



OWASP
Open Web Application
Security Project

Vulnerabilities: Is not fault of
Developer, it is of Delivery
process S-SDLC.

Presentador

Gustavo Nieves Arreaza

Bag: Systems Engineer

IT Security Consultant

CEO in SeguridadAplicativos Black Cap

CCNA and CCIA Cisco

Programming courses in .Net and Java (IBM, Microsoft)

Website: www.seguridadaplicativos.com

Linkedin: <https://cl.linkedin.com/in/gustavo-nieves-arreaza-0548a527>



OWASP
Open Web Application
Security Project



Results



OWASP
Open Web Application
Security Project

When asked 12,000 security professionals to name what is the main security threat to their organization, 69% said vulnerabilities in the application layer * - however, less than 10% ensure that all their critical applications Business are reviewed for safety before and during production.

Dropbox: 68 million accounts

Ashley Madison: Over 30 million

Linkedin: 164 million

Yahoo + 500 million

Panama Papers

<https://tcinet.ru/en/press-centre/technology-news/3863/>

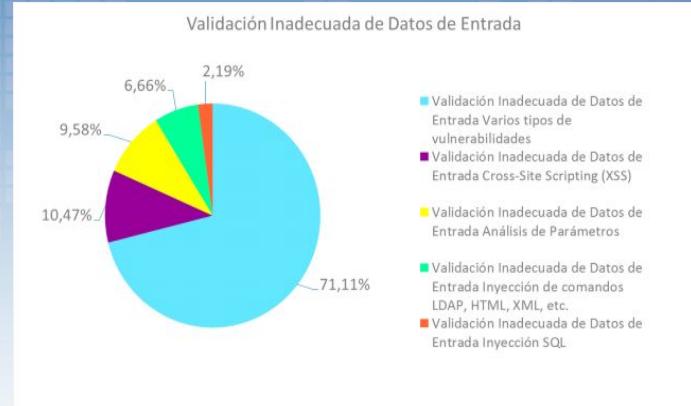
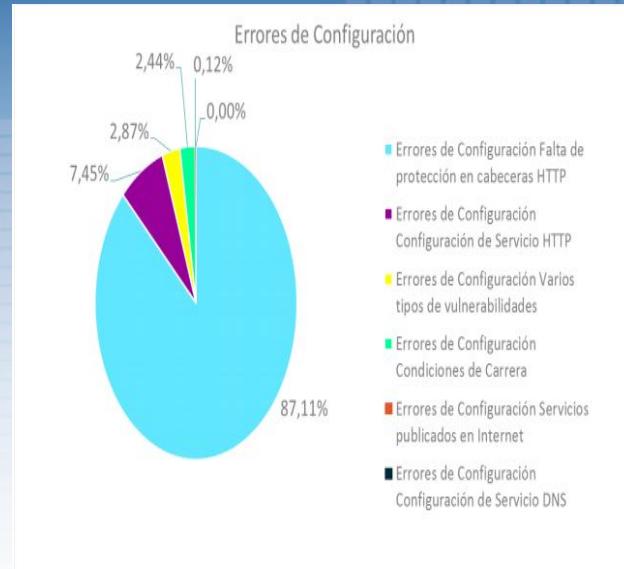
Statistics

Configuration

AThose like, files of free access or servers web on updated

Cryptographic:

Lack of SSL Certificates, or certificates with weak encryption



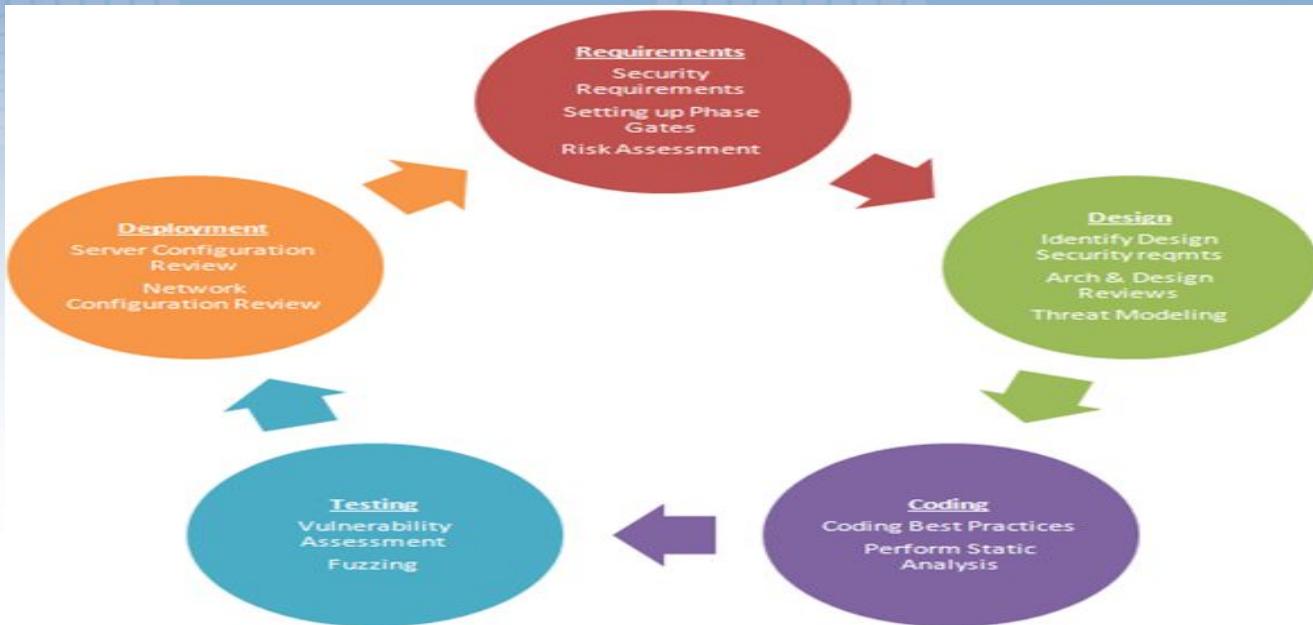
Input chains

Those that do not validate data revenues, nor upload files

S-SDLC(Systems Development Life Cycle)



OWASP
Open Web Application
Security Project



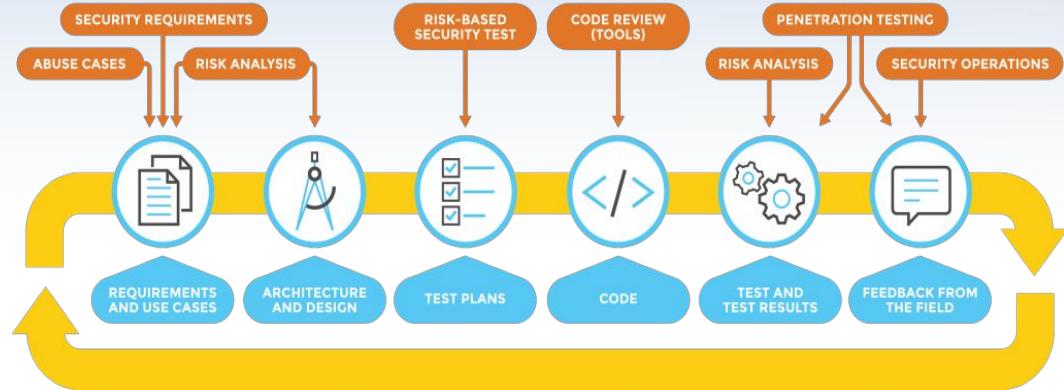
S-SDLC Model more use Security touchpoint



OWASP
Open Web Application
Security Project

Touchpoints are a mix of destructive and constructive activities. Destructive activities are about attacks, exploits, and breaking software. These kinds of things are represented by the black hat (offense). Constructive activities are about design, defense, and functionality. These are represented by the white hat (defense). Both **Software Security Touchpoints** hats are necessary

- Code review
- Architectural risk analysis
- Penetration testing
- Risk-based security tests
- Abuse cases
- Security requirements
- Security operations



S-SDLC more use MS SDL



OWASP
Open Web Application
Security Project

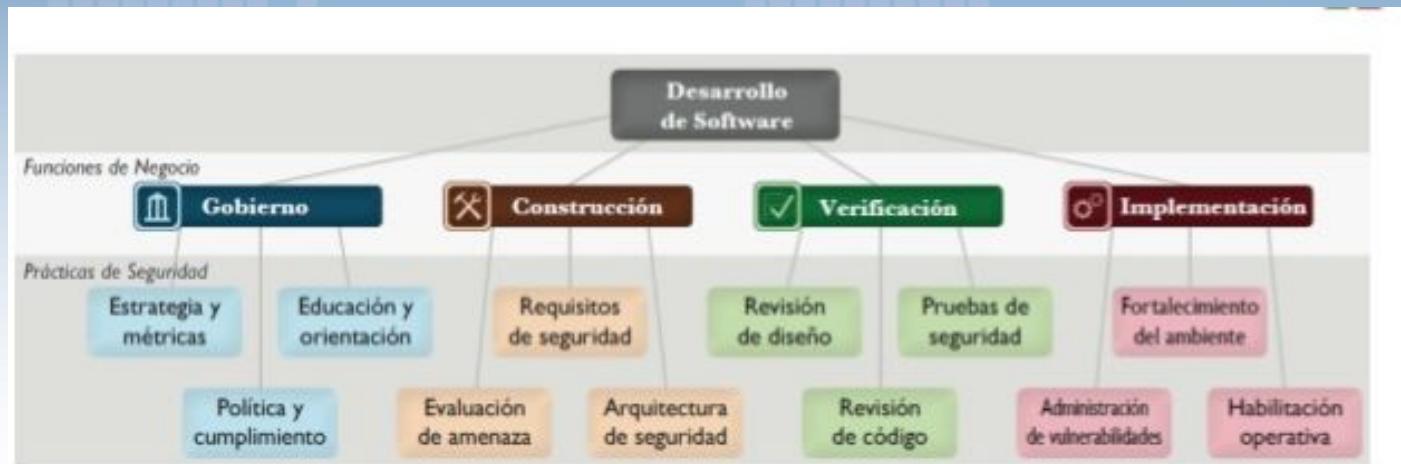
Ciclo de vida de MS Security Development (MS SDL): Uno de los primeros de su tipo, MS SDL fue propuesto por Microsoft de acuerdo a las fases de un SDLC.

1. TRAINING	. REQUIREMENTS	3. DESIGN	4. IMPLEMENTATION	5. VERIFICATION	6. RELEASE	7. RESPONSE
1. Core Security Training	2. Establish Security Requirements	5. Establish Design Requirements	8. Use Approved Tools	11. Perform Dynamic Analysis	14. Create an Incident Response Plan	Execute Incident Response Plan
	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	
	4. Perform Security and Privacy Risk Assessments	7. Use Threat Modeling	10. Perform Static Analysis	13. Conduct Attack Surface Review	16. Certify Release and Archive	

S-SDLC Owasp SAMM



OWASP
Open Web Application
Security Project



SAMM(Software Assurance Maturity Model)



Training

1. Input Validation
2. Output Encoding
3. Authentication and Password Management (includes secure handling of credentials by external services/scripts)
4. Session Management
5. Access Control
6. Cryptographic Practices
7. Error Handling and Logging
8. Data Protection
9. Communication Security
10. System Configuration
11. Database Security
12. File Management
13. Memory Management
14. General Coding Practices



Training



OWASP
Open Web Application
Security Project

Key references

- Stopping XSS in your web application: OWASP

XSS (Cross Site Scripting) Prevention Cheat Sheet

- General Information about injection:

Top 10 2013-A1-Injection

Key tools

OWASP Java Encoder Project

Microsoft .NET AntiXSS Library

OWASP ESAPI

OWASP Encoder Comparison Reference Project

Proactive tools:
to know what
use to remedy,
and how with
need test

Web Goat:
Practice how to
exploit
vulnerabilities



Desing

ASVS
(Application Security Verification Standard)

V3: Session Management Verification Requirements

Control objective

One of the core components of any web-based application is the mechanism by which it controls and maintains the state for a user interacting with it. This is referred to this as Session Management and is defined as the set of all controls governing state-full interaction between a user and the web-based application.

Ensure that a verified application satisfies the following high level session management requirements:

- Sessions are unique to each individual and cannot be guessed or shared
- Sessions are invalidated when no longer required and timed out during periods of inactivity.

Requirements

#	Description	1	2	3	Since
3.1	Verify that there is no custom session manager, or that the custom session manager is resistant against all common session management attacks.	✓	✓	✓	1.0
3.2	Verify that sessions are invalidated when the user logs out.	✓	✓	✓	1.0
3.3	Verify that sessions timeout after a specified period of inactivity.	✓	✓	✓	1.0

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
Likelihood				



OWASP
Open Web Application Security Project

Threat

Modeling:

Know what we have

-Name of application: the name of the application.

-Version of the application: the version of the application.

-Description - A high-level description of the application.

-Document Owner - The owner of the threat modeling document.

Risk Ranking: to establish criticality Of the vulnerabilities

Development

ESAPI (The
OWASP
Enterprise
Security API)

Naming conventions such as this are not part of ESAPI but are good practice

```
$clean = array(); //this is local in scope
$clean_sql = array(); //this is local in scope
$clean['id'] = ESAPI::getValidator()->getValidInput( ... );
$clean_sql['id'] = ESAPI::getEncoder()->encodeForSQL( new MySQLCodec(), $clean['id'] );
```

Step 1 Step 2 This is also an ESAPI control

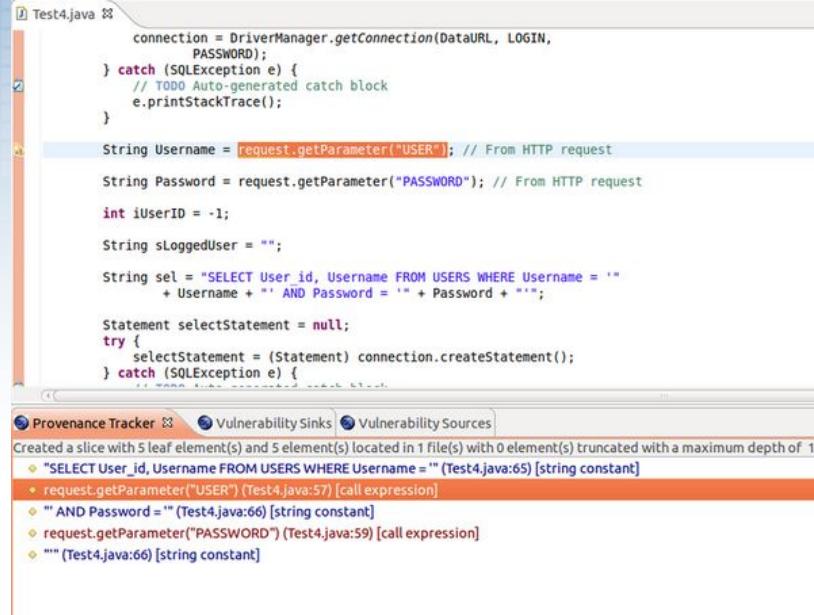
Response Headers

- HTTP Strict Transport Security (HSTS)
- Public Key Pinning Extension for HTTP (HPKP)
- X-Frame-Options
- X-XSS-Protection
- X-Content-Type-Options
- Content-Security-Policy
- X-Permitted-Cross-Domain-Policies



Open Web Application
Security Project
Static code analysis tool, which allows us to
measure how we are going

Owasp Lapse



The screenshot shows the Owasp Lapse interface with a Java code editor and a Provenance Tracker. The code editor contains a snippet of Java code for a database connection and user authentication. The Provenance Tracker shows the flow of data from user input parameters to the SQL query, highlighting potential security risks.

```
Test4.java 23
connection = DriverManager.getConnection(DataURL, LOGIN,
PASSWORD);
} catch (SQLException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}

String Username = request.getParameter("USER"); // From HTTP request
String Password = request.getParameter("PASSWORD"); // From HTTP request

int iUserID = -1;
String sLoggedUser = "";

String sel = "SELECT User_id, Username FROM USERS WHERE Username = "
+ Username + " AND Password = " + Password + "";

Statement selectStatement = null;
try {
selectStatement = (Statement) connection.createStatement();
} catch (SQLException e) {
```

Provenance Tracker: Created a slice with 5 leaf element(s) and 5 element(s) located in 1 file(s) with 0 element(s) truncated with a maximum depth of 1.

- "SELECT User_id, Username FROM USERS WHERE Username = " (Test4.java:65) [string constant]
- request.getParameter("USER") (Test4.java:57) [call expression]
- " AND Password = " (Test4.java:66) [string constant]
- request.getParameter("PASSWORD") (Test4.java:59) [call expression]
- "" (Test4.java:66) [string constant]

Verification



OWASP
Open Web Application
Security Project

V3: Session Management Verification Requirements

Control objective

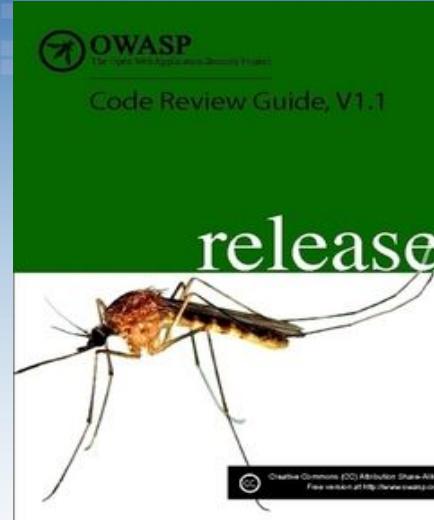
One of the core components of any web-based application is the mechanism by which it controls and maintains the state for a user interacting with it. This is referred to this as Session Management and is defined as the set of all controls governing state-full interaction between a user and the web-based application.

Ensure that a verified application satisfies the following high level session management requirements:

- Sessions are unique to each individual and cannot be guessed or shared
- Sessions are invalidated when no longer required and timed out during periods of inactivity.

Requirements

#	Description	1	2	3	Since
3.1	Verify that there is no custom session manager, or that the custom session manager is resistant against all common session management attacks.	✓	✓	✓	1.0
3.2	Verify that sessions are invalidated when the user logs out.	✓	✓	✓	1.0
3.3	Verify that sessions timeout after a specified period of inactivity.	✓	✓	✓	1.0



that allows only users with administrator privileges to access the content.

Figure A5.11

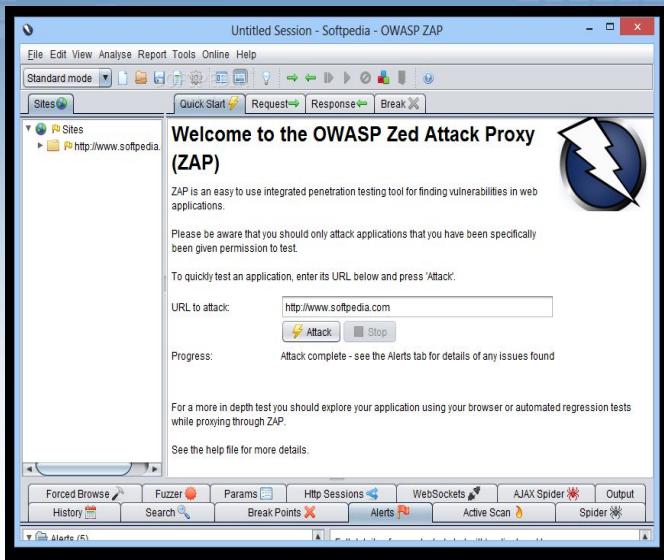
Authorization configuration in IIS

```
<configuration>
  <system.webServer>
    <security>
      <authorization>
        <remove users="*" roles="" verbs="" />
        <add accessType="Allow" users="" roles="Administrators" />
      </authorization>
    </security>
  </system.webServer>
</configuration>
```

Maintence



OWASP
Open Web Application
Security Project



Audit each delivery once a week

Audit the project root separately if it exists

There are websites that try to avoid it, with countermeasures



OWASP
Open Web Application
Security Project

Future of SDLC.....

Secure the house from the ground up

References



OWASP
Open Web Application
Security Project

https://www.owasp.org/index.php/File:OWASP_CRG_BetaReview.docx

https://www.owasp.org/images/3/33/OWASP_Application_Security_Verification_Standard_3.0.1.pdf

https://www.owasp.org/index.php/File:OWASP_CRG_BetaReview.docx

https://www.owasp.org/images/3/33/OWASP_Application_Security_Verification_Standard_3.0.1.pdf

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

https://www.owasp.org/images/0/07/OWASP_Proactive_Controls_v1.pdf