OWASP Guadalajara Chapter Meeting

Eduardo Cerna Meza
eduardo.cerna@owasp.org

Manuel López Arredondo
manuel.lopez@owasp.org

AGENDA

- Introduction
- The Open Web Application Security Project - Overview
- OWASP Guadalajara
- Anatomy of the Most Recent Attacks from Anonymous and Countermeasures
- OWASP Guadalajara – Business Time

AGENDA

- Introduction
- The Open Web Application Security Project - Overview
- OWASP Guadalajara
- Anatomy of the Most Recent Attacks from Anonymous and Countermeasures
- OWASP Guadalajara – Business Time

Mission

Make application security visible so that people and organizations can make informed decisions about true application security risk

- What causes?
- Immediate causes – vulnerabilities themselves
- Developers and operators
- Organizational structure, development process, supporting technology
- Increasing connectivity and complexity
- Legal and regulatory environment
- Asymmetric information in the software market

CORE

The Open Web Application Security Project (OWASP) is a 501c3 not-for-profit worldwide charitable organization focused on improving the security of application software. Our mission is to make application security visible, so that people and organizations can make informed decisions about true application security risks.

Everyone is free to participate in OWASP and all of our materials are available under a free and open software license.

Approach == "Open"

- Open means rough consensus and running code
- Open means free to use and modify
- Open means independent
- Open means open information sharing
- Open means wider audience and participation

Our Successes

- OWASP Tools and Documentation
  - ~15,000 downloads (per month)
  - ~30,000 unique visitors (per month)
  - ~2 million website hits (per month)
- OWASP Chapters are blossoming worldwide
  - 1500+ OWASP Members in active chapters worldwide
  - 20,000+ participants
- OWASP AppSec Conferences
  - Chicago, New York, London, Washington D.C, Brazil, China, Germany, more…
- Distributed content portal
  - 100+ authors for tools, projects, and chapters
- Global Citations
- Professional Association of Information Security Peers

The OWASP Foundation

http://www.owasp.org

Organization and Barter In Trade Supporters

acunetix
ACCUVANT LABS
Adobe
ADP
CERT ae
Akamai
amazon
AppliedTrust
ascure
ASPECT SECURITY
Application Security Experts
ASTECH security
BEST BUY
Black Hat
Cargill
cigital
CORE SECURITY TECHNOLOGIES
DENIM GROUP
DIGITAL DEFENSE INCORPORATED
DREAMLAB TECHNOLOGIES
ENGINEERING
f5
fishnet SECURITY
FUJITSU
gemalto security to be free
GOTHAM DIGITAL SCIENCE
HARRIS CONNECT
HIGH-TECH BRIDGE
protiviti Independent Risk Consulting
QUALYS ON DEMAND SECURITY
Quotium
rackspace
Redspin
salesforce.com Success On Demand
SD ELEMENTS
Security Innovation
SECURITY PS
Seeker
Symantec
TENABLE Network Security
Trustwave
ups
WhiteHat SECURITY

The OWASP Foundation

http://www.owasp.org

Academic Supporters

Chapters

- The more chapters the better!

2011 - OWASP AppSec Conferences

~140 Projects

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

A5

(CSRF)

Threat Agents

Attack Vectors

Exploitability AVERAGE

Security Weakness

Prevalence WIDESPREAD

Detectability EASY

of web applications

A1 – Injection

A2 – Cross-Site Scripting (XSS)

A3 – Broken Authentication and Session Management

A4 – Insecure Direct Object References

A5 – Cross-Site Request Forgery (CSRF)

A6 – Security Misconfiguration (NEW)

A7 – Insecure Cryptographic Storage

A8 – Failure to Restrict URL Access

A9 – Insufficient Transport Layer Protection

Unvalidated Redirects and Forwards (NEW)

OWASP

The Open Web Application Security Project

OWASP Top 10 - 2010

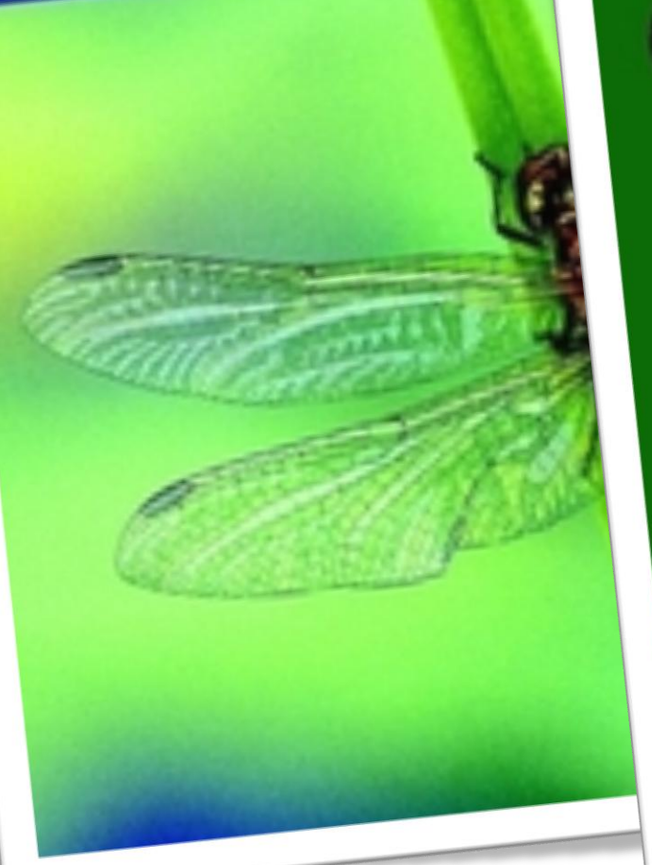
The Ten Most Critical Web Application Security Risks

release

OWASP
The Open Web Application Security Project

OWASP Developers Guide v2.0

release



OWASP
The Open Web Application Security Project

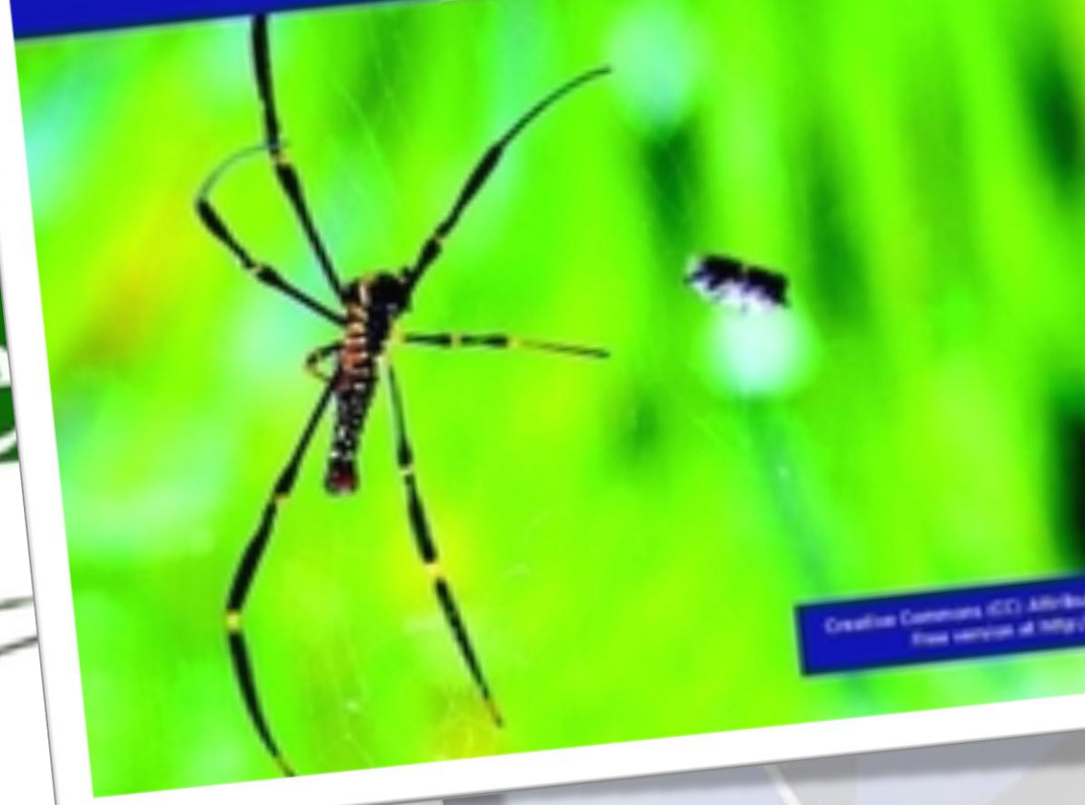
Code Review Guide

rele

OWASP
The Open Web Application Security Project

OWASP Testing Guide v3.0

releas



Bypass a Path Based Access Control Scheme - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://localhost/WebGoat/attack?Screen=5&menu=210 Go

Logout

Bypass a Path Based Access Control Scheme

OWASP WebGoat V5.1

Hints Show Params Show Cookies Show Java Show Solution Lesson Plans

Restart this Lesson

- Admin Functions
- General
- Code Quality
- Concurrency
- Unvalidated Parameters
- Access Control Flaws
  - Using an Access Control Matrix
  - Bypass a Path Based Access Control Scheme
  - LAB: Role Based Access Control
    - Stage 1: Bypass Business Layer Access Control
    - Stage 2: Add Business Layer Access Control
    - Stage 3: Bypass Data Layer Access Control
    - Stage 4: Add Data Layer Access Control
  - Remote Admin Access
- Authentication Flaws
- Session Management Flaws
- Cross-Site Scripting (XSS)
- Buffer Overflows
- Injection Flaws
- Improper Error Handling
- Insecure Storage
- Denial of Service
- Insecure Configuration
- Web Services

The 'guest' user has access to all the files in the lesson_plans directory. Try to break the access control mechanism and access a resource that is not in the listed directory. After selecting a file to view, WebGoat will report if access to the file was granted. An interesting file to try and obtain might be a file like tomcat/conf/tomcat-users.xml

Current Directory is: C:\WebGoat-5.1\tomcat\webapps\WebGoat\lesson_plans

Choose the file to view:

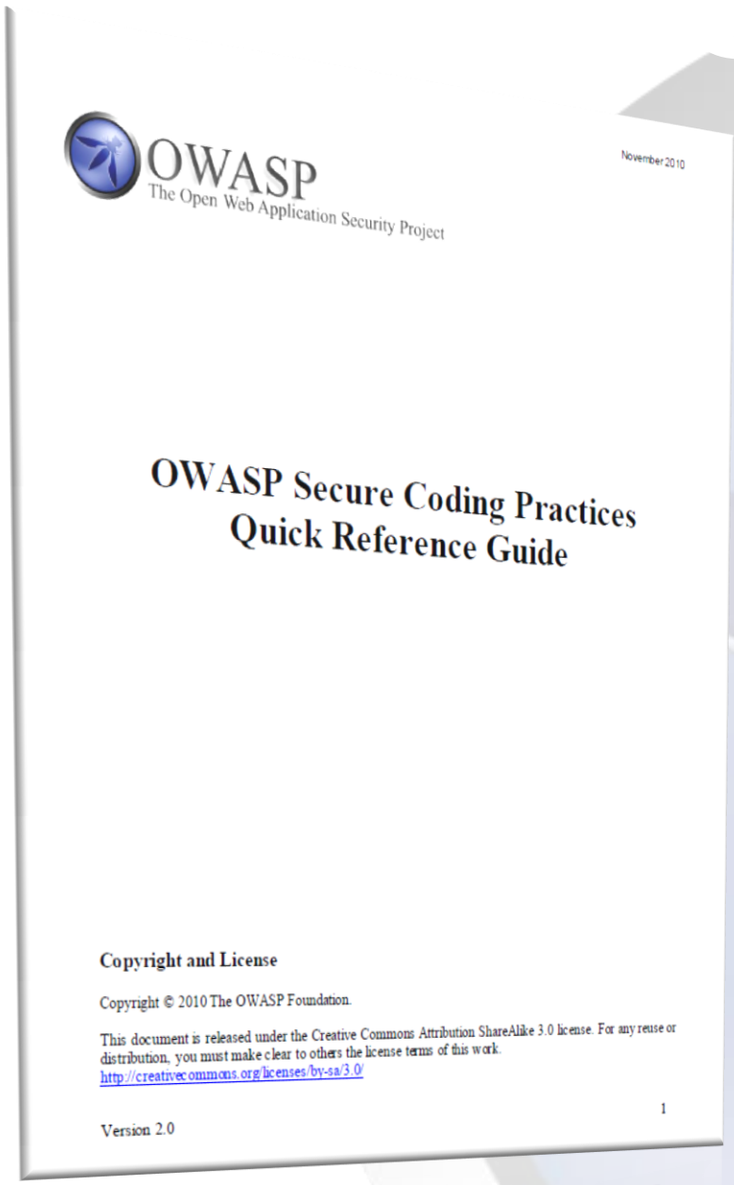
AccessControlMatrix.html
BackDoors.html
BasicAuthentication.html
BlindSqlInjection.html
BufferOverflow.html
ChallengeScreen.html
ClientSideFiltering.html
ClientSideValidation.html
CommandInjection.html
ConcurrencyCart.html
CrossSiteScripting.html
CSRF.html
DangerousEval.html
DBCrossSiteScripting.html
DBSQLInjection.html

View File

Viewing file: C:\WebGoat-5.1\tomcat\webapps\WebGoat\lesson_plans

Local intranet

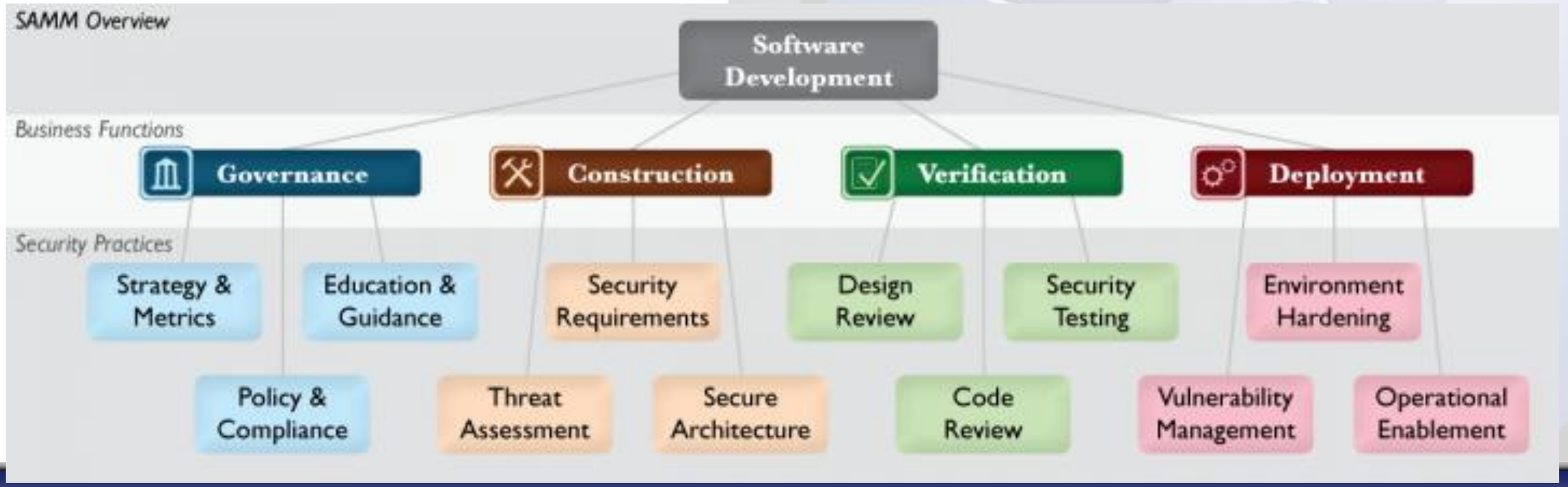

Home
System
OWASP
Live CD
The Open Web Application Security Project
OWASP
All Applications
OWASP Live CD
Games
Graphics
Internet
Multimedia
Office
System
Utilities
Control Center
Find Files/Folders
Personal Files (Home)
Actions
Run Command...
Switch User
Lock Session
Log Out...
KDE 3.5
Burp Suite (Local Proxy)
CAL9000 (Web App Testing Tools)
DirBuster (Directory Brute Forcer)
Documentation (OWASP Live CD Docs)
Firefox 3 (Web Browser)
Grendel Scan (Web App Sec Scanner)
JBroFuzz (Network Protocol Fuzzer)
Paros (Local Proxy)
SQLiX (SQL Injection Scanner)
Start msfweb (Metasploit Web Interface)
Stop msfweb (Metasploit Web Interface)
Subversion client (Manage your source code)
WSFuzzer (Web Services Testing Tool)
WebGoat (Start WebGoat on port 80)
WebGoat (Start WebGoat on port 8080)
WebGoat (Stop WebGoat)
WebScarab (Local Proxy)
Wireshark (Packet Sniffer)
msfcli (Metasploit Command Line Interface)
msfconsole (Metasploit Interactive
nmap (Network



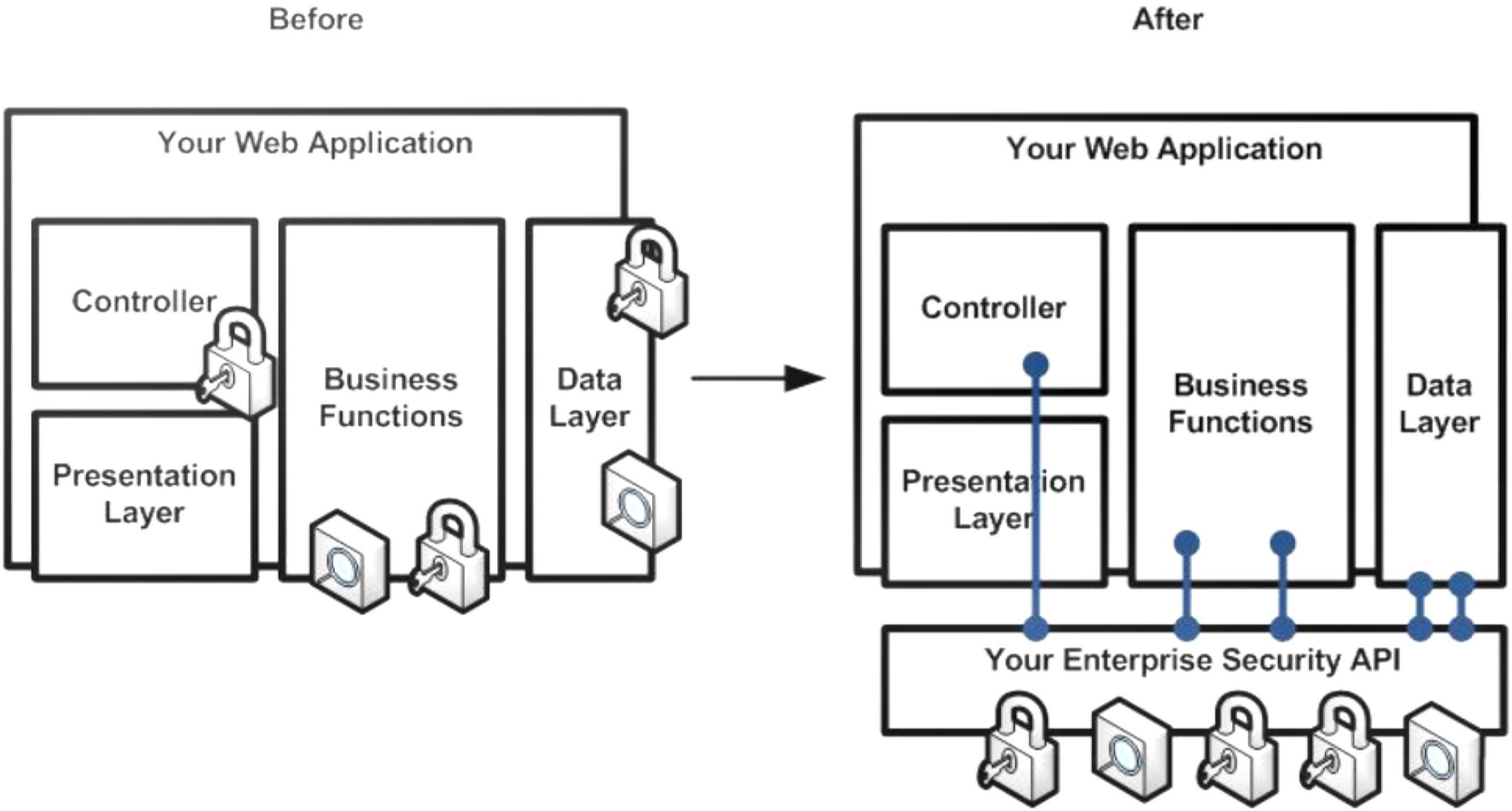
OWASP Cheat Sheets

- Authentication
- Cross-Site Request Forgery (CSRF) Prevention
- Cryptographic Storage
- SQL Injection Prevention
- Transport Layer Protection
- XSS (Cross Site Scripting) Prevention

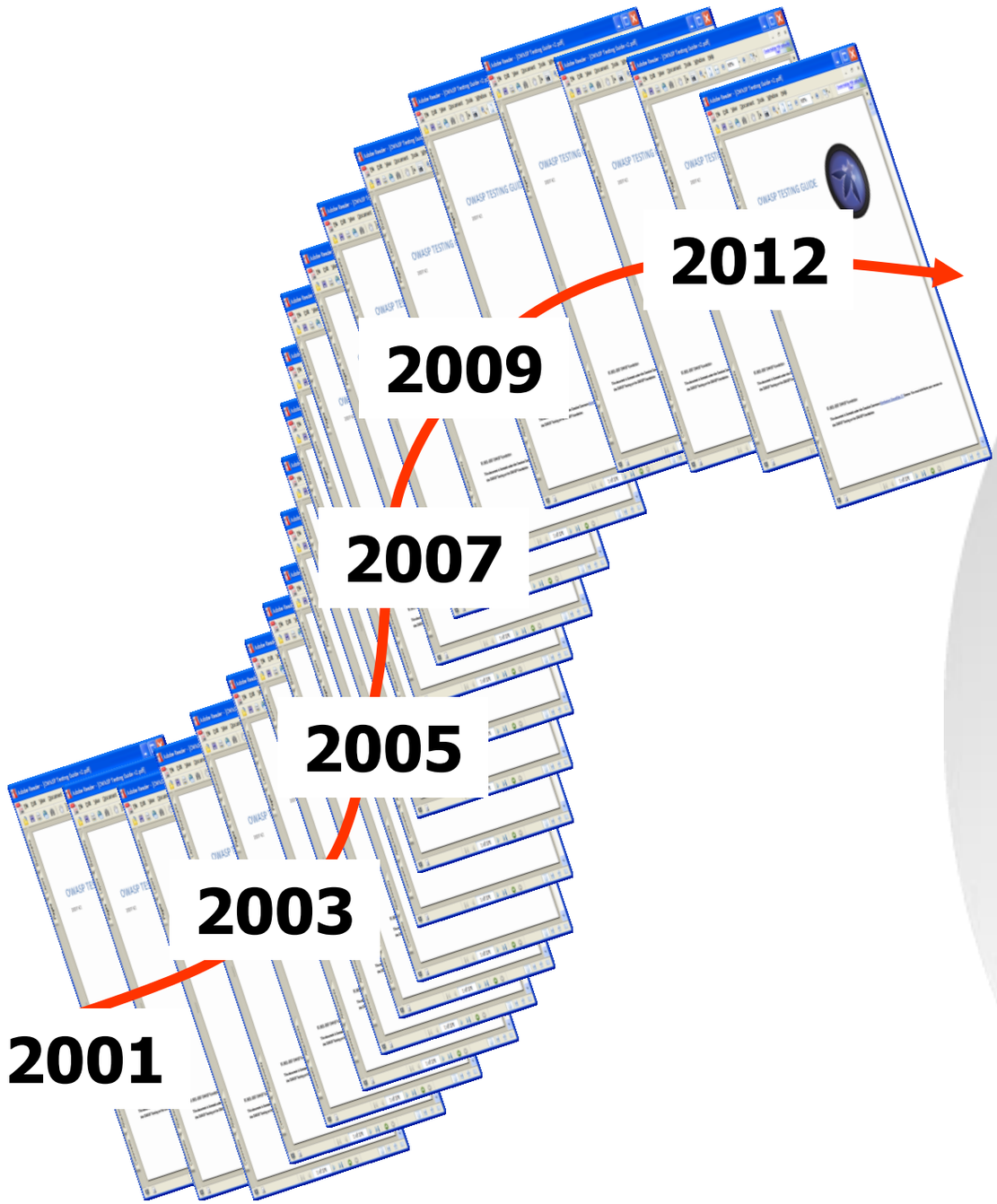
OWASP Software Assurance Maturity Model



OWASP Enterprise Security API



OWASP Projects Are Alive!



...

OWASP Meritocracy

OWASP Users and Participants

OWASP Members

OWASP Leaders
(Chapters and Project)

Projects | Membership | Education | Conferences | Industry | Chapters | Connections

OWASP Foundation
(OWASP Board)

www.owasp.org

22

AGENDA

- Introduction
- The Open Web Application Security Project - Overview
- OWASP Guadalajara
- Anatomy of the Most Recent Attacks from Anonymous and Countermeasures
- OWASP Guadalajara – Business Time

Guadalajara Chapter – House Rules

- Quarterly Meetings
- Local Mailing List
- Presentations & Groups
- Open forum for discussion
- Meet fellow InfoSec professionals
- Create (Web)AppSec awareness in Guadalajara
- Local projects?

Guadalajara Chapter – House Rules

- Free & open to everyone
- No vendor pitches or $ales presentations
- Respect for different opinions
- No flaming (including M$ bashing)
- 1 CISSP CPE for each hour of OWASP chapter meeting
- Sign Sheet & I'll e-mail scan: you claim CPE credits

Subscribe mailing list

https://lists.owasp.org/mailman/listinfo/owasp-guadalajara

Keep up to date!

Want to support OWASP?

Become member, annual donation of:

- $50 Individual
- $5000 Corporate

enables the support of OWASP projects, mailing lists, conferences, podcasts, grants and global steering activities...

AGENDA

- Introduction
- The Open Web Application Security Project - Overview
- OWASP Guadalajara
- Anatomy of the Most Recent Attacks from Anonymous and Countermeasures
- OWASP Guadalajara – Business Time

The Need For Secure Code

Part I

8 February, 2012 | Carlos Fernández de Lara

Hackers filtran datos de 3,000 empleados de Telcel

"Iluminati de México, somos Anonymous S
conocemos sus empresas y sabemos para
las palabras iniciales que preceden a la inf

El mensaje continua con una amenaza: "P
aquellos que atentan contra las garantías

SECTOR 404 HACKED BY:
@SestorLeaks_404
My name is PHANTOM

#OpIlimunati Sabemos quienes mueven este Pais, iremos poco a poco por cada uno de ustedes y sacaremos mucho mas datos ¿que es esto? no es nada, la SEP es NUESTRA! salu2 Mortales.

Host IP: 168.255.254.29 http://promep.sep.gob.mx/ca1/firmadoiesMEJORA.php?

What is Application Security?

The threats that are going to be discussed on this course are beyond our control and will always exist. The overall RISK of compromise is the product of these threats multiplied the likelihood of occurrence or existence of vulnerabilities.

OVERALL RISK = THREATS x VULNERABILITIES

The goal of Application Security is to identify and eliminate (to the maximum extent possible) the vulnerabilities that exist within our Applications, Systems and Source Code. Risk can never be eliminated entirely.

Insecure Software Costs

76%

of Software and Applications Tested Have Serious Design and Implementation Flaws

- Foundstone Survey*

$60 B

in Cost to USA Economy from Poor Software Quality

–US Dept of Commerce

$3B

Cost of Insecure Software to the Financial Services Industry

– NIST Survey 2002

100x

100 Times More Expensive to Fix Security Bug at Production Than Design

– IBM Systems Sciences Institute

¿Why Security?

Protect Assets (Proteger los Activos)

Tangibles Assets (Proteger los Activos Materiales)

Intangibles Assets (brand, the name , reputation) (Proteger marca, nombre, reputación)

Company Assets (Proteger Activos de la Compañía)

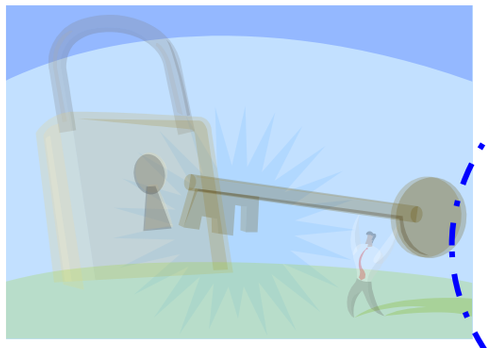
Clients Assets (Proteger Activos de los Clientes)

Security is the strategy, not the goal

¿Why Attackers Target Software?

Objective	Property Compromised	Life Cycle Phase	Type of Software Susceptible
Modify/subvert the functionality of the software (often to more easily accomplish another compromise)	integrity	development or deployment	all systems
Prevent authorized users from accessing the software, or preventing the software from operating according to its specified performance requirements (known as "denial of service")	availability	deployment	all systems
Read data controlled, stored, or protected by the software that he/she is not authorized to read	confidentiality	deployment	information systems
Access to data, functions, or resources he/she is not authorized to access, or perform functions he/she is not authorized to perform	access control	deployment	all systems
Obtain privileges above those authorized to him/her (to more easily accomplish another compromise) (known as "escalation of privilege")	authorization	deployment	all systems

Security Process



Aplicación

Validación de INput
Autenticación
Autorización
Gestión de Configuración
Datos Sensibles

Gestión de Sesiones
Criptografía
Manipulación de Parámetros
Gestión de Excepciones
Auditoría y Logging

Red

Router
Firewall
Switch

Firewall

Servidor Web

Firewall

Servidor de Aplicaciones

Servidor de BD

Servidor

Parches
Servicios
Protocolos

Cuentas
Ficheros y Directorios
Shares

Puertos
Registro
Auditoría y Logging

Security vs Vulnerability

Security- mitigating risk at a cost. (Seguridad – Mitigar riesgo a un costo)

Vulnerability – exploitable fault (Vulnerabilidad – Avería Explotable)

This is for real! Who can Help me out?

- **OWASP**
- **WASC**
- **Security**
- **Other Security Consortiums**

OWASP Top 10 Vulnerabilities

1.-Cross Site Scripting (XSS)

2.-Injection Flaws

3.-Malicious File Execution

4.-Insecure Direct Object Reference

5.-Cross Site Request Forgery (CSRF)

6.-Information Leakage and Improper Error Handling

7.-Broken Authentication and Session Management

8.-Insecure Cryptographic Storage

9.-Insecure Communications

10.-Failure to Restrict URL Access

1. Cross-Site Scripting (XSS)

- XSS is client-side code such as JavaScript injected by malicious users into web pages.

- XSS are commonly used to steal client information, bypass access controls and to craft phishing attacks.

- Attackers take advantage of the not validated user inputs on web applications in order to embed malicious scripts on web pages, the attack is executed once the inserted script is returned to the client.

1.Types of (XSS)

- DOM Based XSS. This vulnerability is being executed when a piece of javascript code accesses a URL request parameter and uses this information to write HTML code to its own page, the XSS will be executed due to the inserted code will be reinterpreted as HTML which could include additional client-side scripts. (Un script local accede a parámetros request de la URL y los utiliza para construir código script).

- Reflected XSS. non-persistent or reflected XSS means that the malicious payload is reflected by the server in an immediate response from an HTTP request made by the victim. (Se utilizan l os datos de entrada de formularios para construir scripts, permitiendo la inyección de código).

- Persistent XSS. Persistent or stored XSS are those where the payload is stored by the system and latter embedded in the vulnerable pages of the systems. This attack is executed every time the affected pages by the payload are being called. (Es la más poderosa. Los datos se almacenan en una base de datos y posteriormente se muestran a los usuarios, inyectando el código continuamente).

1.Vulnerability (XSS)

Cross-site scripting (XSS) attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user

Examples:

';alert(String.fromCharCode(88,83,83))//\';alert(String.fromCharCode(88,83,83))//"
;alert(String.fromCharCode(88,83,83))//\";alert(String.fromCharCode(88,83,83))//-
-></SCRIPT>">'><SCRIPT>alert(String.fromCharCode(88,83,83))</SCRIPT>

<IMG SRC=javascript:alert('XSS')>

Embedded encoded:

**<IMG SRC="jav	ascript:alert('XSS');">**

Hex Encoding

<A HREF="http://0x42.0x0000066.0x7.0x93/">XSS</A>

1.Protection (XSS)

Input validation.

- Positive Validation=“accept known good”
- Please make sure to validate the following:
- -Allowed special character set.(A-Z, a-z, 0-9, @)
- -code filters for ‘ / < >
- -Whether Null or empty value is allowed
- -whether the parameter is required or not.
- -type and format of the field
- -Minimum and Maximum field length
- -Use of patterns (regular expressions)
- if numeric field, we recommend to please work with ranges.
- Reject invalid input rather than attempting to sanitize potentially hostile data.
- Do not forget that error messages might also include invalid data

Output Encoding.

- **Ensure that all user-supplied data is appropriately entity encoded:**
- **Specify the output encoding (such as ISO 8859-1 or UTF 8). Do not allow the attacker to choose this for your users**

.NET Anti-XSS Library

http://msdn.microsoft.com/en-us/security/aa973814

OWASP AntiSamy Project

https://www.owasp.org/index.php/Category:OWASP_AntiSamy_Project#Stage_1_-_Downloading_AntiSamy

https://www.owasp.org/index.php/Top_10_2010-A2-Cross-Site_Scripting_(XSS)

1. XSS - Practice

2.Injection Flaws (SQL Injection)

- SQL is a powerful language which allows us to work with Relational Database Management Systems (RDBMS), offering a great reliability, robustness, and flexibility that in turn could become a big security hole.

- What Is SQL Injection?

The ability to inject SQL commands into the database engine through an existing application.

2.Injection Flaws (SQL Injection)

What is?

```
SqlConnection conn= new SqlConnection(
  “server=localhost;Database=Northwind” +
    “user id=sa;password=pass*word;");

string sqlString="SELECT * FROM Orders WHERE “
        + “CustomerID='" + idCliente + "'";

SqlCommand cmd = new SqlCommand(sqlString, conn);
                conn.Open();
  SqlDataReader reader = cmd.ExecuteReader();

                // ...
```

2.Injection Flaws (SQL Injection)

How works?

Datos validos: PRUEBA

```
SELECT * FROM
Orders WHERE
CustomerID='PRUEBA'
```

Los atacantes nivel superior: PRUEBA';exec xp_cmdshell 'format C:

```
SELECT * FROM
SELECT * FROM
Orders WHERE CustomerID=
'PRUEBA';exec xp_cmdshell 'format C:'
```

2.Injection Flaws (SQL Injection)

There are some characters, which have special meaning for RDBMS since they're part of the RDBMS SQL language itself. The following lists some of those special characters :

- ' or " character String Indicators
- -- or # single-line comment
- /*...*/ multiple-line comment
- + addition, concatenate
- || (double pipe) concatenate
- % wildcard attribute indicator

2.Injection Flaws (SQL Injection)

Let’s think of a SQL command which expects a string variable to be include in the WHERE clause and such command is hard coded in some 4G language.

“SELECT First_Name FROM workers
WHERE employee_id = ‘” + variable +”””;

What if our variable’s value is: ‘ or ‘’=‘ ?

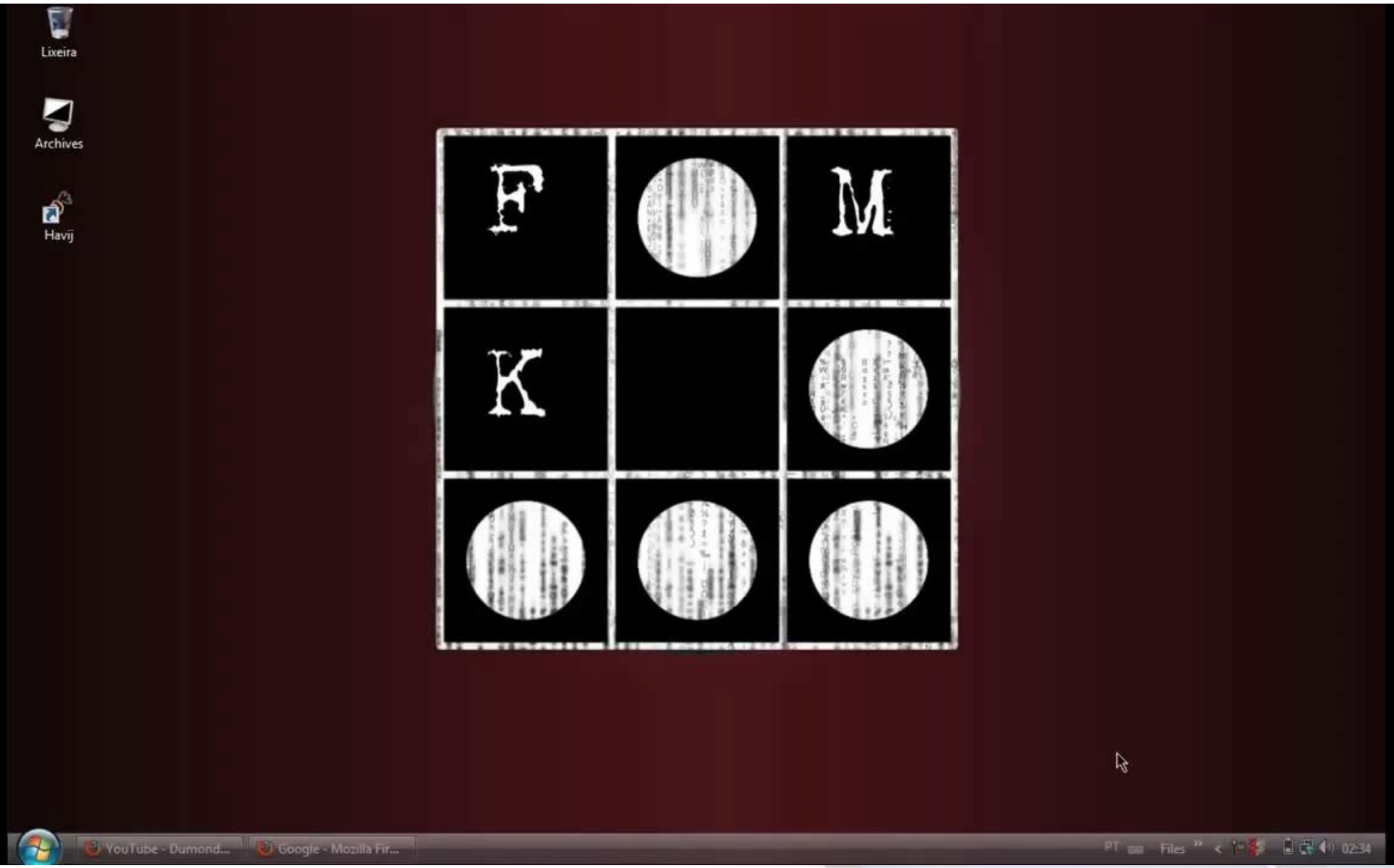
Once it gets parsed it ends as follows:

SELECT First_Name FROM workers
WHERE employee_id = “ or “ =”;

Now our condition is true for all cases. Therefore the actual SQL sentence is:

SELECT First_Name FROM workers;

2. SQL Injection - Example


Lixeira
Archives
Havij
F
M
K
YouTube - Dumond...
Google - Mozilla Fir...
PT
Files
02:34

2. SQL Injection - Practice

2.Protection (SQL Injection)

Input validation.
Positive Validation="accept known good"
Please make sure to validate the following:
-Allowed special character set.(A-Z, a-z, 0-9, @)
-code filters for ` / < >
-Whether Null or empty value is allowed
-whether the parameter is required or not.
-type and format of the field
-Minimum and Maximum field length
-Use of patterns (regular expressions)
if numeric field, we recommend to please work with ranges.
Reject invalid input rather than attempting to sanitize potentially hostile data.
Do not forget that error messages might also include invalid data

- **Use strongly typed parameterized query APIs with placeholder substitution markers, even when calling stored procedures**
- **Enforce least privilege when connecting to databases and other backend systems**
- **Avoid detailed error messages that are useful to an attacker**
- **Do not use dynamic query interfaces (such as mysql_query() or similar)**
- **Watch out for canonicalization errors**

- **https://www.owasp.org/index.php/Top_10_2010-A1**

2.Injection Flaws (SQL Injection)

Let’s think of a SQL command which expects a string variable to be include in the WHERE clause and such command is hard coded in some 4G language.

“SELECT First_Name FROM workers
WHERE employee_id = ‘” + variable +”””;

What if our variable’s value is: ‘ or ‘’=‘ ?

Once it gets parsed it ends as follows:

SELECT First_Name FROM workers
WHERE employee_id = “ or “ =”;

Now our condition is true for all cases. Therefore the actual SQL sentence is:

SELECT First_Name FROM workers;

Other References

CERT

- www.cert.org

Security focus

- www.securityfocus.org

NIST

- www.nist.gov

NSA

- www.nsa.gov

OWASP

- www.owasp.org

MS Security

- microsoft.com/security

Java

- java.sun.com/security

IT Security.com

- www.itsecurity.com

Questions?

Eduardo Cerna Meza

eduardo.cerna@owasp.org

Manuel López Arredondo

manuel.lopez@owasp.org