OWASP Top 10

Introduction to web security
Objective

- Provide you with a quick introduction to web application security
- Increase your awareness and knowledge of security in general
- Show you that “any tester” can (and should) do security testing and not just leave it to the pen testers
What is Security?

- The traditional security areas of concern are:
  - Confidentiality
  - Integrity
  - Availability
  - Accountability.
- Security is dependent on context – different organisations have different needs.
Types of Security

- Physical – doors, walls, locks etc.
- Network – OS, network, firewalls etc.
- Application – software at the application layer

- Perceived vs. Actual security
  - Security theater
  - FUD – Fear Uncertainty & Doubt
  - Absolutism – nothing is ‘totally secure’
Attackers can potentially use many different paths through your application to do harm to your business or organization. Each of these paths represents a risk that may, or may not, be serious enough to warrant attention.
The term *Threat Agent* is used to indicate an individual or group that can manifest a threat. It is fundamental to identify who would want to exploit the assets of a company, and how they might use them against the company.
Attacks are the techniques that attackers use to exploit the vulnerabilities in applications.
A vulnerability is a hole or a weakness in the application, which can be a design flaw or an implementation bug, that allows an attacker to cause harm to the stakeholders of an application. The term "vulnerability" is often used very loosely.
What is a Control?

- Controls are defensive technologies or modules that are used to detect, deter, or deny attacks.
- A weakness or design flaw of a control, or the lack of a necessary controls results in a vulnerability that can make the application susceptible to attacks.
What is Technical impact?

- A technical impact is the system damage that results from a successful security breach. This is just the effect on the technology, not the business.
A business impact is the impact to a business that results from a successful attack. Generally this is in terms of money, lives, reputation, customers, or speed. The business risk is what justifies investment in fixing security problems.
A threat agent detects a weakness in the application and its controls. The agent launches an attack that causes both technical and business impacts.
OWASP Top 10

- OWASP provide a list of the top ten ‘risks’.
- The names of the risks in the Top 10 stem from the type of attack, the type of weakness, or the type of impact they cause. OWASP chose the name that is best known and will achieve the highest level of awareness.
- The primary aim of the OWASP Top 10 is to educate.
How did OWASP Rank Risks?

- The basic standard risk model is:
  \( \text{Risk} = \text{Likelihood} \times \text{Impact} \)

- The OWASP approach used in this document is based on standard methodologies and is customized for application security.
Injection flaws, such as SQL, OS, and LDAP injection, occur when un-trusted 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.
http://iona.net.nz/leaveMsg.asp
http://iona.net.nz/leaveMsgPost.asp

LeaveMsg.asp
• Allows user to enter a message for the bank
• Only one message per user at a time

confirmMsg.asp
• Saves message into queue (form’s target)
• Confirms message saved

Email.asp
• Generates HTML for email to user
SQL Injection

Hi, this is your son’s school. We’re having some computer trouble.

Oh, dear — did he break something? In a way —

DID YOU REALLY NAME YOUR SON Robert’); DROP TABLE Students;-- ?

Oh, yes. Little Bobby Tables, we call him.

Well, we’ve lost this year’s student records. I hope you’re happy.

And I hope you’ve learned to sanitize your database inputs.

http://xkcd.com/327/
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.
New threats - Hash Collision

SHA1 collider

Quick-and-dirty PDF maker using the collision from the SHAtered paper.

Choose two image files (must be JPG, same aspect ratio). For now, the JPEGs must be less than 64kB.

[Input fields for aspect ratio and two file selection options with no file selected]
XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation and 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.
LeaveMsgPost.asp
- Allows user to enter a message for the bank
- Only one message per user at a time
- Form uses Post Method

XSS Me – Cross Site Scripting checker

confirmMsg.asp
- Saves message into queue (form’s target)
- Confirms message saved
Example – XSS via Wi-Fi SSID

Vulnerability Note VU#101462
DrayTek Vigor 2700 ADSL router contains a command injection vulnerability

Overview
DrayTek Vigor 2700 ADSL router version 2.8.3 and possibly earlier versions contain a command injection vulnerability via malicious SSID (CWE-77).

Description
CWE-77: Improper Neutralization of Special Elements used in a Command ("Command Injection")

DrayTek Vigor 2700 ADSL router version 2.8.3 contains a command injection vulnerability. The Vigor router stores discovered SSIDs of surrounding access points into the variables.js file, which is a list of multiple variables used by the web administration console. An attacker can utilize JavaScript code in the SSID value of a malicious nearby access point which will cause the DrayTek router to execute the command when it adds the SSID into the variables.js list.

Impact
An attacker within range of the DrayTek Vigor ADSL router can edit the SSID on their malicious access point to corrupt the variables.js file. This may cause the DrayTek router to call external scripts or make unauthorized changes to the local configuration.

Quick Search

View Notes By
- Date Published
- Date Updated
- CVSS Score

Report a Vulnerability
Please use the Vulnerability Reporting Form to report a vulnerability. Alternatively, you can send us email. Be sure to read our vulnerability disclosure policy.
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.
Example Ecan bus website

Demo of how you might have hacked the site using the command line and Excel

My card number is 733731 and I can see my balance at [http://iona.net.nz/bus.asp?id=733731](http://iona.net.nz/bus.asp?id=733731)
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.
Example – Default Passwords

- http://www.defaultpassword.com

- Default password for DrayTek Vigor 2710 modem:

<table>
<thead>
<tr>
<th>Username</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin</td>
<td>admin</td>
</tr>
<tr>
<td>IP Address</td>
<td>192.168.1.1</td>
</tr>
<tr>
<td>SSID</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Many web applications do not properly protect sensitive data, such as credit cards, SSNs, and authentication credentials, with appropriate encryption or hashing. Attackers may steal or modify such weakly protected data to conduct identity theft, credit card fraud, or other crimes.
A6 – Sensitive Data Exposure Examples

Server Error in '/' Application.

Conversion failed when converting from a character string to uniqueIdentifier.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: Conversion failed when converting from a character string to uniqueIdentifier.

Source Error:
An unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of the exception can be identified using the exception stack trace below.

Stack Trace:

```
Conversion failed when converting from a character string to uniqueIdentifier.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: Conversion failed when converting from a character string to uniqueIdentifier.

Source Error:
An unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of the exception can be identified using the exception stack trace below.

Stack Trace:
```
Not all users should have access to all functions. Sometimes, function level protection is managed via configuration, and the system is misconfigured or sometimes, developers forget to include the proper checks.
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 forces the victim’s browser to generate requests application thinks are legitimate.
DEMO – CSRF
Login.asp
• User logs in and is given session cookie

User opens another page in new browser tab
• img tab sends request “on behalf of user” to one of routers CGI pages

Router system processes message as if it was a user request.
A9 – Using Components with Known Vulnerabilities

- Virtually every application has these issues because most development teams don’t focus on ensuring their components/libraries are up to date. In many cases, the developers don’t even know all the components they are using, never mind their versions. Component dependencies make things even worse.
Example:
A10 – Unvalidated Redirects and Forwards

- Web applications frequently redirect and forward users to other pages and websites, and use un-trusted 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.

<table>
<thead>
<tr>
<th>Threat Agents</th>
<th>Attack Vector</th>
<th>Weakness</th>
<th>Technical Impacts</th>
<th>Business Impacts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exploitability</td>
<td>AVERAGE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prevalence</td>
<td>UNCOMMON</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>detectability</td>
<td>EASY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Impact</td>
<td>MODERATE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
http://iona.net.nz/home.asp

Link to site in email
• User is not logged in
  redirect to login.asp

Login.asp
• User logs in and is given session cookie

Surprise!
The OWASP Top 10 is a place to start. It does **not** define all the possible threats.

The Top 10’s roll is to **educate** and provide an **introduction** a number of critical threats.

Testing for weaknesses does not provide as much assurance as having a secure development lifecycle.

Build security in don’t “bolt it on”. 
Tools used in demos

- **Firebug**
  - [https://getfirebug.com](https://getfirebug.com)

- **HttpFox**

- **Tamper Data**

- **XSS Me**

- **HackBar**
Piggy bank’s pages

- Login.asp – Access for authorised users
  - admin admin or user password
- Home.asp – Home of piggy bank’s Intranet
- LeaveMsg.asp – Leave a message (GET)
- LeaveMsgPost.asp – Leave a message (POST)
- ConfirmMsg.asp – confirms message
- Email.asp – generates HTML for email
- DisplayMsg.asp – Banks view of waiting messages

- Base URL http://iona.net.nz/