OWASP Top 10 Risks

Dean.Bushmiller@ExpandingSecurity.com

Many thanks to Dave Wichers & OWASP
My Mom

• I got on the email and did a google on my boy
• My boy works in this Internet thing
• He makes cyber cafes a safe place for me
• He speaks a whole different language  
  – It is called TCIP
• He is an Internet security teacher

Awww! Mom, you got it all screwed up.
Goals

• High level overview of the Top 10 in 45 min.
  – Definition, Illustration, & Protection/Avoidance
• With abstraction comes loss of clarity
• Spend a whole life/week/day/hour
• Explain it to me like
  – You should be able to explain it to someone else
  – Your mom can get it
  – Your boss’s boss will care
The Top 10 Web Risks

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
A7: Insecure Cryptographic Storage
A8: Failure to Restrict URL Access
A9: Insufficient Transport Layer Protection
A10: Unvalidated Redirects and Forwards
A1: Injection
A1 – Injection

Defined
- commands inserted in the data, interpreted by interpreter

Command Interpreters
- Perl, OS Shell, LDAP, Xpath...

SQL injection
- Web site are connected to Databases

Impact
- Impact Availability, Confidentiality, Integrity
SQL Injection – Illustrated

1. Application presents a form to the attacker
2. Attacker sends an attack in the form data
3. Application forwards attack to the database in a SQL query
4. Database runs query containing attack and sends results back to application
5. Application sends results to the user
A1 – Avoiding Injection

• Encode all user input before passing it to the interpreter
  – Everything that is data, only treat it as data
• Avoid using interpreter
• Reduce amount of data available
  – Least Privilege
A2: Cross-Site Scripting (XSS)
A2 – Cross-Site Scripting (XSS)

**Defined**
- Evil raw data from attacker is sent to an innocent user’s browser

**Raw data...**
- Link retrieved from valid / trusted web site
- Link is a request for data sent directly to client

**Web 2.0**
- Everyone posts to everyone else’s site
- You can not get around it

**Impact**
- Confidentiality, Integrity
- Attacker may observe and direct all user’s behavior
Cross-Site Scripting Illustrated

1. Attacker sets the trap – update my profile
   - Attacker enters a malicious script into a web page that stores the data on the server

2. Victim views page – sees attacker profile
   - Script runs inside victim’s browser with full access to the DOM and cookies

3. Script silently sends attacker Victim’s session cookie

Application with stored XSS vulnerability

Custom Code
- Accounts
- Finance
- Administration
- Transactions
- Communication
- Knowledge Mgmt
- E-Commerce
- Bus. Functions
A2 – Avoiding XSS Flaws

• Don’t include user-supplied input in the output page
• Convert all user-supplied input to data only
• Whitelist input validation on all user input
• Use OWASP’s AntiSamy to sanitize this HTML
A3: Broken Authentication and Session Management
# A3 – Broken Authentication and Session Management

## Defined
- Convert user name & password to Session ID
- If attacker can predict Session ID, they can steal it

## Session management flaws
- Credentials must be appended with every request
- SESSION ID is used to track state since HTTP doesn’t
- SESSION ID is typically exposed

## Other session entry points
- Change my password, remember my password, forgot my password, logout, email address, etc...

## Impact
- Confidentiality, Integrity
- Accounts compromised or sessions hijacked
Broken Authentication Illustrated

1. User sends credentials

2. Site uses URL rewriting (i.e., put session in URL)

3. User clicks on a link to http://www.hacker.com in a forum

4. Hacker checks referer logs on www.hacker.com and finds user’s JSESSIONID

5. Hacker uses JSESSIONID and takes over victim’s account
A3 – Avoiding Broken Authentication and Session Management

• Architecture
  – Authentication = simple, centralized, and standardized
  – SSL from cradle to grave

• Implementation
  – Check your SSL certificate
  – Examine all the authentication-related functions
  – Verify that logoff actually destroys the session
  – WebScarab to test
  – No automated analysis
A4: Insecure Direct Object References
A4 – Insecure Direct Object References

**Defined**
- Internal files and executables lead to other internal sensitive functions
- Attacker tampers with parameter value

**Flaws**
- Listing the ‘authorized’ objects
- Hiding the object references

**Impact**
- Confidentiality
- Attackers are able to access unauthorized files or data
Insecure Direct Object References Illustrated

- Attacker notices his acct parameter is 6065
  \( ?\text{acct}=6065 \)

- He modifies it to a nearby number
  \( ?\text{acct}=6066 \)

- Attacker views the victim’s account information

https://www.onlinebank.com/user?acct=6065
A4 – Avoiding Insecure Direct Object References

- Eliminate the direct object reference
  - temporary mapping value (e.g. 1, 2, 3)
- Validate the direct object reference

http://app?file=1
http://app?id=9182374
http://app?id=7d3J93

Internal Access Reference Map

Reference Monitor

Doc Request 1

External Request

Acct:9182374

SERVER

Client
A5: Cross-Site Request Forgery (CSRF)
A5 – Cross Site Request Forgery (CSRF)

**Defined**
- Victim’s browser is tricked into issuing a command to a vulnerable web application under attacker’s control

**Cause**
- Browsers automatically including user authentication data with each request

**Automatically Provided Credentials**
- Session cookie, Basic authentication header, IP address
- Client side SSL certificates
- Windows domain authentication

**Impact**
- Confidentiality, Integrity
- Initiate transactions
- Access sensitive data
Attacker sets the trap on some website on the internet (or simply via an e-mail)

While logged into vulnerable site, victim views attacker site

Hidden <img> tag contains attack against vulnerable site

<img> tag loaded by browser – sends GET request (including credentials) to vulnerable site

Vulnerable site sees legitimate request from victim and performs the action requested
A5 – Avoiding CSRF Flaws

• Add a secret, not automatically submitted

• Tokenize to ALL sensitive requests
  – Cryptographically strong or random

• Don’t allow attackers to store attacks on your site
  – Properly encode all input on the way out
  – This renders all links/requests inert in most interpreters
A6: Security Misconfiguration
A6 – Security Misconfiguration

Defined
• Unpatched operating systems and applications are an attack vector

Other code
• Anything you install is an attack vector

Impact
• Availability, Confidentiality, Integrity
Security Misconfiguration Illustrated

- Accounts
- Finance
- Administration
- Transactions
- Communication
- Knowledge Mgmt
- E-Commerce
- Bus. Functions
- Custom Code
- App Configuration
- Framework
- App Server
- Web Server
- Hardened OS
- Database

Insider
- Development
- QA Servers
- Test Servers
- Source Control
A6 – Avoiding Security Misconfiguration

• Hardening
  – Operating System
  – Utilities
  – Applications
  – Agents
• Patch
• Change Control
A7: Insecure Cryptographic Storage
A7 – Insecure Cryptographic Storage

**Defined**
- Unidentified sensitive data at rest

**Data**
- Databases, files, directories, log files, backups...

**Impact**
- Confidentiality, Integrity
- Expense of cleaning up the incident
- Sued and/or fined
Insecure Cryptographic Storage Illustrated

1. Victim enters credit card number in form

2. Error handler logs CC details because merchant gateway is unavailable

3. Logs are accessible to all members of IT staff for debugging purposes

4. Malicious insider steals 4 million credit card numbers

Log files
A7 – Avoiding Insecure Cryptographic Storage

• Identify all sensitive data and locations
• Encryption as much as you can afford
• Use the mechanisms correctly
  – Use standard strong algorithms
  – Generate, distribute, and protect keys properly
  – Be prepared for key change
• Verify and test
A8: Failure to Restrict URL Access
A8 – Failure to Restrict URL Access

**Defined**
- If authentication is used on any part of the site and not on all parts of the site

**A common mistake ...**
- Displaying only authorized links and menu choices
- Attacker types the URL directly

**Impact**
- Confidentiality
- Access other user’s accounts and data
- Perform privileged actions
Failure to Restrict URL Access Illustrated

- Attacker notices the URL indicates his role
  `/user/getAccounts`

- He modifies it to another directory (role)
  `/admin/getAccounts`, or
  `/manager/getAccounts`

- Attacker views more accounts than just their own
For each URL, a site needs to do 3 things:
- Restrict access to authenticated users (if not public)
- Enforce any user or role-based permissions (if private)
- Completely disallow requests to unauthorized page types

Verify the server configuration disallows requests to unauthorized file types

Use WebScarab to forge unauthorized requests
A9: Insufficient Transport Layer Protection
A9 – Insufficient Transport Layer Protection

Define

• Sensitive data is transmitted in clear

SSL & TLS

• Server side certificate normal
• Client side certificates are rare
• Server to server is possible

Impact

• Confidentiality
• Attackers use data as launching point for further attack
Insufficient Transport Layer Protection Illustrated

1. **External attacker** steals credentials and data off network.

2. **Internal attacker** steals credentials and data from internal network.

- **External Victim**
- **Custom Code**
- **Backend Systems**
- **Business Partners**
- **Employees**
- **External Attacker**
- **Internal Attacker**
A9 – Avoiding Insufficient Transport Layer Protection

• Use SSL/TLS on all connections with sensitive data

• Use certificates correctly
  – Use current standard strong algorithms
  – Manage keys/certificates properly

• Client side
  – Verify SSL certificates before using them
A10: Unvalidated Redirects and Forwards
### A10 – Unvalidated Redirects and Forwards

<table>
<thead>
<tr>
<th>Defined</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-supplied parameters (Controlled by the Attacker) in the destination URL request data from unauthorized sites</td>
</tr>
<tr>
<td>Attacker can send victim to a site of their choice</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Forwards</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internally send the request to a new page in the same application</td>
</tr>
<tr>
<td>Sometimes parameters define the target page</td>
</tr>
</tbody>
</table>

| Transfer in .NET |

<table>
<thead>
<tr>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrity</td>
</tr>
<tr>
<td>Redirect victim to phishing or malware site</td>
</tr>
</tbody>
</table>
Unvalidated Redirect Illustrated

1. Attacker sends attack to victim via email or webpage

From: Internal Revenue Service
Subject: Your Unclaimed Tax Refund
Our records show you have an unclaimed federal tax refund. Please click here to initiate your claim.

2. Victim clicks link containing unvalidated parameter

3. Application redirects victim to attacker’s site

4. Evil site installs malware on victim, or phish’s for private information

Unvalidated Forward Illustrated

1. Attacker sends attack to vulnerable page they have access to

Request sent to vulnerable page which user does have access to. Redirect sends user directly to private page, bypassing access control.

2. Application authorizes request, which continues to vulnerable page

3. Forwarding page fails to validate parameter, sending attacker to unauthorized page, bypassing access control

```java
public void doPost(HttpServletRequest request, HttpServletResponse response) {
    try {
        String target = request.getParameter("dest");
        ...
        request.getRequestDispatcher(target).forward(request, response);
    }
    catch ( ... )
}
```

```java
public void sensitiveMethod(HttpServletRequest request, HttpServletResponse response) {
    try {
        // Do sensitive stuff here.
        ...
    }
    catch ( ... )
}
```
A10 – Avoiding Unvalidated Redirects and Forwards

• Avoid using redirects and forwards
• Do not involve user parameters in defining the target URL
• If you ‘must’ involve user parameters
  – Validate each parameter
  – Server side mapping
What I do for a living

www.ExpandingSecurity.com

We protect your business on the Internet
By teaching managers and technologists
How to integrate security into your business

• CISSP, ISSMP, ISSAP, CEH, Penetration testing, packet analysis, network security, business security

• Come get a pocket protector at my booth