# OWASP Application Security Verification Standard

## November 2018 Update

# Jim

# Application Security

OWASP

Open Web Application Security Project

The Open Web Application Security Project (OWASP) is a **501(c)(3) worldwide not-for-profit charitable organization** focused on improving the security of software.

Our mission is to **make software security visible**, so that individuals and organizations worldwide can make informed decisions about true software security risks.

Everyone is free to participate in OWASP and all of our materials are **available under a free and open software license**. You'll find everything about OWASP linked from our wiki and current information on our OWASP Blog.

https://owasp.org

# The OWASP Top Ten – The Good

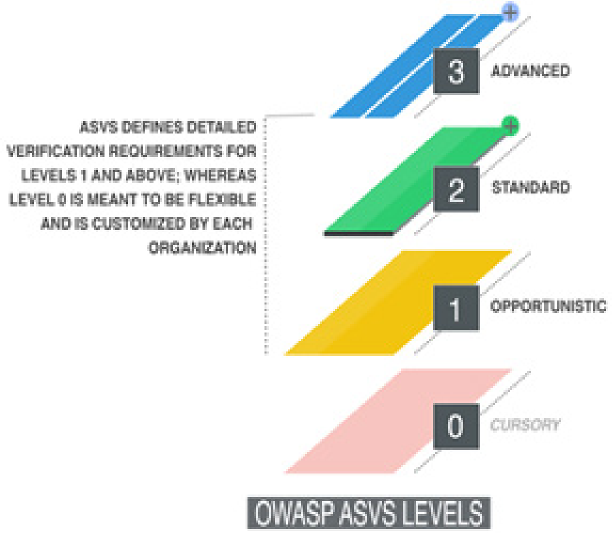
- Project members include a **variety of security experts from around the world** who have shared their expertise to produce this list.
  - Andrew van der Stock
  - Neil Smithline
  - Torsten Gigler
  - Brian Glas
- Significant public comments and conversation on Top Ten 2017 choices https://github.com/OWASP/Top10/tree/master/2017
- Frequently cited application security awareness document

# The OWASP Top Ten- Struggles

- The OWASP Ten has a history of vendor shenanigans that dates back a decade.
- The OWASP Top 10 list is meant to **spread awareness** regarding Web Security issues. It is not a standard. I'm looking at you *PCI-DSS and others who incorrectly list it as so.*
- The OWASP Top Ten is not a comprehensive list of web security risks.
- The OWASP Top Ten does not go into detail about how to fix or prevent these issues.

# So what standard can we use for web applications and webservice security?

# Application Security Verification Standard 3.0.1


3
ADVANCED
2
STANDARD
1
OPPORTUNISTIC
0
CURSORY
ASVS DEFINES DETAILED VERIFICATION REQUIREMENTS FOR LEVELS 1 AND ABOVE; WHEREAS LEVEL 0 IS MEANT TO BE FLEXIBLE AND IS CUSTOMIZED BY EACH ORGANIZATION
OWASP ASVS LEVELS

- First application security standard **by developers, for developers**
- Defines three risk levels with **more than 150 controls**
- Similar but not the same: ISO 27034
- Useful standard to **ensure your teams are doing complete assessment**
- Useful standard to **help drive developer secure coding practices**

# Application Security Verification Standard 3.0.1

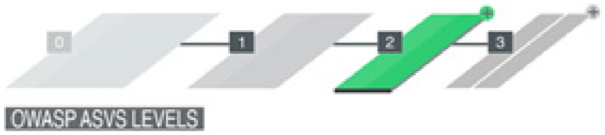
| # | Description | 1 | 2 | 3 | Since |
|---|---|---|---|---|---|
| 7.2 | Verify that all cryptographic modules fail securely, and errors are handled in a way that does not enable oracle padding. | ✓ | ✓ | ✓ | 1.0 |
| 7.6 | Verify that all random numbers, random file names, random GUIDs, and random strings are generated using the cryptographic module's approved random number generator when these random values are intended to be not guessable by an attacker. |  | ✓ | ✓ | 1.0 |
| 7.7 | Verify that cryptographic algorithms used by the application have been validated against FIPS 140-2 or an equivalent standard. | ✓ | ✓ | ✓ | 1.0 |
| 7.8 | Verify that cryptographic modules operate in their approved mode according to their published security policies. |  |  | ✓ | 1.0 |
| 7.9 | Verify that there is an explicit policy for how cryptographic keys are managed (e.g., generated, distributed, revoked, and expired). Verify that this key lifecycle is properly enforced. |  | ✓ | ✓ | 1.0 |
| 7.11 | Verify that all consumers of cryptographic services do not have direct access to key material. Isolate cryptographic processes, including master secrets and consider the use of a virtualized or physical hardware key vault (HSM). |  |  | ✓ | 3.0.1 |

# Application Security Verification Standard 3.0.1
# **Level 1:** Opportunistic

0
1
2
3
OWASP ASVS LEVELS

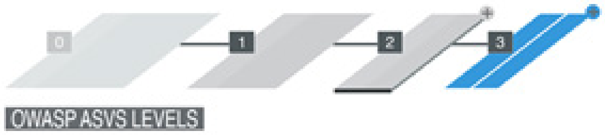
- Minimum required for all apps
- Mostly fully testable
- Mostly automatable
- Straight forward developer fixes
- Not enough for high risk apps

# Application Security Verification Standard 3.0.1
# **Level 2:** Standard



- Suitable for sensitive data
- About 75% testable
- Somewhat test automatable
- Moderate to complex developer security challenges

# Application Security Verification Standard 3.0.1
# **Level 3:** Advanced



- Suitable for critical apps
- Mostly testable, but many more manual verifications required
- Not amenable to automation
- Significant developer challenges

# How to fork and use ASVS

## Building the Application Security Verification Standard into your SDLC

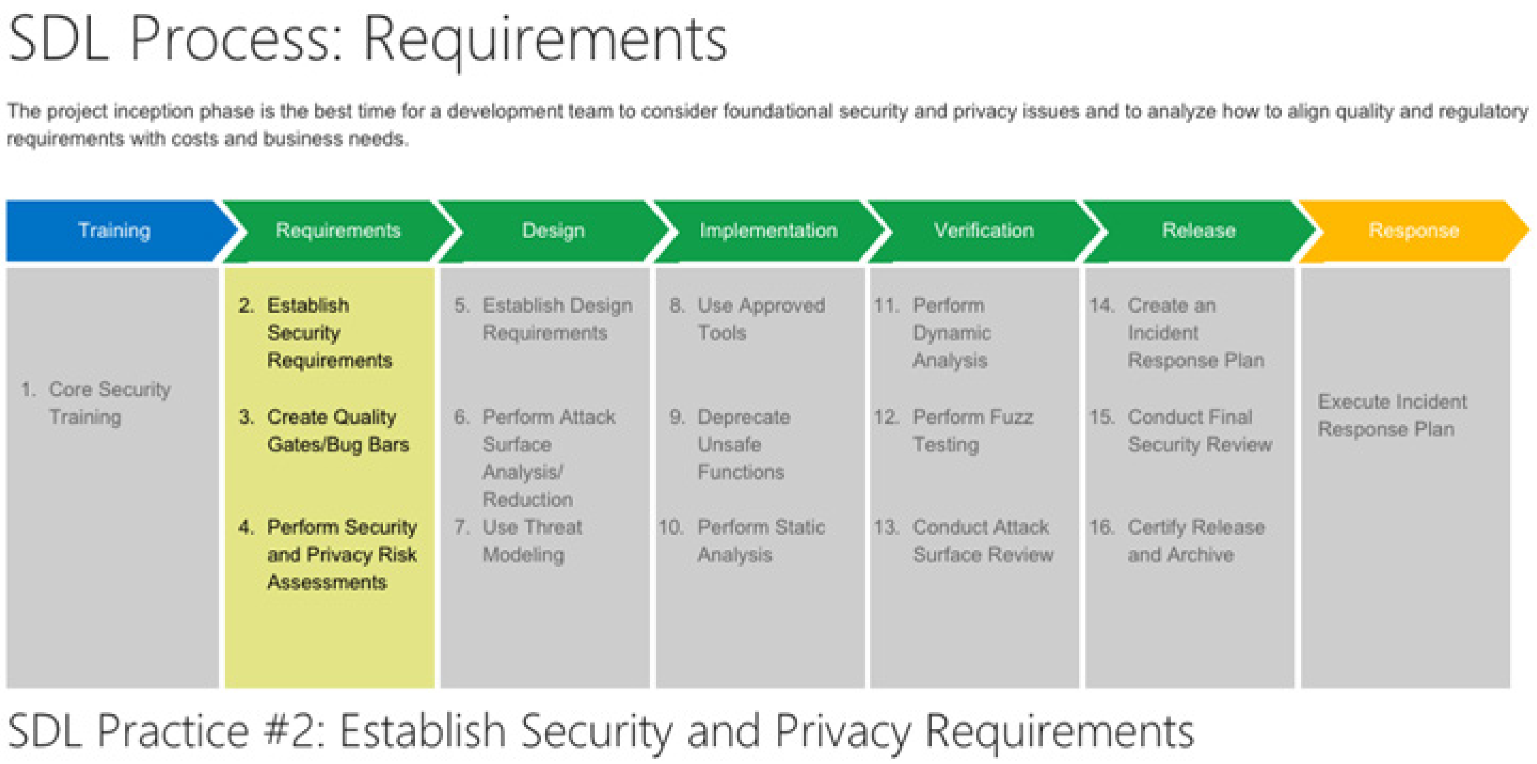
# ASVS Anti-Patterns

- Security leaders mailing the ASVS document to the development teams tealling them "security says you have to follow this now. Good luck!"
- Avoid engaging developers on ASVS items before making it policy
- Using ASVS out of the box without customizing it for your organization
- Setting ASVS as a standard in way where it's never used, never read or never considered. But given to customers.

# ASVS Effective Adoption

- The goal of ASVS adoption is for developers to actively use it in their development and architectural work every day.
- Work with developers early on in forking ASVS.
- Let developers lead in version 1 as to what requirements will be accepted by the team.
- Like any complex legislation, just get it in there and modify it over time after version 1.

# Microsoft SDL on Defining Requirements



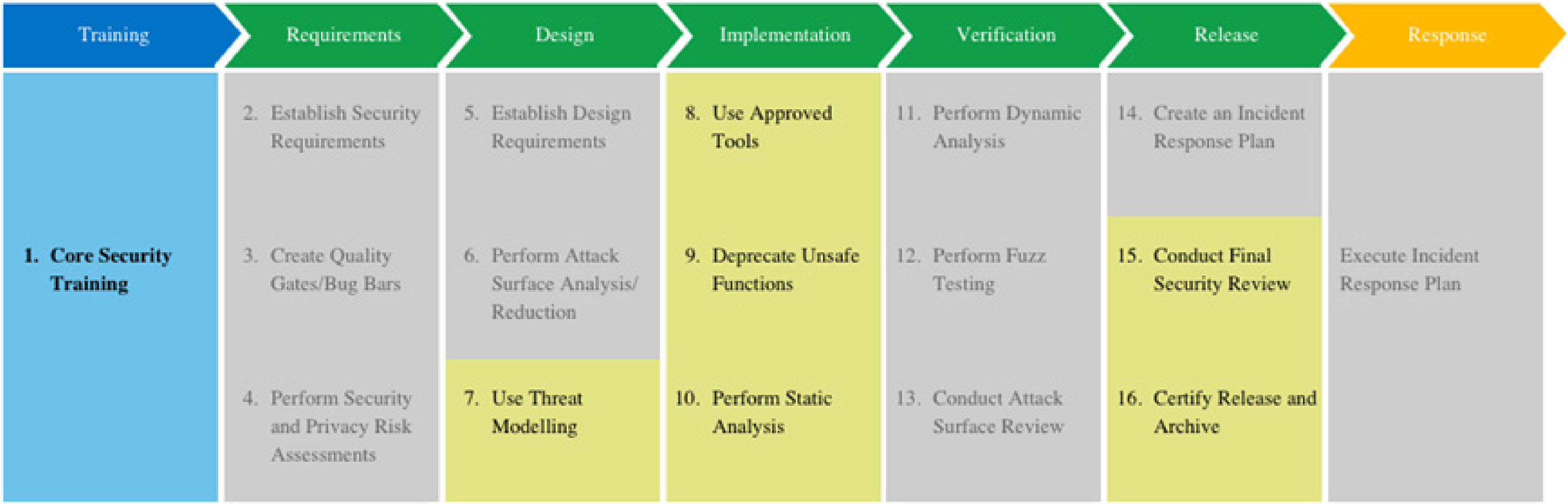
## SDL Process: Requirements

The project inception phase is the best time for a development team to consider foundational security and privacy issues and to analyze how to align quality and regulatory requirements with costs and business needs.

| Training | Requirements | Design | Implementation | Verification | Release | Response |
|---|---|---|---|---|---|---|
| 1. Core Security Training | 2. Establish Security Requirements<br>3. Create Quality Gates/Bug Bars<br>4. Perform Security and Privacy Risk Assessments | 5. Establish Design Requirements<br>6. Perform Attack Surface Analysis/ Reduction<br>7. Use Threat Modeling | 8. Use Approved Tools<br>9. Deprecate Unsafe Functions<br>10. Perform Static Analysis | 11. Perform Dynamic Analysis<br>12. Perform Fuzz Testing<br>13. Conduct Attack Surface Review | 14. Create an Incident Response Plan<br>15. Conduct Final Security Review<br>16. Certify Release and Archive | Execute Incident Response Plan |

SDL Practice #2: Establish Security and Privacy Requirements

# Microsoft SDL on Using Requirements



| Training | Requirements | Design | Implementation | Verification | Release | Response |
|---|---|---|---|---|---|---|
| | 2. Establish Security Requirements | 5. Establish Design Requirements | **8. Use Approved Tools** | 11. Perform Dynamic Analysis | 14. Create an Incident Response Plan | |
| **1. Core Security Training** | 3. Create Quality Gates/Bug Bars | 6. Perform Attack Surface Analysis/ Reduction | **9. Deprecate Unsafe Functions** | 12. Perform Fuzz Testing | **15. Conduct Final Security Review** | Execute Incident Response Plan |
| | 4. Perform Security and Privacy Risk Assessments | **7. Use Threat Modelling** | **10. Perform Static Analysis** | 13. Conduct Attack Surface Review | **16. Certify Release and Archive** | |

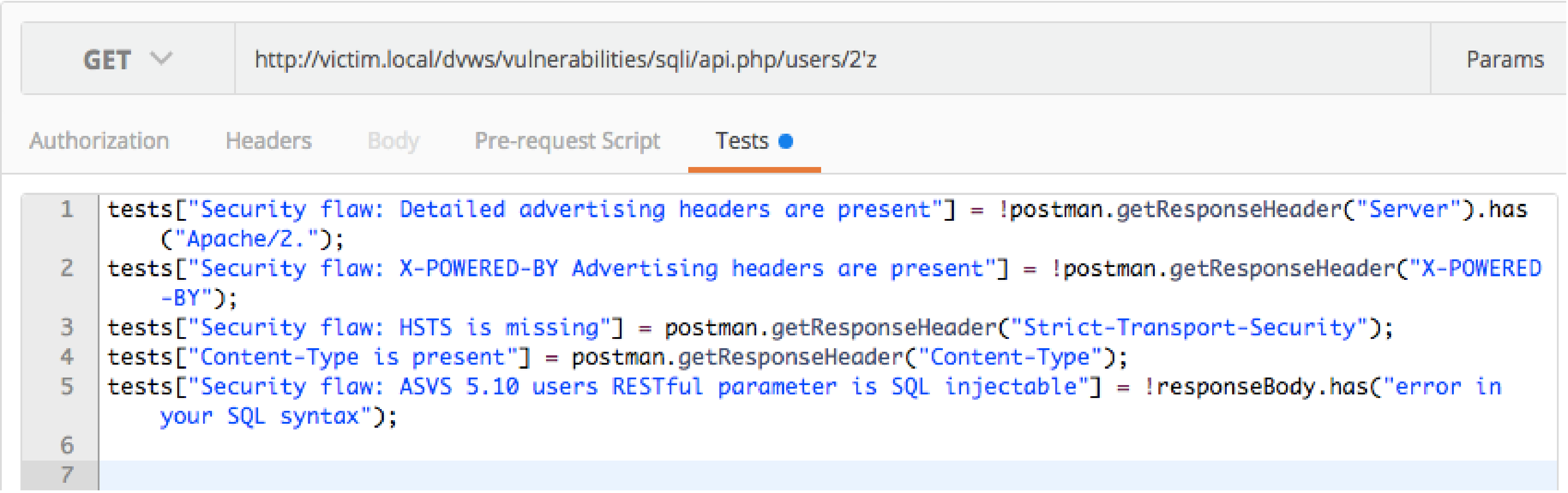
# Writing Unit Tests Using ASVS

Write unit tests to validate your application each and every build

Allows penetration testers to concentrate on difficult to automate tests, such as business logic flaws, access control issues, and things you forgot in the unit tests

# Writing Integration Tests

Integration tests can be written
using Postman, Selenium,
OWASP Zap API



http://victim.local/dvws/vulnerabilities/sqli/api.php/users/2'z

```
tests["Security flaw: Detailed advertising headers are present"] = !postman.getResponseHeader("Server").has("Apache/2.");
tests["Security flaw: X-POWERED-BY Advertising headers are present"] = !postman.getResponseHeader("X-POWERED-BY");
tests["Security flaw: HSTS is missing"] = postman.getResponseHeader("Strict-Transport-Security");
tests["Content-Type is present"] = postman.getResponseHeader("Content-Type");
tests["Security flaw: ASVS 5.10 users RESTful parameter is SQL injectable"] = !responseBody.has("error in your SQL syntax");


```

# What will be new in 4.0?

- Moving all of authentication requirements inline with NIST 800-63-3
- Changing session requirements to acknowledge new world of JWT's and stateless mechanisms
- Removing mobile requirements due to MASVS
- Lots of small edits and clarifications on requirement language
- Colapsing many requirements that duplicate concepts
- Triaging hundreds of comments from the field
- Moving to markdown for primary text
- Moar GDPR
- https://github.com/OWASP/ASVS/tree/master/3.1/en

jim@manicode.com