistencies from two recent papers

Clickjacking



Also known as **UI redressing**

- Abuse CSS to put a transparent frame towards a target site on top of a decoy site
- Used to be prevented by means of JS-based **frame busting**
- Now better prevented by means of **security headers**

Protecting against clickjacking

X-Frame-Options (XFO)

Three possible configurations:

- SAMEORIGIN
- DENY
- ALLOW-FROM url

Deprecated: introduced in browsers before a proper standardization

Double framing attacks

Content Security Policy (CSP)

Whitelist-based defense through the frame-ancestors directive

Full expressive power of CSP

Applies to all ancestors of the frame: no double framing anymore!

Modern alternative to XFO: browsers ignore XFO when CSP is present

Consistent protection

Consistency: all clients have the same level of protection (ideal goal)

Problem: different clients might speak different languages…

- IE does not support CSP
- Chrome lacks ALLOW-FROM
- Weird behavior in header parsing complicates things

10 browsers = 6 semantics...

Browser	CSP	ALLOW-FROM	Multiple Headers	Header Parsing	Double Framing
Chrome	✓	✗	✓	✓	✓
Chrome for Android	✓	✗	✓	✓	✓
Edge	✓	✓	✗	✗	✗
Firefox	✓	✓	✓	✓	✓
Internet Explorer	✗	✓	✗	✗	✗
Opera Mini	✗	✗	✗	✗	✓
Safari	✓	✗	✓	✓	✓
Safari for iOS	✓	✗	✓	✓	✓
Samsung Internet	✓	✗	✓	✓	✓
UC Browser	✓	✗	✓	✓	✗

Table 3: Framing control semantics of popular browsers

Inconsistencies in the wild

Collected data from Tranco Top 10k (Q4 2019)

- ~17,600 policies, including 1,800 inconsistent policies (~10%)
- Not all inconsistencies are equally dangerous!

CSP	XFO	Dangerous?
*.foo.com	DENY	Unlikely
*.foo.com	(absent)	Yes
‘self’	SAMEORIGIN, DENY	Yes

Inconsistencies in the wild

Unfortunately, a significant number of inconsistencies lead to **insecurity**

- 64% of the inconsistent policies allow **arbitrary framing** in at least one client
- Good news: reduction to 16% if we focus on CSP-enabled clients alone
- Bad news: dropping support for ALLOW-FROM in Firefox had a major impact

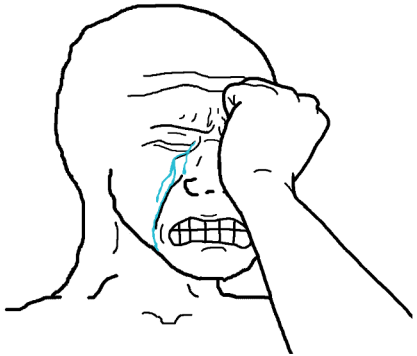
Additional findings:

- 88% of the collected policies use XFO alone
- Only 8% of the policies use XFO and CSP together (54% are inconsistent)

Inconsistent policies

Inconsistent use of client-side security mechanisms can **void protection!**

- Cookies occasionally set without the Secure attribute
- CSP sometimes configured with ‘unsafe-inline’
- HSTS with max-age set to 0 in some pages



This is a **general problem** of client-side security mechanisms

- Security headers are too fine-grained!
- Different pages in the same security boundary may introduce **conflicts**

More inconsistencies in the wild

Large-scale measurement: deep crawling of Tranco Top 15k (Q1 2020)

9% of sites use cookie security attributes inconsistently (4% of all cookies)

- Most inconsistently used attribute: SameSite (15% of cookies using it)

46% of the origins deploying a safe CSP make an inconsistent use of CSP

- Key reason: missing CSP on some pages (81%)

HSTS inconsistently used on 16% of the origins and 81% of the sites

Case study: Cookie leakage

We measured the impact of HSTS inconsistencies on (non-Secure) cookies:

- 17 sites have host-only cookies from a host vulnerable to HSTS deactivation
- 1,254 sites have domain cookies, but do not activate includeSubDomains
- 54 sites have domain cookies, but a subdomain vulnerable to HSTS deactivation

Roughly, 33% of the sites activating HSTS may leak cookies in clear!

Vulnerable sites include alipay.com, taobao.com and wired.com

Countermeasures

Yes, we also studied how to prevent these issues

- Clickjacking case: relatively easy, designed and implemented security proxy
- General case: much harder, Origin Policy is great but does not suffice
- Proposed Site Policy as a better defense mechanism (prototype stage)

Full details in the papers:

- A Tale of Two Headers: A Formal Analysis of Inconsistent Click-Jacking Protection on the Web (USENIX Security 2020)
- Reining in the Web's inconsistencies with Site Policy (NDSS 2021)

Take-away messages

Inconsistencies might give a **false sense of security!**

- For practitioners: client-side security mechanisms are deceptively simple
- For researchers: reasoning on client-side security is surprisingly hard

Concrete problems with existing web standards:

- XFO is underspecified and created a messy state of affairs
- HSTS is just too hard to configure as intended
- We probably need better ways to express client-side security policies...

Ongoing work

The **Security Lottery**: how much is security affected by the client?

- User agent, including its version
- Geolocation: VPNs, proxy servers, ...
- Client configuration and state: language, private mode, ...
- Non-deterministic factors: time of the day, DNS resolution, …

We have a lot of fun in breaking and fixing the Web! **Do you want to join us?**

Join us!

Generally open to discuss research ideas and possible collaborations

Upcoming Virtual OWASP Italy Day

- April 28th over Zoom: one-day, informal event (free of charge)
- Call for presentations is available online (deadline on March 31st) https://owasp.org/www-chapter-italy/cfp/owasp-day-210428
- Registration opening on March 31st

Support OWASP with technical documentation and open-source projects

Q&A

Stefano Calzavara
stefano.calzavara@unive.it
https://www.dais.unive.it/~calzavara

[owasp-italy@owasp.org for OWASP-related stuff]

Matteo Meucci
matteo.meucci@owasp.org
https://owasp.org/www-chapter-italy/

Social Links

- Meetup
- OWASP-Italy Google group
- OWASP-Italy Linkedin
- OWASP-Italy Twitter
- Slack chapter-italy channel

Vi aspettiamo al prossimo OWASP-Italy day il 28 Aprile 2021!