ISACA®
Servire i professionisti dell'IT Governance
Rome Chapter

OWASP Testing Guide v2

La nuova metodologia per l'audit di sicurezza degli applicativi web

(a cura di **Matteo Meucci – CISA, CISSP - INS**)

Agenda:

- OWASP Projects
- The new Testing Guide: goals and deliverables
- The OWASP Testing Framework
- The Testing Methodology: how to test
- The reporting: how to value the risk and write a report
- How the Guide will be useful to the web security industry
- Q&A

36,000+
successful client engagements

1,100+
certifications in 96+ categories

900
consulting and management employees

38
markets across North America, Europe & Asia

15
years in business-centric technology consulting

14
partnerships with top technology leaders

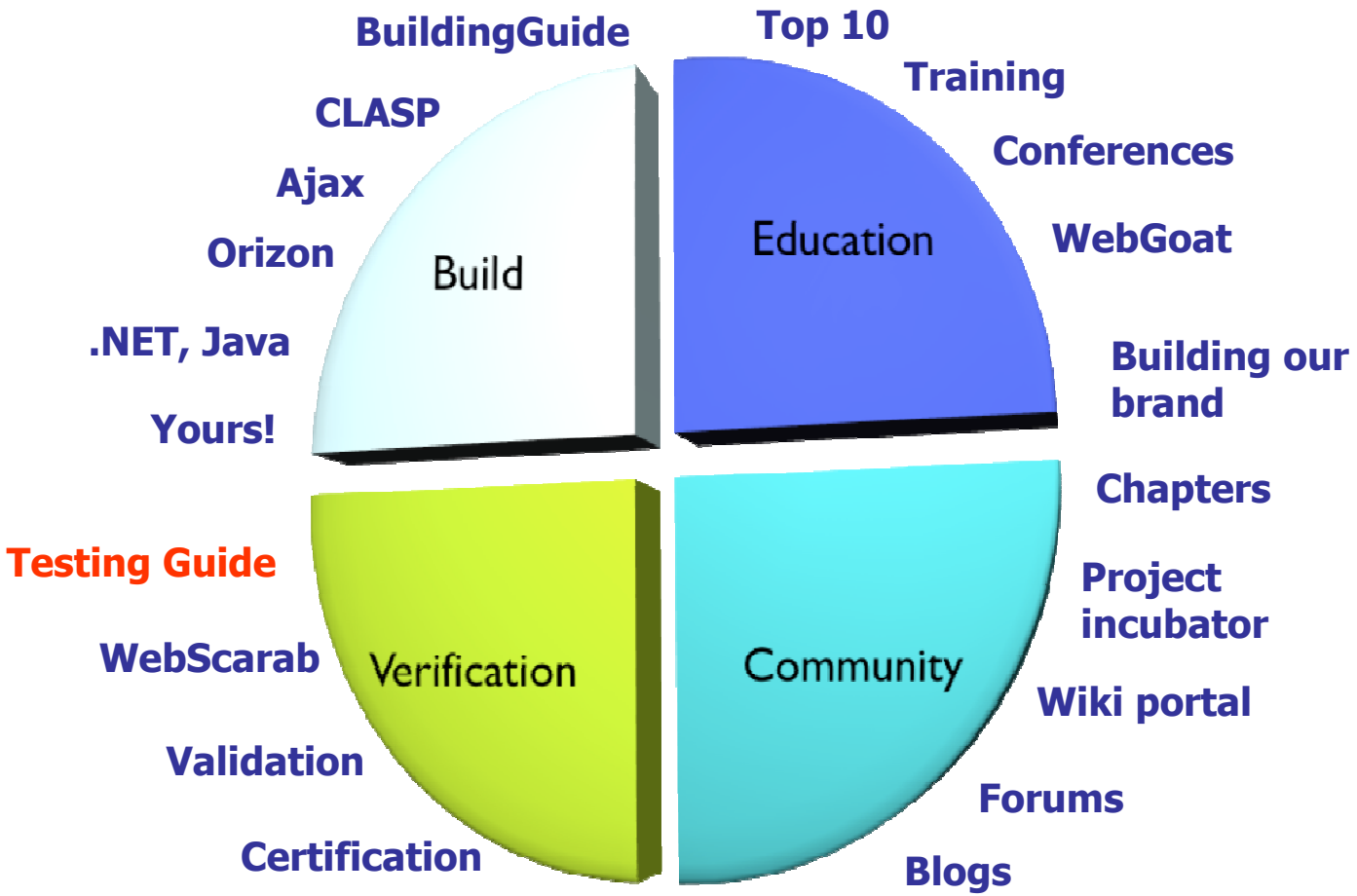
Speaker:

- INS Consultant
- 6+ years on Information Security focusing on Application Security
- OWASP Italy founder and Chair
- OWASP Testing Guide AoC lead

INS:

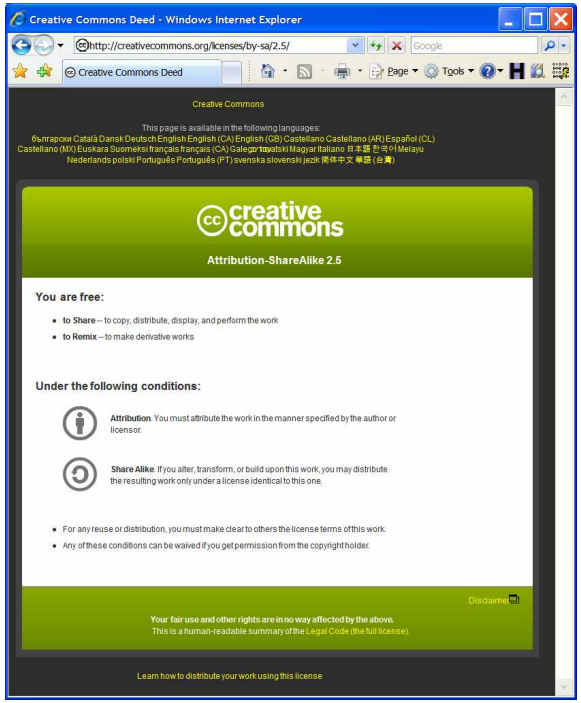
- Focus on aligning technology and operations to business needs
- Multidisciplinary, IT infrastructure-to-application consulting expertise
- 900 employees worldwide, with more than 600 enterprise and service provider clients
- Dedicated quality program with world-class Customer Loyalty Index

- The Open Web Application Security Project (OWASP) is dedicated to finding and fighting the causes of insecure software. The OWASP Foundation is a 501c3 not-for-profit charitable organization that ensures the ongoing availability and support for our work.
- Participation in OWASP is free and open to all.
- Everything here is free and open source.
- Main objective: produce tools, standards and documentation related on Web Application Security.
- Thousands active members, 82 local chapter in the world
- Millions of hits on www.owasp.org
- Defense Information Systems Agency (DISA) , US Federal Trade Commisson (FTC), VISA, Mastercard, American Express has adopted OWASP in their standards and guidelines


BuildingGuide
CLASP
Ajax
Orizon
.NET, Java
Yours!
Build
Top 10
Training
Conferences
WebGoat
Education
Building our brand
Chapters
Project incubator
Wiki portal
Forums
Blogs
Community
Testing Guide
WebScarab
Validation
Certification
Verification

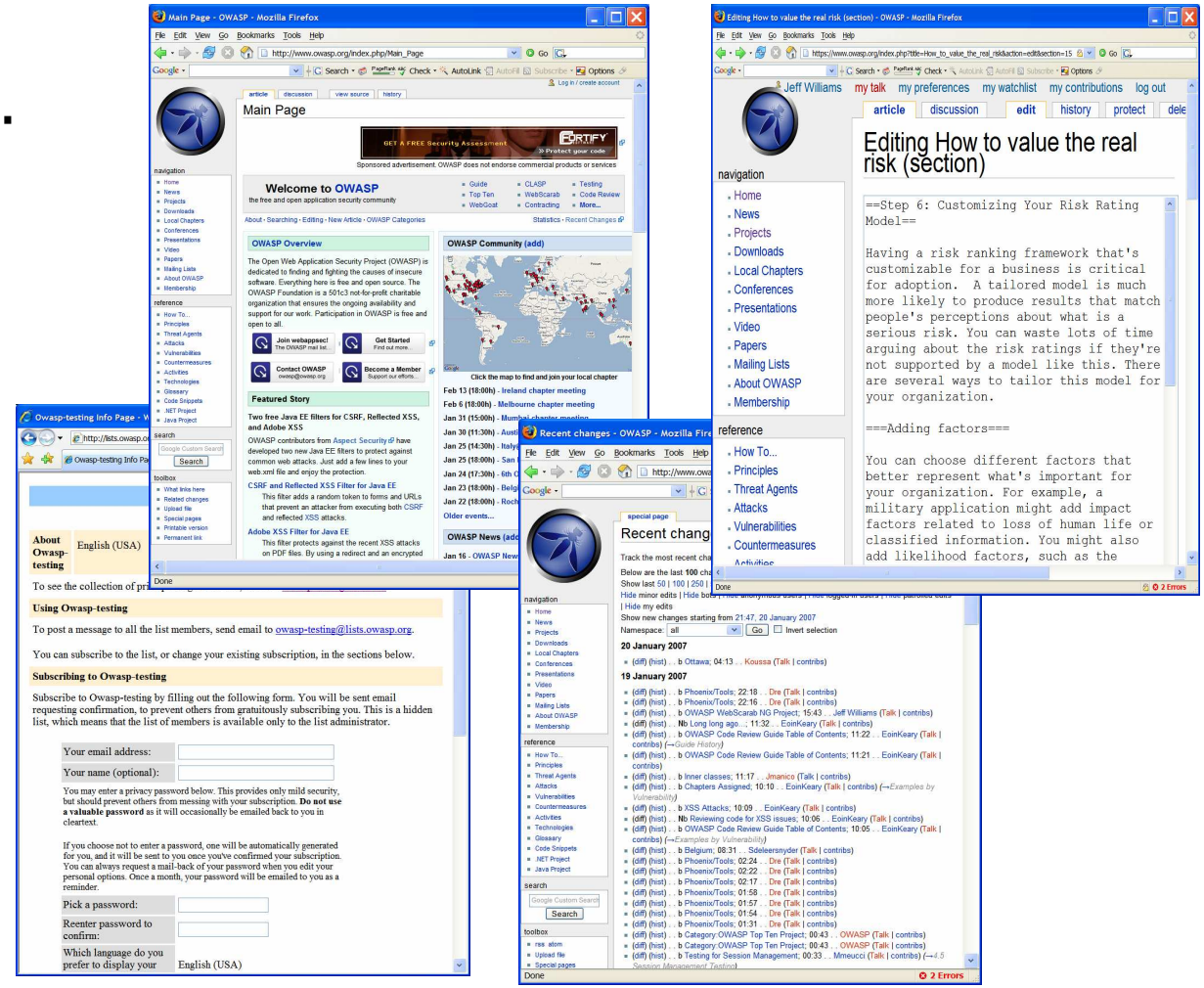
What Is the OWASP Testing Guide?

- Free and open…



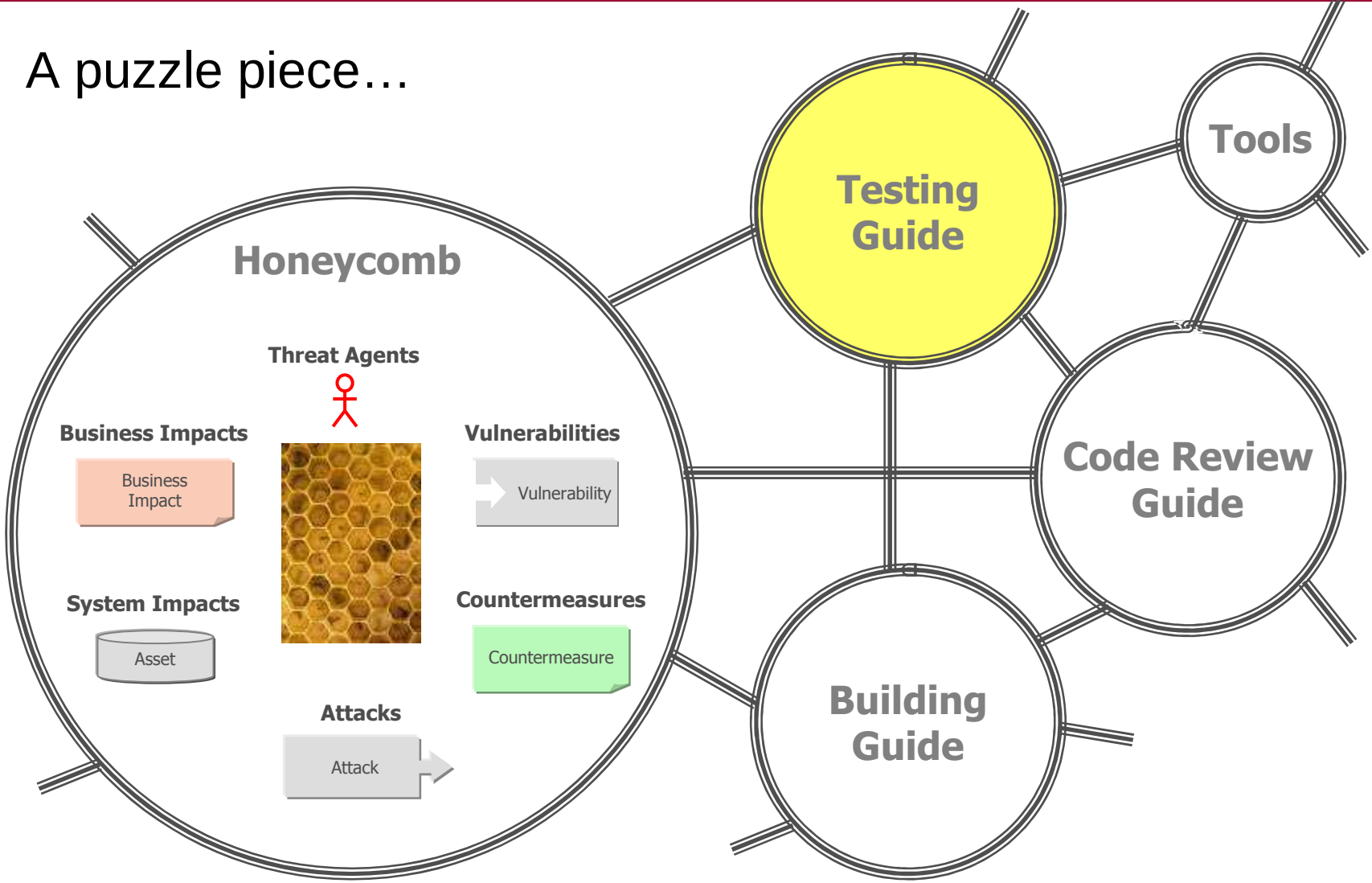
What Is the OWASP Testing Guide?

- A project…



What Is the OWASP Testing Guide?

- A puzzle piece…



OWASP Testing Guide v2: Goals

- Review all the documentation on testing:
  - July 14, 2004
    - "OWASP Web Application Penetration Checklist", Version 1.1
  - December 2004
    - "The OWASP Testing Guide", Version 1.0
- Create a complete new project focused on Web Application Penetration Testing
- Create a reference for application testing
- Describe the new OWASP Methodology
- Describe how to test each control

OWASP Testing Guide v2: Action Plan

Action Plan:

Oct 2006:

- Collect all old docs
- Brainstorming for the Index and template
- Involve major world experts on this field:

* Vicente Aguilera
* Mauro Bregolin
* Tom Brennan
* Gary Burns
* Luca Carettoni
* Dan Cornell
* Mark Curphey
* Daniel Cuthbert
* Sebastien Deleersnyder
* Stephen DeVries
* Stefano Di Paola
* David Endler
* Giorgio Fedon
* Javier Fernández-Sanguino
* Glyn Geoghegan
* Stan Guzik
* Madhura Halasgikar
* Eoin Keary
* David Litchfield
* Andrea Lombardini
* Ralph M. Los
* Claudio Merloni
* Matteo Meucci
* Marco Morana
* Laura Nunez
* Gunter Ollmann
* Antonio Parata
* Yiannis Pavlosoglou
* Carlo Pelliccioni
* Harinath Pudipeddi
* Alberto Revelli
* Mark Roxberry
* Tom Ryan
* Anush Shetty
* Larry Shields
* Dafydd Studdard
* Andrew van der Stock
* Ariel Waissbein
* Jeff Williams

OWASP Testing Guide v2: Action Plan (2)

Action Plan:

Nov 2006:

- Write articles using our Wiki model
- Review articles

Dec 2006:

- Review all the Guide
- Write the Guide in doc format

Jan 2007:

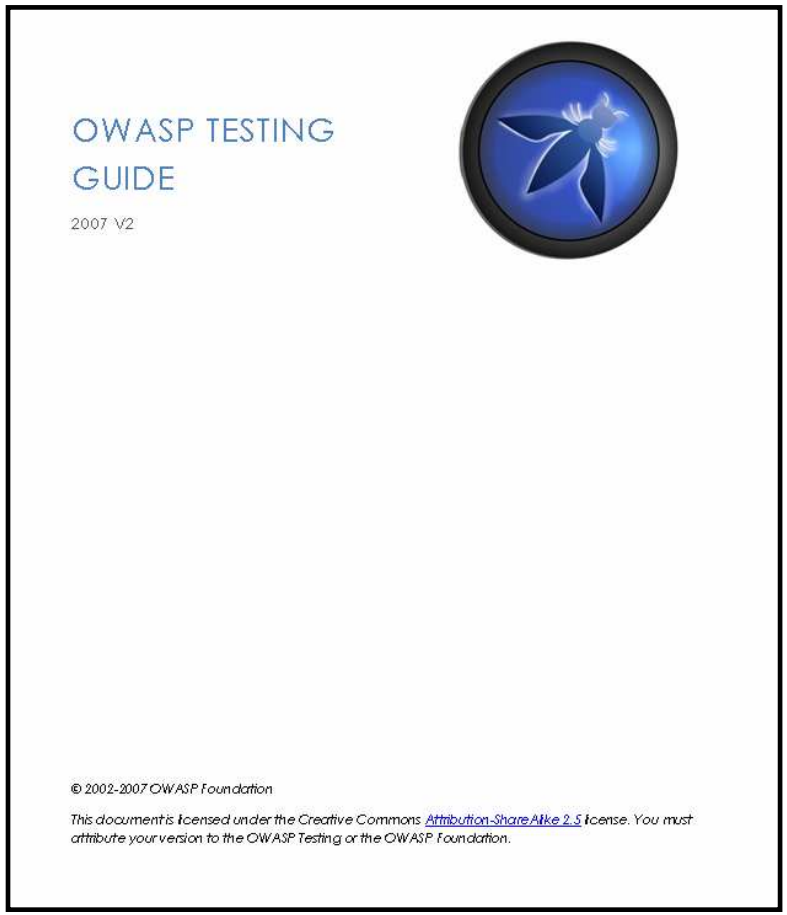
- OWASP Testing Guide Release Candidate 1: 270 pages, 48 tests
- Feedback and review

Feb 2007:

- OWASP Testing Guide v2 will be officially released

Testing Guide v2 RC1: Index

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



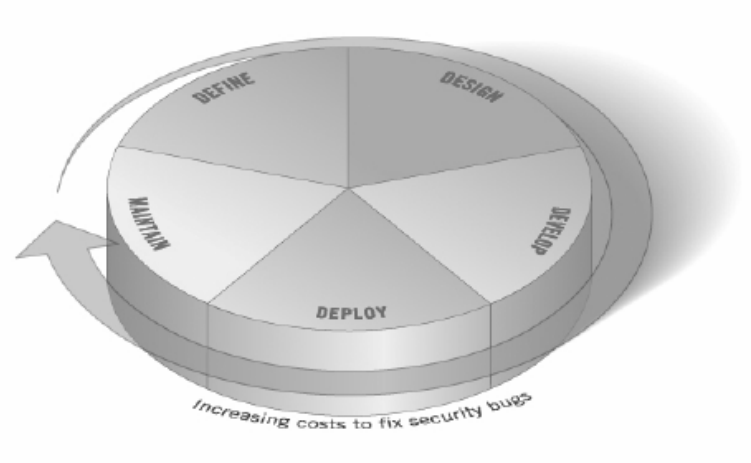
I. Introduction

- The problem of insecure software: companies next challenge
- Why OWASP?
  - "It's impossible to underestimate the importance of having this guide available in a completely free and open way"– *Jeff Williams (OWASP Chair)*
- Principles of Testing: comparing the state of something against a set of criteria defined and complete.
  - We want security testing not be a black art
- Testing Techniques:
  - Manual Inspections & Reviews
  - Threat Modeling
  - Code Review
  - Penetration Testing

II. The OWASP Testing Framework

Phase 1: Before Development Begins

Before application development has started:



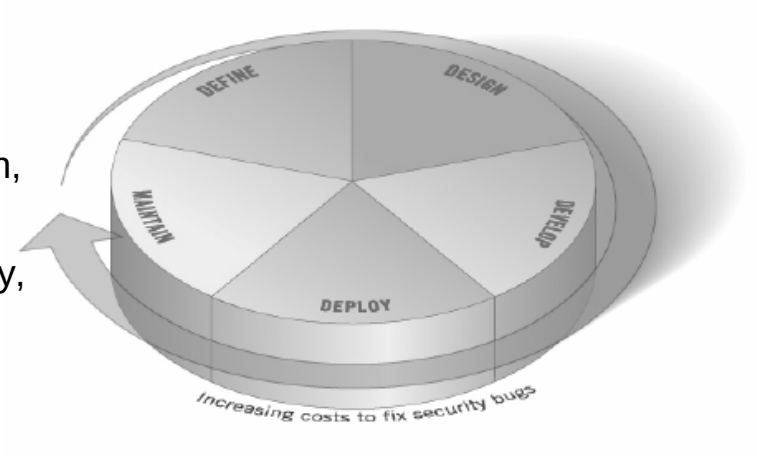
- Test to ensure that there is an adequate SDLC where security is inherent.
- Test to ensure that the appropriate policy and standards are in place for the development team.
- Develop Measurement and Metrics Criteria (Ensure Traceability)

II. The OWASP Testing Framework

Phase 2: During Definition and Design

Before application development has started:

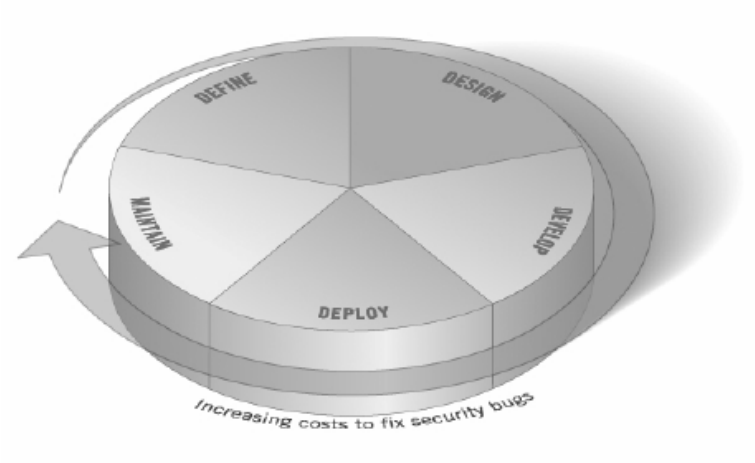
- Security Requirements Review:
  - User Management (password reset etc.), Authentication, Authorization, Data Confidentiality, Integrity, Accountability, Session Management,Transport Security, Privacy
- Design an Architecture Review
- Create and Review UML Models
  - How the application works
- Create and Review Threat Models
  - Develop realistic threat scenarios



II. The OWASP Testing Framework

Phase 3: During Development

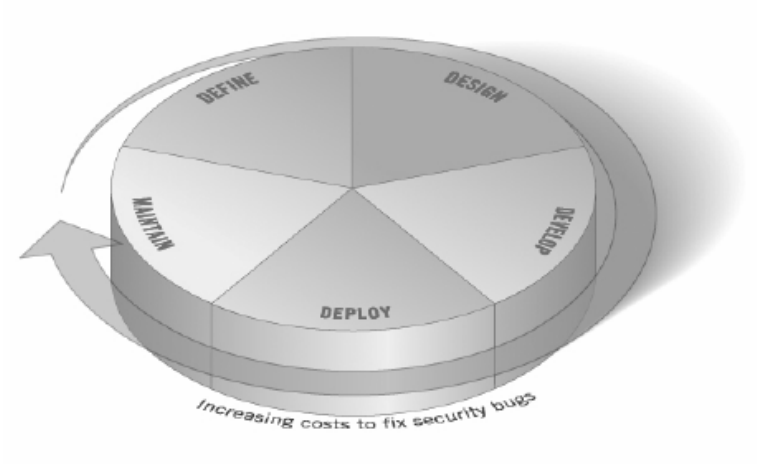
- Code Walkthroughs:
  - high-level walkthrough of the code where the developers can explain the logic and flow.
- Code Reviews:
  - Static code reviews validate the code against a set of checklists:
    - CIA Triad
    - OWASP Top10, OWASP Code Review
    - Sox, ISO 17799, etc…



II. The OWASP Testing Framework

Phase 4: During Deployment

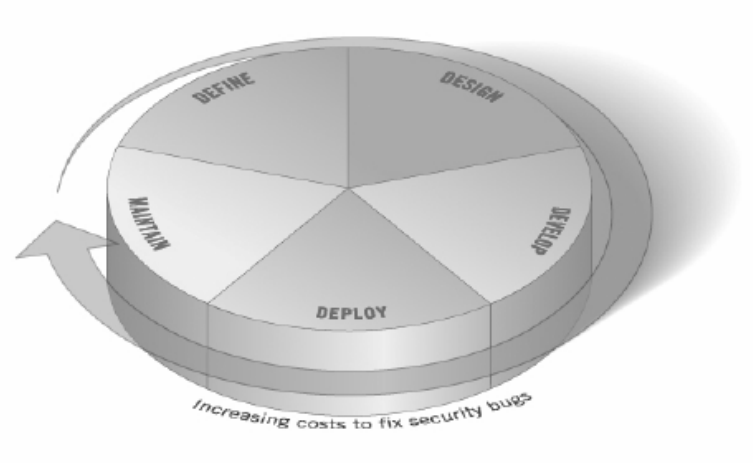
- Application Penetration Testing
  - Focus of this guide
- Configuration Management Testing
  - The application penetration test should include the checking of how the infrastructure was deployed and secured.



II. The OWASP Testing Framework

Phase 5: Maintenance and Operations

- Conduct operational management reviews
  - Process in place which details how the operational side, of the application and infrastructure, is managed.
- Conduct periodic health checks
  - Monthly or quarterly health checks should be performed
- Ensure change verification
  - The change is checked to ensure that the level of security hasn't been affected by the change.



III. Web Application Penetration Testing

- What is a Web Application Penetration Testing?
  - The process involves an active analysis of the application for any weaknesses, technical flaws or vulnerabilities
- What is a vulnerability?
  - A weakness on a asset that makes a threat possible
- Our approach in writing this guide
  - Open
  - Collaborative
- Defined testing methodology
  - Consistent
  - Repeatable
  - Under quality
- OWASP Testing Methodology
  - Penetration testing is only an appropriate technique for testing the security of web applications under certain circumstances.
  - Our goal is to collect all the possible testing techniques, explain them and keep the guide updated.

Testing model

- Black box approach:
  - Tester: Who performs the testing activities
  - Tools and methodology: The core of this Testing Guide project
  - Application: The black box to test
- The test is divided in 2 phases:
  - Passive mode: find all the access points (gates) of the application (e.g. Header HTTP, parameters, cookies).
    - https://www.example.com/login/Autentic_Form.html
    - http://www.example.com/Appx.jsp?a=1&b=1
  - Active mode: test using the methodology described.
- We have split the set of tests in 8 sub-categories (46 controls):
  - Information Gathering
  - Business logic testing
  - Authentication Testing
  - Session Management Testing
  - Data Validation Testing
  - Denial of Service Testing
  - Web Services Testing
  - AJAX Testing

Testing paragraph Template

- **Brief Summary**

Describe in "natural language" what we want to test.

- **Description of the Issue**

Short Description of the Issue: Topic and Explanation

- **Black Box testing and example**
  - **Testing for Topic X vulnerabilities:**

  …
  - **Result Expected:**

  …
- **Gray Box testing and example**
  - **Testing for Topic X vulnerabilities:**

  …
  - **Result Expected:**

  …
- **References**
  - **Whitepapers**
  - **Tools**

Information Gathering

- The first phase in security assessment is focused on collecting all the information about a target application.
- Using public tools (search engines), scanners, sending simple HTTP requests, or specially crafted requests, it is possible to force the application leak information by sending back error messages revealing the versions and technologies used by the application.
- **Application Fingerprint**

  First step: knowing the version and type of a running web server allows testers to determine known vulnerabilities and the appropriate exploits to use during testing.

```
$ nc 216.48.3.18 80
HEAD / HTTP/1.0
HTTP/1.1 200 OK
Date: Mon, 16 Jun 2003 02:53:29 GMT
Server: Apache/1.3.3 (Unix)  (Red Hat/Linux)
Last-Modified: Wed, 07 Oct 1998 11:18:14 GMT
ETag: "1813-49b-361b4df6"
Accept-Ranges: bytes
Content-Length: 1179
Connection: close
Content-Type: text/html
```

Information Gathering

- Application Discovery

  Is the process aimed at identifying web applications on given infrastructure: find out which particular applications are hosted on a web server.

```
nmap –P0 –sT –sV –p1-65535 192.168.1.100
PORT      STATE SERVICE     VERSION
22/tcp    open  ssh         OpenSSH 3.5p1 (protocol 1.99)
80/tcp    open  http        Apache httpd 2.0.40 ((Red Hat Linux))
443/tcp   open  ssl         OpenSSL
901/tcp   open  http        Samba SWAT administration server
1241/tcp  open  ssl         Nessus security scanner
3690/tcp  open  unknown
8000/tcp  open  http-alt?
8080/tcp  open  http        Apache Tomcat/Coyote JSP engine 1.1
```

- Spidering and googling
  - Our goal is to create a map of the application with all the points of access (gates) to the application (wget)
  - Using advanced tips of google, the goal is to find web-site information published on internet

Information Gathering

- Testing for error code

  Error codes generated from applications or web servers reveal a lot of information about databases, bugs, and other technological components directly linked with web application(s).

```
Microsoft OLE DB Provider for ODBC Drivers (0x80004005)
[DBNETLIB][ConnectionOpen(Connect())] - SQL server does not exist or access denied
```

- SSL/TLS Testing

SSL Test Results		
OpenSSL Cipher Name	Cipher Description	Cipher Strength
NULL-MD5	Key Exchange: None; Authentication: None; Encryption: None; MAC: MD5	No Security
NULL-SHA	Key Exchange: None; Authentication: None; Encryption: None); MAC: SHA1	No Security
EXP-DES-CBC-SHA	Key Exchange: RSA(512); Authentication: RSA; Encryption: DES(40); MAC: SHA1	Weak Security
EXP-RC2-CBC-MD5	Key Exchange: RSA(512); Authentication: RSA; Encryption: RC2(40); MAC: MD5	Weak Security
EXP-RC4-MD5	Key Exchange: RSA(512); Authentication: RSA; Encryption: RC4(40); MAC: MD5	Weak Security
EXP1024-DHE-DSS-DES-CBC-SHA	Key Exchange: EDH (EXPORT - 1024); Authentication: DSS; Encryption: DES(56); MAC: SHA1	Weak Security
EXP1024-DHE-DSS-RC4-SHA	Key Exchange: EDH (EXPORT - 1024); Authentication: DSS; Encryption: RC4(56); MAC: SHA1	Weak Security
EXP1024-DES-CBC-SHA	Key Exchange: RSA (EXPORT - 1024); Authentication: RSA; Encryption: DES(56); MAC: SHA1	Weak Security
EXP1024-RC4-SHA	Key Exchange: RSA (EXPORT - 1024); Authentication: RSA; Encryption: RC4(56); MAC: MD5	Weak Security
DES-CBC-SHA	Key Exchange: RSA; Authentication: RSA; Encryption: DES(56); MAC: SHA1	Weak Security
ADH-AES128-SHA	Key Exchange: ADH; Authentication: RSA; Encryption: AES(128); MAC: SHA1	Weak Security
ADH-AES256-SHA	Key Exchange: ADH; Authentication: RSA; Encryption: DES(256); MAC: SHA1	Weak Security
DH-DSS-AES128-SHA	Key Exchange: DH; Authentication: DSS; Encryption: AES(128); MAC: SHA1	Strong Security
DH-RSA-AES128-SHA	Key Exchange: DH; Authentication: RSA; Encryption: AES(128); MAC: SHA1	Strong Security

Information Gathering

- DB Listener Testing

  The DB listener is the entry point for remote connections to an Oracle database: obtain detailed information on the Listener, database, and application configuration.

- File extensions handling

  Determining how web servers handle requests corresponding to files having different extensions may help to understand web server behaviour depending on the kind of files we try to access.

- Old, backup and unreferenced files

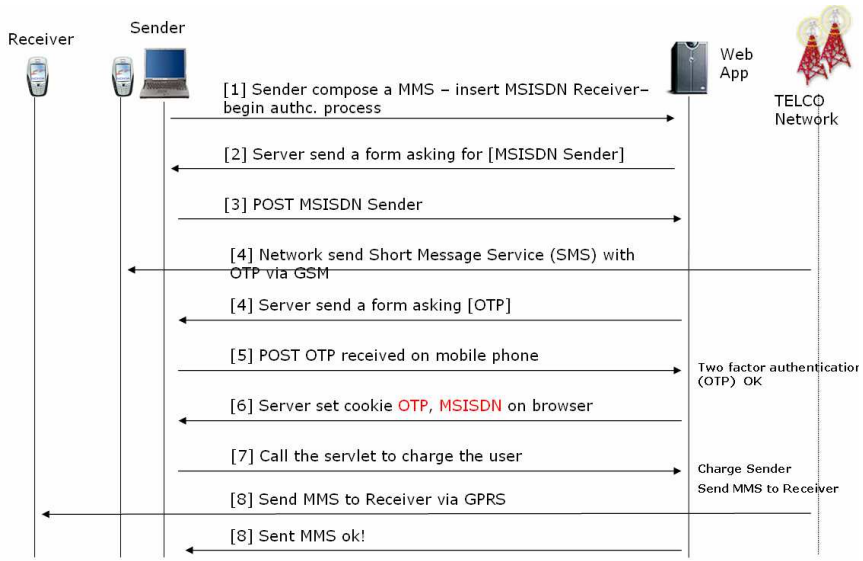
  Leaving in the web tree old files or unreferenced files may reveal sensitive data

Business logic testing

Testing for business logic comprises:

- Business rules that express business policy (such as channels, location, logistics, prices, and products); and
- Workflows that are the ordered tasks of passing documents or data from one participant (a person or a software system) to another.

Test the logic: perhaps you are supposed to do operations in a particular order, but an attacker could invoke them in a different order.



Authentication testing

Testing the authentication schema means understanding how the authentication process works and using that information to circumvent the authentication mechanism.

- **Default or guessable account**

  We test for leave backdoors to easily access and test the application and later forgetting to remove them, non-removable default accounts with a pre-set username and password and blank passwords.

- **Brute Force**

  Systematically enumerating all possible candidates for the solution and checking whether each candidate satisfies the problem's statement.

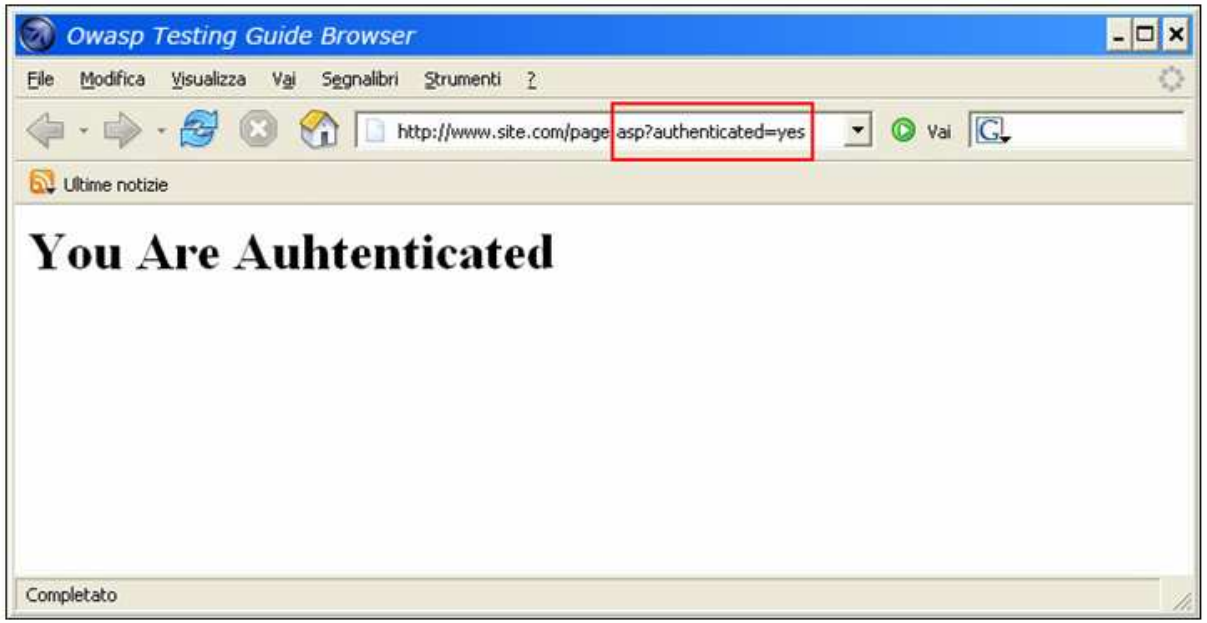
  Brute force on username given a set of password, or bruteforce on password given a set of username.

- **Bypassing authentication schema**

  Test if it’s possible to bypass authentication measures by tampering with requests and tricking the application into thinking that we're already authenticated

Authentication testing (2)

- Bypassing authenticatic
  - Direct page request
  - Parameter Modifica
  - Session ID Predictic
  - Sql Injection
- Directory traversal/file ir
  - Input Vectors Enum vector)
    - http://exam
    - http://example.com/getUserProfile.jsp?item=../../../etc/passwd
    - http://example.com/index.php?file=content
    - http://example.com/main.cgi?home=index.htm



Authentication testing

- Vulnerable remember password and pwd reset

  We test the password reset schema (“security question”) and “cache password" function

- Logout and Browser Cache Management Testing
  - Check that the application provides a logout function
  - Check the session token at logout, and “back button”
  - Re-set the original authc token to test tha application answer
  - Test the time-out logout
  - Cached pages: check for “Pragma: no-cache” directive

Session Management Testing

Session Management Schema

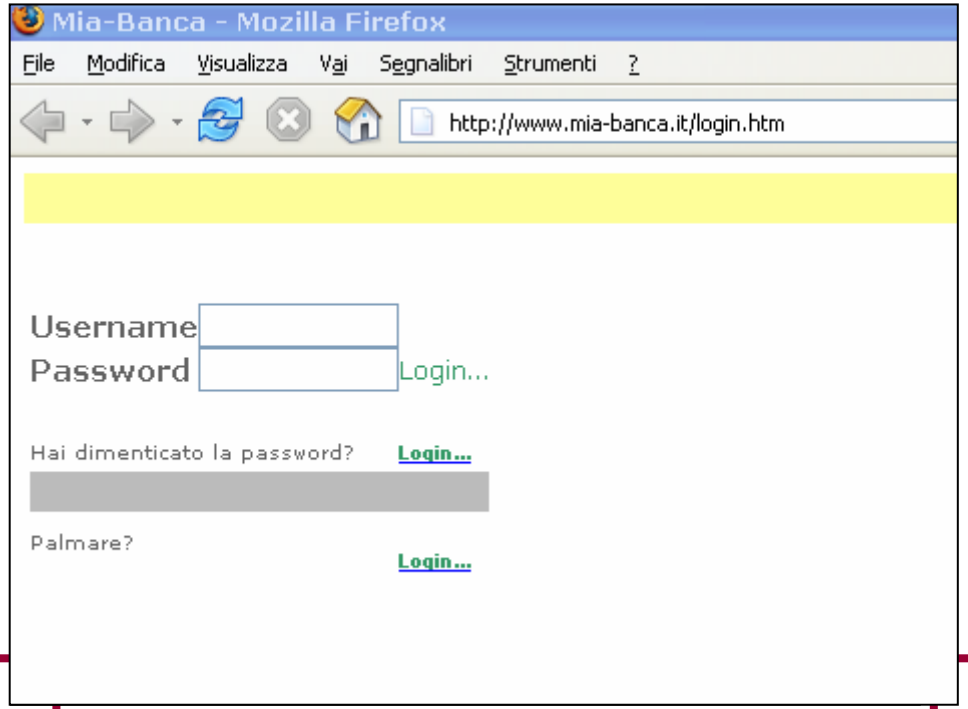
Mario Rossi

-- Authentication process --

Web Application

[1] https://www.mia-banca.it

[2] Sent authentication form over HTTPS



Session Management testing

Mario Rossi

-- Authentication process --

Web Application

[1] https://www.mia-banca.it

[2] Send authentication form over HTTPS

Username/password

[3] Insert username/password via HTTPS

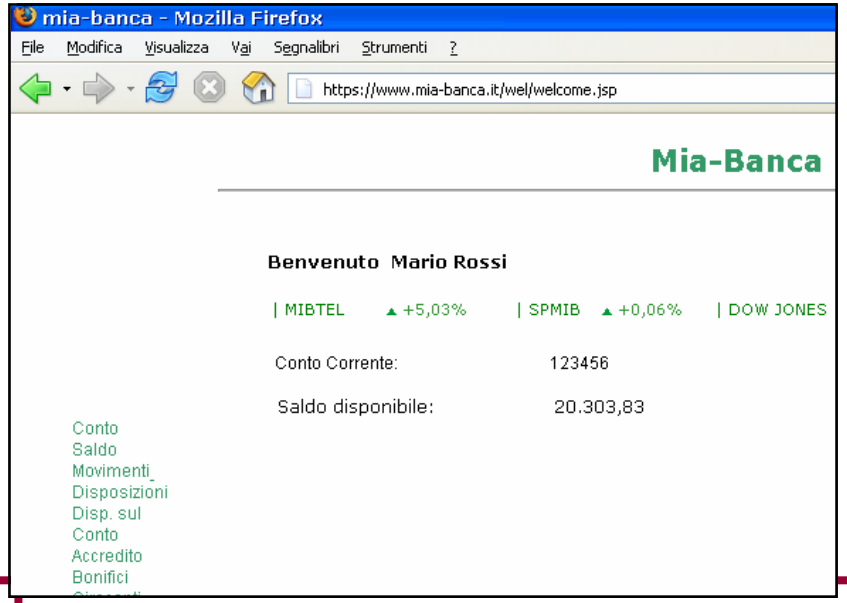
Credential verify: if ok → client authenticated

→ Cookie generation

[4] Personal Welcome page and Set Cookie

Cookie=TWFyaW8123

authentication token



Session management testing

Mario Rossi

--Following request--

Web Application

Cookie=TWFyaW8123

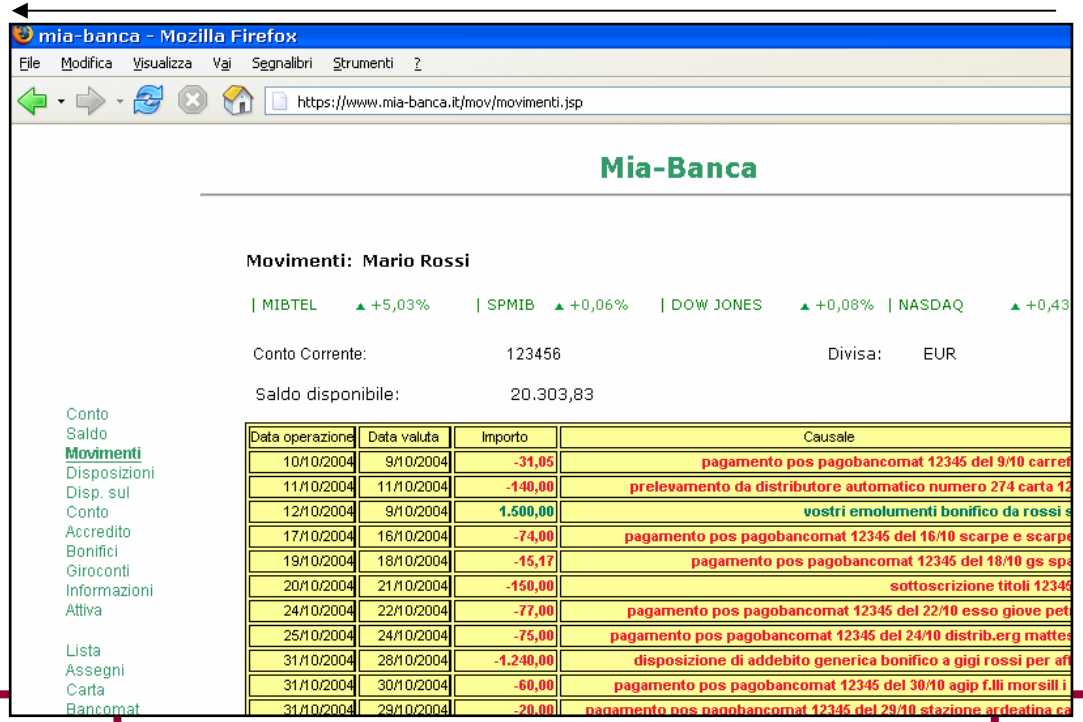
Authentication token

[5] Request "movimenti"

Cookie=TWFyaW8123

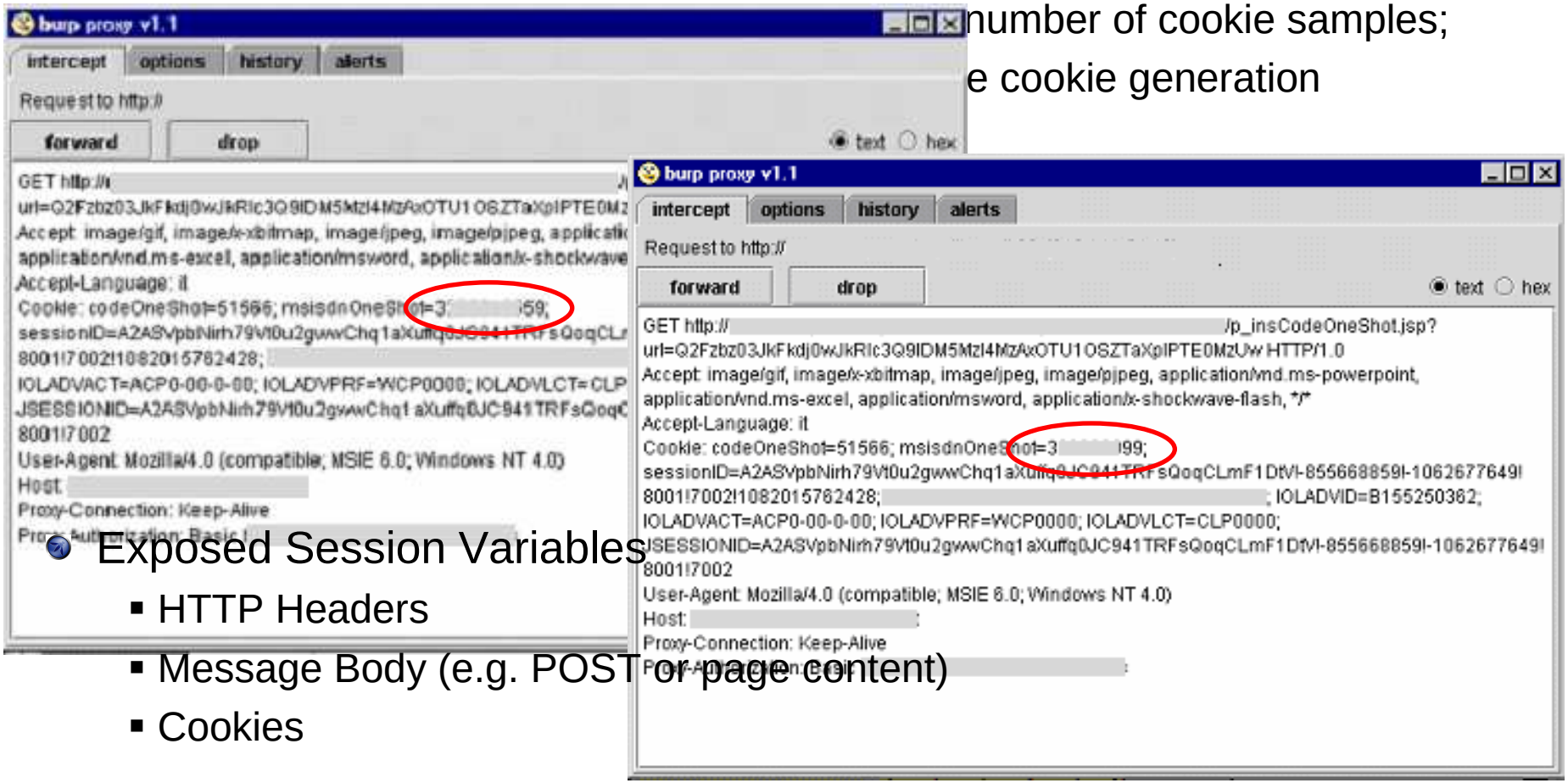
Cookie verifing:
→ Identify user
Send data to user

[6] Response with user data



Session management testing

- Session Token Manipulation



number of cookie samples;

e cookie generation

- Exposed Session Variables
  - HTTP Headers
  - Message Body (e.g. POST or page content)
  - Cookies

Session management testing

- Cross Site Request Forgery

  Test if it is possible to force a user to submit an undesirable command to the logged application:

  <html><body>

  ..

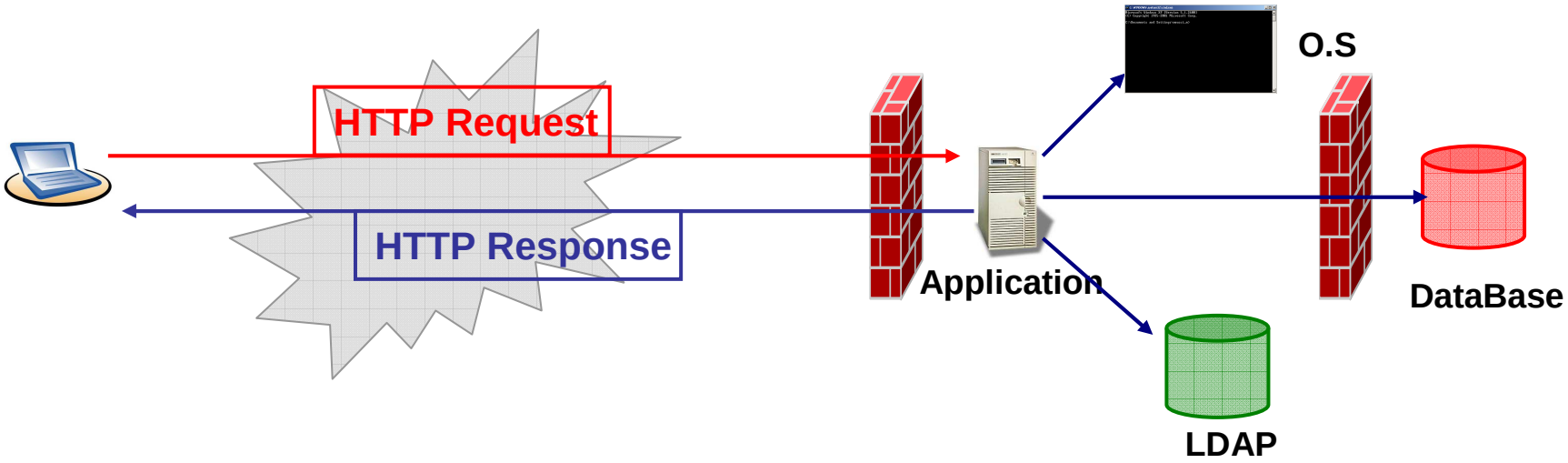
  <img src="https://www.company.example/action" width="0" height="0">

  ...

  </body></html>

- HTTP Exploit
  - HTTP splitting
  - HTTP smuggling

Data validation testing



When an HTTP request arrives from a client:

the application must validate it before interact with all other application's components:

- File System, output, HTTP methods, DB, LDAP, XML doc, IMAP/SMTP command, OS command, code

Data validation testing

- **Cross site scripting**

  Cross Site Scripting (XSS) testing when we try to manipulate the parameters that the application receive in input. A XSS breaks the following pattern:

  Input -> Output == cross-site scripting

- **HTTP Methods and XST**

  Check that the web server is not configured to allow potentially dangerous HTTP methods and that XST is not possible. A XST breaks the following pattern:

  Input -> HTTP Methods == XST

- **SQL Injection**

  The goal is to simulate a manipulation of data in the database that represents the core of every company. An SQL Injection breaks the following pattern:

  Input -> Query SQL == SQL injection

  The Guide analyze Oracle, MySql, Ms SQL Servers testing

Data validation testing

- **LDAP Injection**

  Similar to SQL Injection Testing: the differences are that we use LDAP protocol instead of SQL and the target is an LDAP Server instead of an SQL Server. An LDAP Injection breaks the following pattern:

  Input -> Query LDAP == LDAP injection

- **XML Injection**

  try to inject a particular XML doc to the application: if the XML parser fails to make an appropriate data validation the test will results positive.An XML Injection breaks the following pattern:

  Input -> XML doc == XML injection

- **SSI Injection**

  If the web server's SSI support is enabled, the server will parse the directives received by the HTML. It can enable an attacker to inject code into html pages, or even perform remote code execution.

Data validation testing

- IMAP/SMTP Injection
- Code Injection
- OS Commanding
- Buffer overflow
- Incubated vulnerability

Denial of Service Testing

Usually not performed in "live" environment because you can cause service not available.

DoS are types of vulnerabilities within applications that can allow a malicious user to make certain functionality or sometimes the entire website unavailable. These problems are caused by bugs in the application, often resulting from malicious or unexpected user input.

- Locking Customer Accounts
- User Specified Object Allocation
- User Input as a Loop Counter
- Writing User Provided Data to Disk
- Failure to Release Resources
- Storing too Much Data in Session

Web Services Testing

SOA (Service Oriented Architecture)/Web services applications are up-and-coming systems which are enabling businesses to interoperate and are growing at an unprecedented rate.

The vulnerabilities are similar to other "classical" vulnerabilities such as SQL injection, information disclosure ad leakage etc but web services also have unique XML/parser related vulnerabilities

- XML Structural Testin

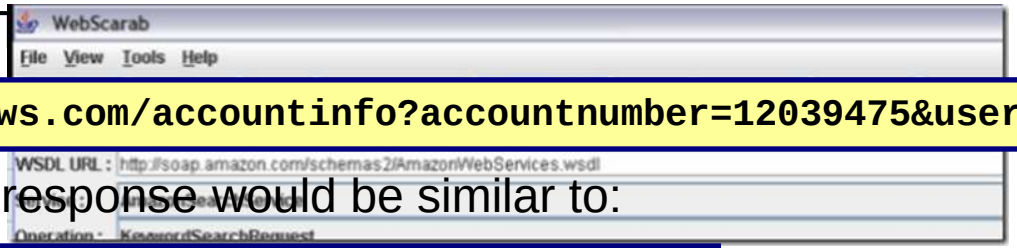
```
<Envelope>
<Header>
   <wsse:Security>
     <Hehehe>I am a Large String (1MB)</Hehehe>
     <Hehehe>I am a Large String (1MB)</Hehehe>
     <Hehehe>I am a Large String (1MB)</Hehehe>…
    <Signature>…</Signature>
   </wsse:Security>
 </Header>
 <Body>
   <BuyCopy><ISBN>0098666891726</ISBN></BuyCopy>
 </Body></Envelope>
```

WS Testing

- XML content-level Testing

  An attacker can craft an XML document (SOAP message) that contains malicious elements in order to compromise the target system. We test for proper content validation.

- HTTP GET




```
https://www.ws.com/accountinfo?accountnumber=12039475&userId=asi9485jfuhe92
```

The resultant response would be similar to:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Account="12039475">
<balance>€100</balance>
<body>Bank of Banana account info</body>
</Account>
```

```
OR 1=1</password>
```

Testing the data validation on this REST web service.Try vectors such as:

```
https://www.ws.com/accountinfo?accountnumber=12039475' exec
master..xp_cmdshell 'net user Vxr pass /Add &userId=asi9485jfuhe92
```

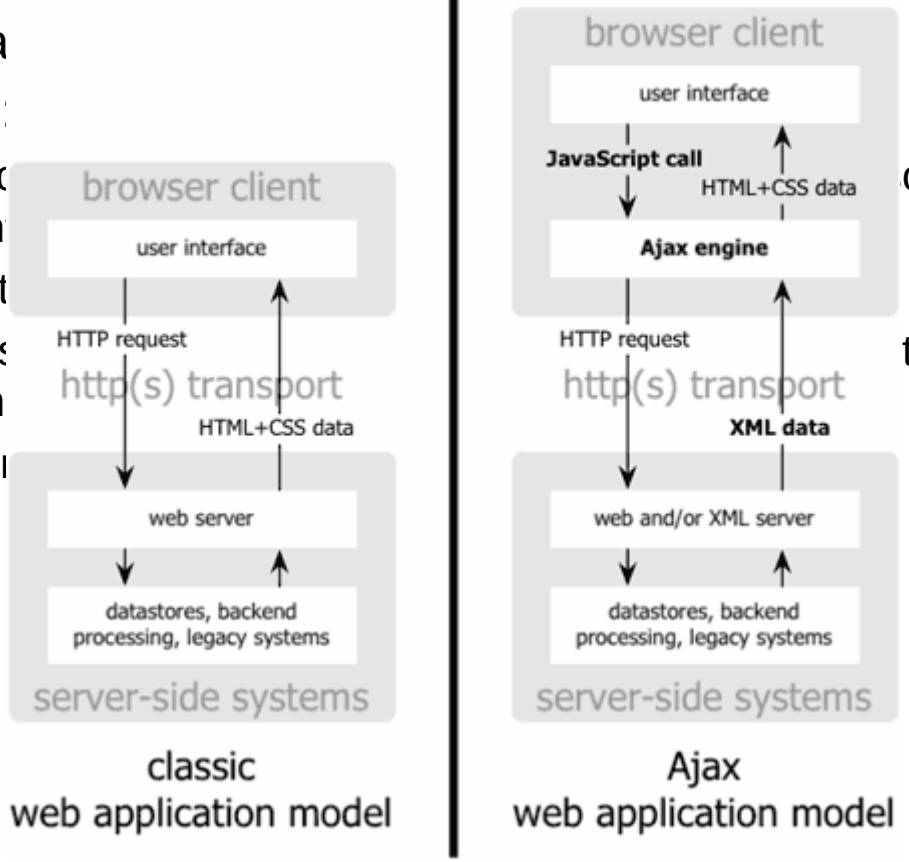
WS Testing

- Naughty SOAP attachments

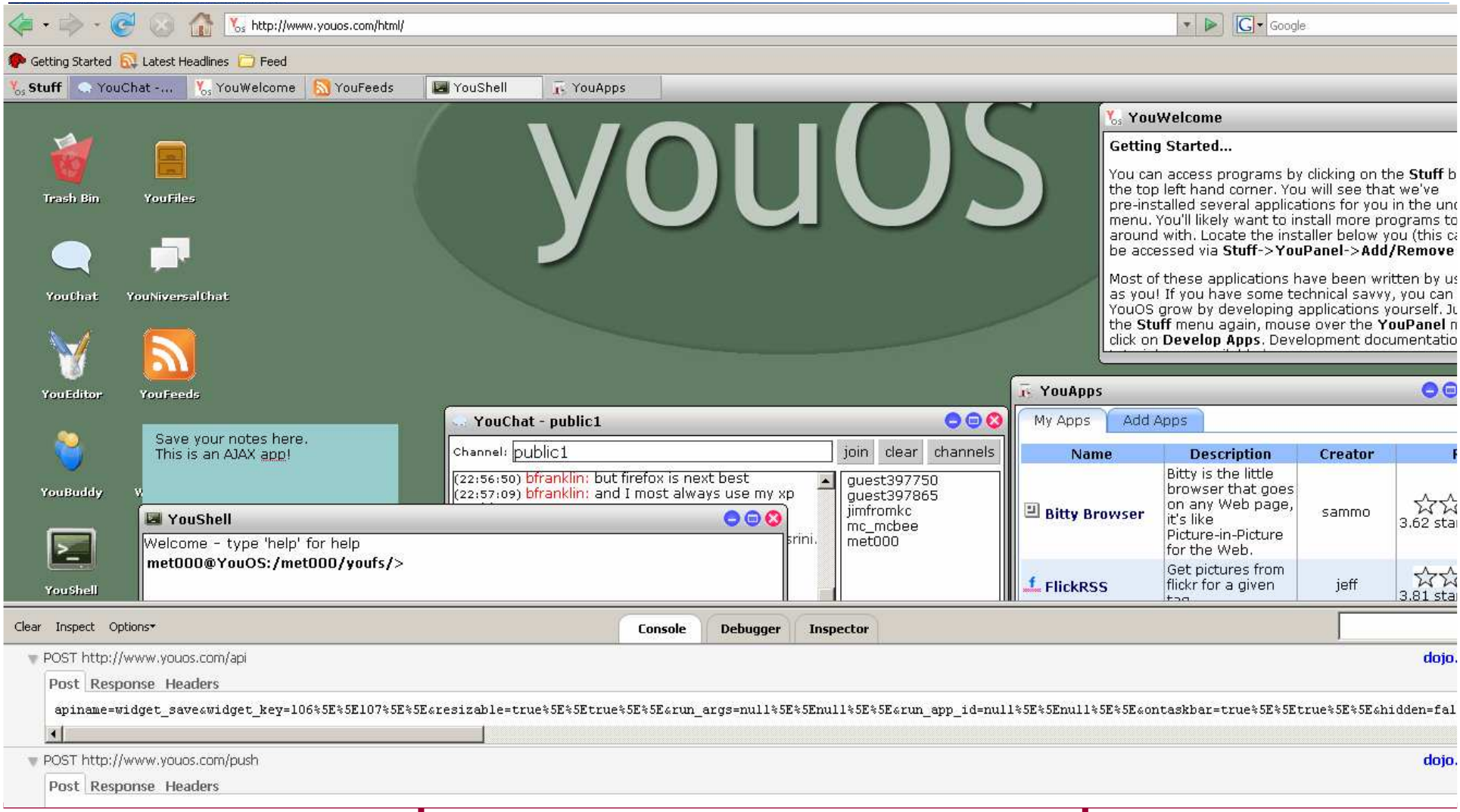
```
POST /Service/Service.asmx HTTP/1.1
Host: somehost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: http://somehost/service/UploadFile
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<UploadFile xmlns="http://somehost/service">
<filename>eicar.pdf</filename>
<type>pdf</type>
<chunk>X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*</chunk>
<first>true</first>
</UploadFile>
</soap:Body>
</soap:Envelope>
```

AJAX Testing

- AJAX (Asynchronous JavaScript and XML) is a web development technique used to create more responsive web applications.
- *XMLHttpRequ* ...onous requests for all communica...
- Security issue...
  - AJAX applic... ...ome logic is moved on the clien...
  - Exposed int...
  - Client acces... ...ty and encoding mechanism...
  - Failure to pr...



AJAX Testing



Testing Report

- The OWASP Risk Rating Methodology
  - Estimate the severity of all of these risks to your business
  - This is not universal risk rating system: vulnerability that is critical to one organization may not be very important to another
- Simple approach to be tailored for every case
  - standard risk model: **Risk = Likelihood * Impact**
- Step 1: identifying a risk

  You'll need to gather information about:
  - the threat agent involved
  - the attack they're using
  - the vulnerability involved
  - the impact of a successful exploit on your business.

Testing Report

- **Step 2: factors for estimating likelihood**

  Generally, identifying whether the likelihood is low, medium, or high is sufficient. Rate 0-9.

  Threat Agent Factors:

  - **Skill level**
  - **Motive**
  - **Opportunity**
  - **Size**

  Vulnerability Factors:

  - **Ease of discovery**
  - **Ease of exploit**
  - **Awareness**
  - **Intrusion detection**

Testing Report

- Step 3: factors for estimating impact

Technical impact:

- **Loss of confidentiality**
- **Loss of integrity**
- **Loss of availability**
- **Loss of accountability**

Business impact:

- **Financial damage**
- **Reputation damage**
- **Non-compliance**
- **Privacy violation**

Testing Report

- Step 4: determining the severity of the risk

Threat agent factors				Vulnerability factors			
Skill level	Motive	Opportunity	Size	Ease of discovery	Ease of exploit	Awareness	Intrusion detection
5	2	7	1	3	6	9	2
Overall likelihood=4.375 (MEDIUM)							

Technical Impact				Business Impact			
Loss of confidentiality	Loss of integrity	Loss of availability	Loss of accountability	Financial damage	Reputation damage	Non-compliance	Privacy violation
9	7	5	8	1	2	1	5
Overall technical impact=7.25 (HIGH)				Overall business impact=2.25 (LOW)			

- In the example above, the likelihood is MEDIUM, and the technical impact is HIGH, so from technical the overall severity is HIGH. **But business impact is actually LOW**, so the overall severity is best described as **LOW** as well.

Testing Report

- Step 5: Deciding What To Fix

  As a general rule, you should fix the most severe risks first.

  Some fix seems to be not justifiable based upon the cost of fixing the issue but may be reputation damage from the fraud that could cost the organization much more than implement a security control
- Step 6: Customizing Your Risk Rating Model
  - Adding factors
  - Customizing options
  - Weighting factors

Writing Report

Category	Ref. Number	Name	Affected Item	Finding	Comment/Solution	Risk
Information Gathering	OWASP-IG-001	Application Fingerprint				
	OWASP-IG-002	Application Discovery				
	OWASP-IG-003	Spidering and googling				
	OWASP-IG-004	Analysis of error code				
	OWASP-IG-005	SSL/TLS Testing				
	OWASP-IG-006	DB Listener Testing				
	OWASP-IG-007	File extensions handling				
	OWASP-IG-008	Old, backup and unreferenced files				
Business logic testing	OWASP-BL-001	Testing for business logic				
	OWASP-AT-001	Default or guessable account				
	OWASP-AT-002	Brute Force				

What's next

- You should adopt this guide in your organization
- Continuously
- What's next:
  - Continuous
  - **OWASP ar**

(P8) Security Analysis Pr

	Suggested Penetration Test and Vulnerability Analysis Procedures	√
Wireless continued	WL110, Enterasys Roamabout Elsa Airlancer MC-11), freeware software, and an antenna and GPS. One technique used for finding a wireless network is War Driving. This is done by detecting the beacon and broadcast. War Driving is used to capture and map wireless band signal.	
	Crack the WEP (Wired Equivalent Privacy) keys by using automated tools such as WEPCrack and AirSnort. The techniques used include IV Collisions and Weak key packet capture.	
	Sniff and analyse the network traffic to ascertain the number of packet passes, SSID, etc. There are a variety of automated tools, such as PrismDump, Iris, AiroPeek and Sniffer Wireless.	
	After the key is known, reassemble the packet to complete the penetration test. Document all issues noted for management review. Before this test, it is best to consult legal representatives practicing within the individual countries and, where necessary, local and state municipalities to provide reasonable assurance that performing this test will not violate any laws or regulations due to picking up information packets from other unintended targets.	
Web Application	Analyse the web application and environment by first crawling through the web pages to gather the information including mapping of all pages and general understanding of all functionality to ascertain risk. Specifically, manually surf the application with a recording proxy (e.g., webproxy, ebsleuth) to find hidden data and locate form weaknesses. In conjunction with this survey, complete the following: ■ Review inventory SSL/TLS ciphers to determine accordance with policies or standard industry practices. ■ Analyse session tracking including mechanism and session ID. ■ Identify authentication methods employed, including client certificates, auditing and revoking certificates, use of encryption or HTTP basic authentication and deployment of SSL. ■ Identify sign-on and sign-off (use of anticaching techniques and session inactivity cause automatic sign-off) mechanisms. ■ Identify all points of user input by recording every form element, specifically: – Test SQL injection – Attempt buffer overflow to gain control – Cross-site scripting (XSS) – Special characters (pipes, returns, etc.) – For numeric input try 0, a negative value, a really large value	

Thank you!

http://www.owasp.org
http://www.owasp.org/OWASP_Testing_Project

matteo.meucci@owasp.org

References :

- OWASP Foundation – “OWASP Building Guide v3” 2006
  http://www.owasp.org/index.php/OWASP_Guide_Project
- OWASP Foundation – “ OWASP Testing Guide v2 RC1” 2007
  http://www.owasp.org/index.php/OWASP_Testing_Project
- OWASP Foundation – “ OWASP Top10”
  http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
- OWASP Foundation Software:

  WebGoat – *http://www.owasp.org/index.php/OWASP_WebGoat_Project*

  WebScarab – *http://www.owasp.org/index.php/OWASP_WebScarab_Project*