Secure Coding, some simple steps help.

OWASP EU Tour 2013

OWASP

The Open Web Application Security Project

About Me

OWASP
The Open Web Application Security Project

- Steven van der Baan
- Dutch
- 7Safe, part of PA Consulting Group
  - Developer
  - Pentester
  - Consultant
- CISSP, OSCP


OWASP
The Open Web Application Security Project
QUESTIONS

OWASP

The Open Web Application Security Project

OWASP
The Open Web Application Security Project

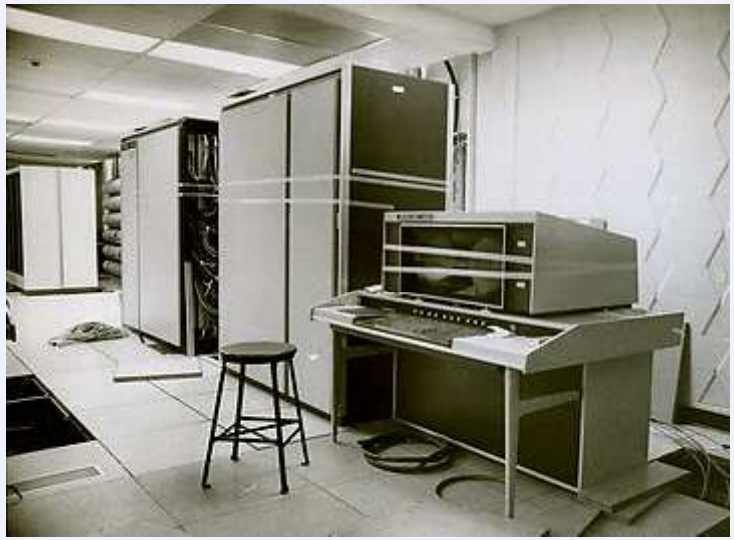
- “It's amazing how many drivers think the brakes are for slowing the car down”

- “Brakes allow you to travel faster because you have the power to stop.”

OWASP

The Open Web Application Security Project


WOB VB 467




Mainframe

OWASP

The Open Web Application Security Project


OWASP
The Open Web Application Security Project
SANS
WhiteHat
SECURITY

OWASP

The Open Web Application Security Project

T10 OWASP Top 10 Application Security Risks – 2013

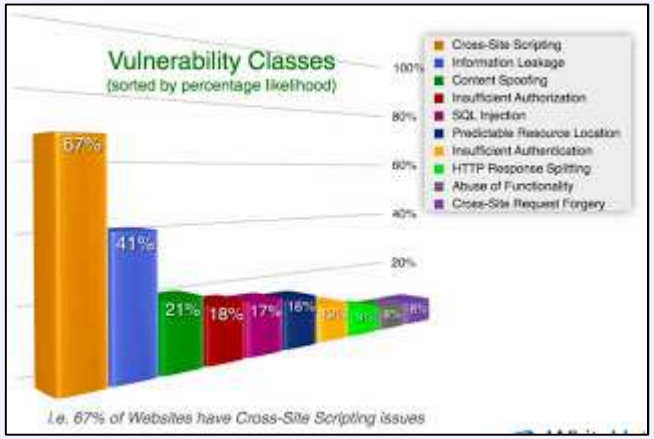
A1 – Injection	•Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing unauthorized data.
A2 – Broken Authentication and Session Management	•Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, session tokens, or exploit other implementation flaws to assume other users' identities.
A3 – Cross-Site Scripting (XSS)	•XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
A4 – Insecure Direct Object References	•A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
A5 – Security Misconfiguration	•Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. All these settings should be defined, implemented, and maintained as many are not shipped with secure defaults. This includes keeping all software up to date.
A6 – Sensitive Data Exposure	•Many web applications do not properly protect sensitive data, such as credit cards, tax ids, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct identity theft, credit card fraud, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.
A7 – Missing Function Level Access Control	•Virtually all web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access unauthorized functionality.
A8 - Cross-Site Request Forgery (CSRF)	•A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
A9 - Using Components with Known Vulnerabilities	•Vulnerable components, such as libraries, frameworks, and other software modules almost always run with full privilege. So, if exploited, they can cause serious data loss or server takeover. Applications using these vulnerable components may undermine their defenses and enable a range of possible attacks and impacts.
A10 – Unvalidated Redirects and Forwards	•Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

This is a brief listing of the Top 25 items, using the general ranking.

NOTE: 16 other weaknesses were considered for inclusion in the Top 25, but their general scores were not high enough. They are listed in a separate "On the Cusp" page.

Rank	Score	ID	Name
[1]	93.8	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[2]	83.3	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[3]	79.0	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[4]	77.7	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[5]	76.9	CWE-306	Missing Authentication for Critical Function
[6]	76.8	CWE-862	Missing Authorization
[7]	75.0	CWE-798	Use of Hard-coded Credentials
[8]	75.0	CWE-311	Missing Encryption of Sensitive Data
[9]	74.0	CWE-434	Unrestricted Upload of File with Dangerous Type
[10]	73.8	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[11]	73.1	CWE-250	Execution with Unnecessary Privileges
[12]	70.1	CWE-352	Cross-Site Request Forgery (CSRF)
[13]	69.3	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[14]	68.5	CWE-494	Download of Code Without Integrity Check
[15]	67.8	CWE-863	Incorrect Authorization
[16]	66.0	CWE-829	Inclusion of Functionality from Untrusted Control Sphere
[17]	65.5	CWE-732	Incorrect Permission Assignment for Critical Resource
[18]	64.6	CWE-676	Use of Potentially Dangerous Function
[19]	64.1	CWE-327	Use of a Broken or Risky Cryptographic Algorithm
[20]	62.4	CWE-131	Incorrect Calculation of Buffer Size
[21]	61.5	CWE-307	Improper Restriction of Excessive Authentication Attempts
[22]	61.1	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')
[23]	61.0	CWE-134	Uncontrolled Format String
[24]	60.3	CWE-190	Integer Overflow or Wraparound
[25]	59.9	CWE-759	Use of a One-Way Hash without a Salt

CWE-89 - SQL injection - delivers the knockout punch of security weaknesses in 2011. For data-rich software applications, SQL injection is the means to steal the keys to the kingdom. CWE-78, OS command injection, is where the application interacts with the operating system. The classic buffer overflow (CWE-120) comes in third, still pernicious after all these decades. Cross-site scripting (CWE-79) is the bane of web applications everywhere. Rounding out the top 5 is Missing Authentication (CWE-306) for critical functionality.



OWASP

The Open Web Application Security Project

OWASP
The Open Web Application Security Project

Security Principles

SECURITY NINJA

OWASP
The Open Web Application Security Project

Who..

..with what role..

..what rights.

..may do..

in which process?

OWASP

The Open Web Application Security Project

	Specific vulnerabilities for each principle		
	OWASP top 10	**WhiteHatSec top 10**	**Sans top 25**
Principles			
Input Validation	Cross Site Scripting, Injection Flaws, Malicious File Execution	Cross Site Scripting, SQL Injection, Content Spoofing	Improper Input Validation, Failure to Preserve SQL Query Structure, Failure to Preserve Web Page Structure, Failure to Preserve OS Command Structure, Failure to Constrain Operations within the Bounds of a Memory Buffer, Failure to Control Generation of Code, Client-Side Enforcement of Server-Side Security
Output Encoding	Cross Site Scripting	Cross Site Scripting	Improper Encoding or Escaping of Output, Failure to Preserve Web Page Structure
Error Handling	Information Leakage and Improper Error Handling	Information Leakage	Error Message Information Leak
Authentication and Authorisation	Broken Authentication and Session Management	Insufficient Authorisation, Insufficient Authentication, Abuse of Functionality	Impropert Access Control, Hard-Coded Password, Insecure Permission Assignment for Critical Resource, Execution with Unnecessary Privileges
Session Management	Broken Authentication and Session management, Cross Site Request forgery	Cross Site Request Forgery	Cross Site Request Forgery, Use of Insufficient Random Values
Secure Communications	Insecure Communications		Use of a Broken or Risky Cryptographic Algorithm, Cleartext Transmission of Sensitive Information, Use of Insufficiently Random Values
Secure Resource Access	Insecure Direct Object Reference, Failure to Restrict URL Access	Predictable Resource Location	External Control of File Name or Path, Untrusted Search Path
Secure Storage	Insecure Cryptographic Storage		Use of a Broken or Risky Cryptographic Algorithm, Cleartext Transmission of Sensitive Information, External Control of Critical State Data

Security principles

OWASP
The Open Web Application Security Project

- **Input Validation**
- **Output Encoding**
- **Error Handling**
- **Authentication and Authorisation**
- **Session Management**
- **Secure Communications**
- **Secure Resource Access**
- **Secure Storage**

OWASP

The Open Web Application Security Project

Input validation

Input Validation

OWASP
The Open Web Application Security Project

Input validation

OWASP
The Open Web Application Security Project

- Identify and define the data your application must accept
- Create regEx's to validate EACH datatype (**content and size**)
  - For example, a creditcard data type: **\d{12,16}$**
- Use whitelisting where possible
- Blacklist approach harder and potential less secure
  - Blacklist example, replace single quotes:
    - S.replaceAll(Pattern.quote("'"));
    - Matcher.quoteReplacement("'");

OWASP

The Open Web Application Security Project

Output Encoding

Output Encoding

OWASP
The Open Web Application Security Project

Output Encoding

OWASP
The Open Web Application Security Project

- Identify and define the data your application must output
- Understand where (**i.e. in an URL**) your data should end up
- Choose the correct output encoding for the data's destination
  - Proper encoding means this attack:
    - www.example.com/home.html?day=<script>alert(document.cookie)</script>

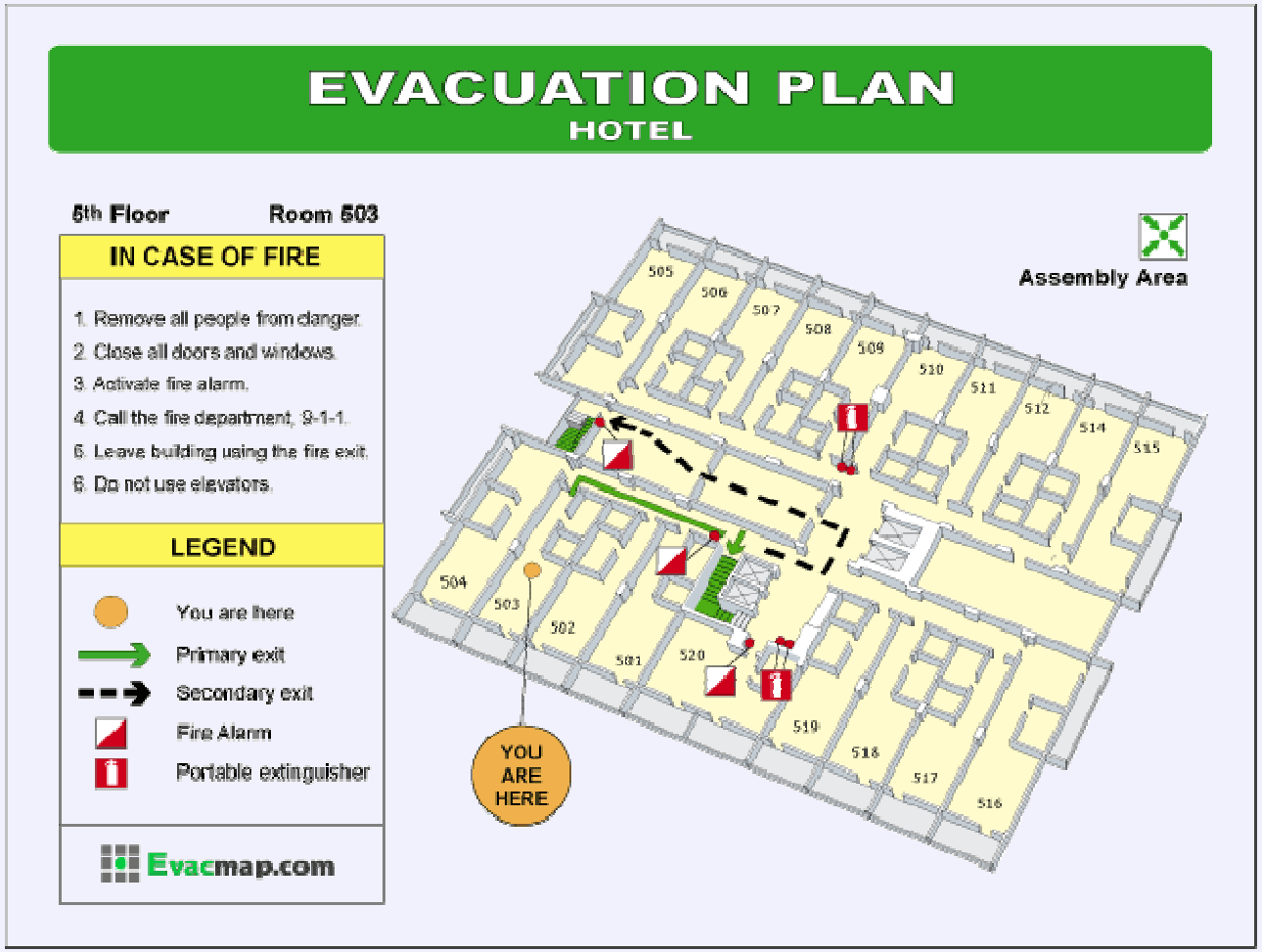
    Becomes
    - day=%3Cscript%3Ealert%28document.cookie%29%3C/script%3E

OWASP
The Open Web Application Security Project

Error Handling

Error Handling

OWASP
The Open Web Application Security Project



Error handling

OWASP
The Open Web Application Security Project

- Even the best apps will crash at some point, be prepared!
- Crashed/errors can help an attacker if you don't handle them
- Handle error conditions securely, sanitize the message sent
- No error handling = information leakage

Microsoft OLE DB Provider for ODBC Driver(0x80040E14)
[Microsoft][ODBC SQL Server Driver][SQL Server]Invalid column name

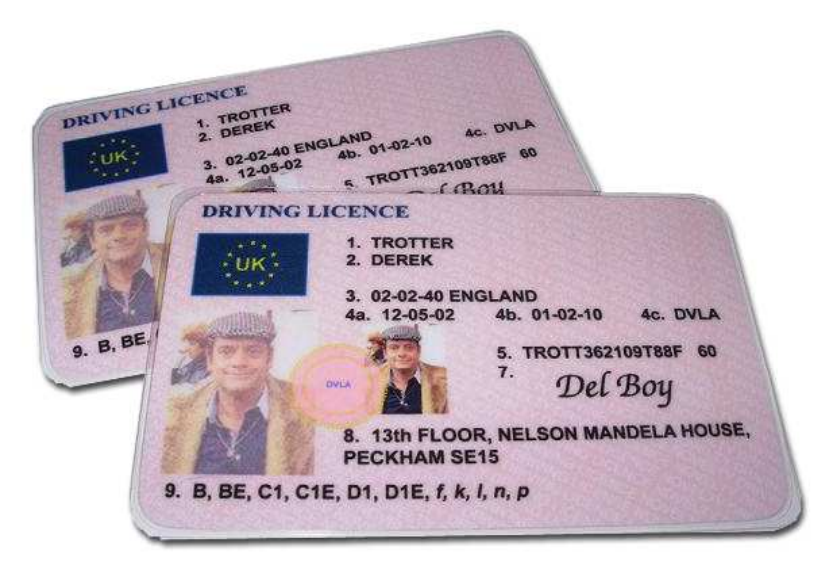
/example/login.asp, line 10

OWASP
The Open Web Application Security Project

Authentication & Authorization

Authentication and Authorisation

OWASP
The Open Web Application Security Project



Authentication and Authorisation

OWASP
The Open Web Application Security Project

- Even simple apps often have a need to authenticate users
- Often at least two levels of authorisation
- Need to prevent horizontal and vertical privilege escalation
- Implement strong passwords and management system
- Ensure A+A is secure, not a false sense of security (**CAPTCHA?**)
- Don't rely on fields that are easily spoofed (**refer(r)er field**)

OWASP
The Open Web Application Security Project

Session Management

Session Management

OWASP
The Open Web Application Security Project

Session Management

OWASP
The Open Web Application Security Project

- Used to manage authenticated users, no need to re-auth
- You need to make shure your sessionID's have sufficient entropy
- SessionID's must not be predictable or reusable
- Never build your own session management, it will fail
- Protect SessionID's when in transit (**i.e. SSL!**)
- Issue a new value for sensitive actions (**i.e. funds transfer**)

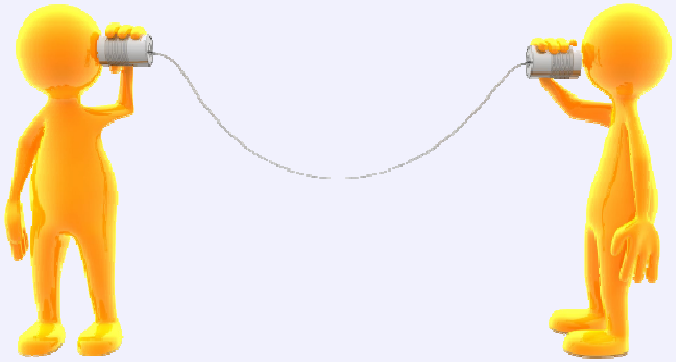
OWASP
The Open Web Application Security Project

Secure Communications

Secure Communications

OWASP

The Open Web Application Security Project



Secure Communications

OWASP
The Open Web Application Security Project

- Protect data (**i.e. CC no, passwords, sessionID's**) in transit
- As with all crypto, don't create your own
- Don't use broken protection mechanisms (**i.e. SSLv2**)
- Don't just use SSL/TLS for logon pages, protect the session!
- Try to avoid mixing secure and insecure traffic on a page

OWASP
The Open Web Application Security Project

Secure Resource Access

Secure Resource Access

OWASP
The Open Web Application Security Project

Secure Resource Access

OWASP
The Open Web Application Security Project

- Obscurity != security, don't try to hide sensitive resources
- Understand the users flow through an app, cover weak points
  - T-Mobile didn't do the above, Paris Hiltons account hacked

OWASP

The Open Web Application Security Project

Secure Storage

Secure Storage

OWASP
The Open Web Application Security Project

Secure Storage

OWASP
The Open Web Application Security Project

- Protect data (**i.e. CC no, passwords, sessionID's**) when stored
- As with all crypto, DON'T create your own
- Don't use broken protection mechanisms (**i.e. DES**)
- Don't store data in places where you can't confidently secure it
- Strong protection mechamisms, how strong should it be?

OWASP

The Open Web Application Security Project