OWASP Top 10
Attacks & Countermeasures

Sherif Koussa

www.softwaresecured.com
About Your Instructor

• 10 Years of Experience Writing Enterprise Software
• 8 Years of Experience in the Application Security Field
• OWASP Ottawa Chapter Leader
• WASC SATEC Project Leader
• Founder of Software Secured, a boutique application security company in Ottawa, Canada.
Exercise: What’s Common Between the Following Names?

A1. INJECTION
A1 – Injection Flaws

- **Exploitability:** EASY
- **Prevalence:** COMMON
- **Detectability:** AVERAGE
- **Impact:** SEVERE

- Occurs 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.
- Examples of injection flaws include:
  - SQL Injection
  - LDAP Injection
  - Command Injection
  - XML Injection
Injection Theory

Untrusted Data → Injection Context → Parser
Injection Types

- SQL Injection
- Command Injection
- XML Injection
- LDAP Injection
- XPath Injection
- Code Injection
Anatomy of an SQL Injection Attack

Consider the following code snippet, what is the problem?

```java
public static ResultSet getReviews(Connection con, String dbName, HttpServletRequest request) throws SQLException {
    ResultSet rst = null;
    Statement stmt = con.createStatement();
    rst = stmt.executeQuery("SELECT * FROM Reviews WHERE ProductID = " + request.getParameter("id"));
    return rst;
}
```

Resulting SQL:

```
SELECT * FROM Reviews WHERE ProductID = 12
```
Anatomy of an SQL Injection Attack

Consider the following code snippet, what is the problem?

```java
public static ResultSet getReviews(Connection con, String dbName, HttpServletRequest request) throws SQLException {
    ResultSet rst = null;
    Statement stmt = con.createStatement();
    rst = stmt.executeQuery("SELECT * FROM Reviews WHERE ProductID = " + request.getParameter("id"));
    return rst;
}
```

Resulting SQL:

```
SELECT * FROM Reviews WHERE ProductID = 12 UNION SELECT * FROM USERS;
```
A2. BROKEN AUTHENTICATION & SESSION MANAGEMENT
A2 - Broken Authentication & Session Management

- Exploitability: AVERAGE
- Prevalence: COMMON
- Detectability: AVERAGE
- Impact: SEVERE

- Usually the result of an inadequately developed login page, authentication logic, backdoor, custom session management or authentication scheme.
- Frequently has flaws in areas such as logout, password management, timeouts, remember me, secret question, account update, etc.
Authentication Process

Collection

Replacement

Transmission

Validation

Storage
Authentication Process - Collection

- **Username Enumeration:**
  - Is an attack that aims at enumerating valid usernames that can be used later for a DoS attack or a Brute Force attack.

- **Caused By:**
  - Application trying to be too usable
  - Error Message at Login
Consider the following example:


• The application supports session URL rewriting
• URLs are shared, logged and emailed
• This could allow other users to hijack the session
Authentication Process - Storage

- **Insecure Password Storage:** happens when users’ passwords are stored in an insecure manner inside the database; this could lead to unauthorized access to users’ accounts in case of a data breach.

- **Caused By:**
  - Storing passwords in plaintext format.
  - Storing passwords using reversible encryption algorithm.
  - Using weak hashing algorithm.
  - Lack of SALTs
Authentication Process - Validation

Consider the following example. What is the issue?

```java
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    try {
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        Boolean isFound = fetchUser(username, password);
        if (!isFound){
            response.sendRedirect("/login");
            log("Login Failed");
        }
    }
    catch (Exception ex) {
        log(ex.getMessage());
    }
    response.sendRedirect("/homepage");
    log("Login Succeeded");
}
```
Authentication Process - Validation

Consider the following example:

```java
protected void doPost2(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    try {
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        Boolean isFound = fetchUser(username, password);
        if (isFound)_boolean.getboolean(request.getParameter("isAdmin")) {
            response.sendRedirect("/login");
            log("Login Failed");
        }
    } catch (Exception ex) {
        log(ex.getMessage());
    }
    response.sendRedirect("/homepage");
    log("Login Succeeded");
}
```

Setting isAdmin=true will give admin user access!
Authentication Process - Validation

Consider the following example:

```java
String[] ids = {"334364642945","94985984723","98759390257"};
protected void doPost(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {

  try {
    String username = request.getParameter("username");
    String password = request.getParameter("password");

    Boolean isFound = fetchUser(username, password);
    if (isFound && Arrays.asList(ids).contains(username)) {
      response.sendRedirect("/homepage");
      log("Login Succeeded");
    }
  }
  catch (Exception ex) {
    log(ex.getMessage());
  }
  response.sendRedirect("/login");
  log("Login Failed");
```

User granted unconditional access based on a hardcoded ID!
Authentication Process - Replacement

• **Insecure Change Password**: occurs when the application does not tighten security controls around the change password process.

• **Caused By:**
  – Failure to require the old password before accepting the new one.
  – Failure to enforce a strong password history policy.
Authentication Process - Replacement

• **Insecure Forgot-Password Mechanism:** occurs when the application does not tighten security controls around the forgotten password process.

• **Caused By:**
  – Displaying the password after answering a security question that has a small set of answers (e.g. what’s your favorite color?)
  – Sending the username and password together in the same email.
How Weak are Weak Passwords?

Open Security Research
Sponsored by Foundstone

Brute Force Calculator

Password Length: 10
Keys per second: 7680K k/s
Charset [len:69]: alphabet-numeric-all-space

To brute force the entire keyspace it will take about 10 thousand years

More specifically: 10327 years 306 days 10 hours 24 minutes and 30 seconds (2402167502723212300 password combinations)

http://calc.opensecurityresearch.com/
A3. CROSS-SITE SCRIPTING
Anatomy of a Cross-site Scripting Attack

Consider the following example:

```
https://mysite.com?user=skoussa
```

```html
<html>
<head>
<title>Welcome to Secure App</title>
</head>
<body>
<div>Welcome <%= request.getParameter("username") %></div>
</body>
</html>
```

The result would be:

```html
<html>
<head>
<title>Welcome to Secure App</title>
</head>
<body>
<div>Welcome skoussa</div>
</body>
</html>
```
Anatomy of a Cross-site Scripting Attack

Consider the following malicious input:

```
https://mysite.com?username=user<script>alert(document.cookie)</script>
```

The result would be:

```
<html>
<head>
<title>Welcome to Secure App</title>
</head>
<body>
<div>Welcome % request.getParameter("username") %</div>
</body>
<html>
```
Cross-site Scripting Attacks - Reflected

http://vulnerableapp.com/welcome?msg=<script>alert(1)</script>

<html>
  <head></head>
  <body>
    ...snip...
    <script>alert(1)</script>
  </body>
</html>
Cross-site Scripting Attacks - Stored

http://vulnerableapp.com/addComment?msg=<script>alert(1)</script>

http://vulnerableapp.com/getComments

<html>
...snip...
<script>alert(1)</script>
</html>

Insert into comments values (1, “<script>alert(1)</script>”);

Select * from comments;
A4. INSECURE DIRECT OBJECT REFERENCES
A4 – Insecure Direct Object References

• Applications frequently use the actual name or key of an object when generating web pages.
• Applications don’t always verify the user is authorized for the target object.
• This results in an insecure direct object reference flaw.

Exploitability: EASY
Prevalence: COMMON
Detectability: EASY
Impact: MODERATE
Insecure Direct Object References - Example

Consider the following code, what is the problem?


```javascript
function doGet(request, response) {
    // Get the user's logon status
    var isLoggedIn = checkStatus(request.sessionId);
    // Get the account user ID
    var accountId = request.getUser();
    // SQL prepared/parameterized query
    var query = sql.Prepare('SELECT * FROM accounts WHERE accountId=?');
    if (isLoggedIn) {
        // Get account info
        var results = query.fetchResults(accountId);
        for (a in results) {
            response.write(a);
        }
    } else {
        // Otherwise, redirect to error page
        showAccessDenied(request, response);
    }
}
```
Insecure Direct Object References - Example

Consider the following example:

```javascript
function doGet(request, response) {
  // Get the user's logon status
  var isLoggedIn = checkStatus(request.sessionId);
  // Get the account user ID
  var accountId = request.getParam('user');
  // SQL prepared/parameterized query
  var query = sql.Prepare('SELECT * FROM accounts WHERE accountId=?');
  if (isLoggedIn == true) {
    // Get account info
    var results = query.fetchResults(accountId);
    for (a in results) {
      response.write(a);
    }
  } else {
    // Otherwise, redirect to error page
    showAccessDenied(request, response);
  }
}
```

https://vulnerable.com/viewAccount.php?accountId=87776

This will result in gaining unauthorized access to account ID 87776
Nokia: Insecure Direct Object Reference

• In 2014, Nokia had a vulnerability in their Account Payable Portal.
• An attacker could manipulate a consecutive id to gain unauthorized access to other company data.
• The data included things such as:
  – Company name
  – VAT#
  – Full Address
  – PO number
  – The ability to Cancel the request
Nokia: Insecure Direct Object Reference

Nokia: Insecure Direct Object Reference

A5. SECURITY MISCONFIGURATION
A5 – Security Misconfiguration

- Exploitability: EASY
- Prevalence: COMMON
- Detectability: EASY
- Impact: MODERATE

- Can happen at any level of an application stack, including the platform, web server, application server, framework, and custom code.
- Examples of misconfiguration include using default accounts, enabling unnecessary services, missing patches, storing clear-text passwords, etc.
Example – Detailed Error Pages

Consider the following example, what is the problem?

Server Error in '/' Application.

Cannot open database "..." requested by the login. The login failed. Login failed for user '...'.

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.


Source Error:

Line 16: corDB = New SqlConnection(session("strDBSQL"))
Line 17: corDB = New SqlConnection(session("strDBSQL"))
Line 19: corDB = New SqlConnection(session("strDBSQL", "corDB"))

Source File: Line: 10

Stack Trace:

[SqlException (0x80131904): Cannot open database "..." requested by the login. The login failed.]
Login failed for user '...'.
System.Data.ProviderBase.DbConnectionPool.GetConnection(DbConnection owningObject) +437
System.Data.ProviderBase.DbConnectionFactory.GetConnection(DbConnection owningConnection) +82
System.Data.ProviderBase.DbConnectionClosed.OpenConnection(DbConnection outerConnection, DbConnectionFactory connectionFactory) +105
myCMS.SessionClass.myCMS_SessionInit() in
ASP_INDEX.aspx.Page_Load() in
System.Web.UI.Control.OnLoad(EventArgs e) +99
System.Web.UI.Control.LoadRecursive() +47
System.Web.UI.Page.ProcessRequestMain(Boolean includeStagesBeforeAsyncPoint, Boolean includeStagesAfterAsyncPoint) +1061

Version Information: Microsoft .NET Framework Version:2.0.50727.42; ASP.NET Version:2.0.50727.42
Consider the following example, what is the problem?

```
HTTP Status 500 -

Description: The server encountered an internal error () that prevented it from fulfilling this request.

An error occurred at lines 22 in the JSP file: /admin/index.jsp
The method onRequestHandler() in the type Solicitor is not applicable for the arguments (Class<ReplicationHandler>)
20: <%-- jsp:include page="header.jsp"/ %> 30: <%-- do a verbatim include so we can use the local vars -->%
21: <%#include file="header.jsp" %>
22: <boolean replicationHandler = !core.getRequestHandler(ReplicationHandler.class).isNotEmpty();%>
23: %if clear="all">%
24: <table>
25:

Stacktrace:
org.apache.jasper.compiler.DefaultErrorHandler.javacError(DefaultErrorHandler.java:92)
org.apache.jasper.compiler.ErrorDispatcher.javacError(ErrorDispatcher.java:396)
org.apache.jasper.compiler.JDTCompiler.generateClass(JDTCompiler.java:419)
org.apache.jasper.compiler.Compiler.compile(Compiler.java:388)
org.apache.jasper.compiler.Compiler.compile(Compiler.java:327)
org.apache.jasper.compiler.Compiler.compile(Compiler.java:314)
org.apache.jasper.JspCompilationContext.compile(JspCompilationContext.java:509)
org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:317)
org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:313)
org.apache.jasper.servlet.JspServlet.service(JspServlet.java:260)
java.servlet.http.HttpServlet.service(HttpServlet.java:717)
org.apache.catalina.core.StandardWrapper.service(StandardWrapper.java:253)
org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:295)
org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:106)
org.apache.catalina.core.StandardFilterValve.invoke(StandardFilterValve.java:196)
org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:116)
org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:350)
org.apache.tomcat.util.net.JIoEndpoint$Worker.run(JIoEndpoint.java:986)
```

The full stack trace of the root cause is available in the Apache Tomcat/6.0.26 logs.
Consider the following example, what is the problem?

<table>
<thead>
<tr>
<th>File Name</th>
<th>Date Modified</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sitting-area-150x150.jpg</td>
<td>27-Mar-2011 12:43</td>
<td>32K</td>
</tr>
<tr>
<td>Sitting-area-425x332.jpg</td>
<td>27-Mar-2011 12:43</td>
<td>28K</td>
</tr>
<tr>
<td>Sitting-area.jpg</td>
<td>27-Mar-2011 12:43</td>
<td>28K</td>
</tr>
<tr>
<td>Social-Media-Passwords.xls</td>
<td>17-Mar-2011 17:29</td>
<td>70K</td>
</tr>
<tr>
<td>StPatricksDay-150x150.jpg</td>
<td>17-Mar-2011 12:37</td>
<td>12K</td>
</tr>
<tr>
<td>StPatricksDay-300x171.jpg</td>
<td>17-Mar-2011 12:37</td>
<td>23K</td>
</tr>
<tr>
<td>StPatricksDay.jpg</td>
<td>17-Mar-2011 12:37</td>
<td>94K</td>
</tr>
<tr>
<td>StPatricksDay1.jpg</td>
<td>17-Mar-2011 17:28</td>
<td>94K</td>
</tr>
<tr>
<td>StPatricksDay2.jpg</td>
<td>17-Mar-2011 17:29</td>
<td>94K</td>
</tr>
<tr>
<td>StPatricksDay3-150x150.jpg</td>
<td>17-Mar-2011 17:41</td>
<td>12K</td>
</tr>
<tr>
<td>StPatricksDay3-300x171.jpg</td>
<td>17-Mar-2011 17:41</td>
<td>23K</td>
</tr>
</tbody>
</table>
Example – Environment Misconfiguration

Unintended secrets are publicly available:

![Spreadsheet with example data]
Example – Environment Misconfiguration

Consider the following example, what is the problem?
A6. SENSITIVE DATA EXPOSURE
A6 – Sensitive Data Exposure

- Exploitability: DIFFICULT
- Prevalence: UNCOMMON
- Detectability: AVERAGE
- Impact: SEVERE

- Lacking or improper encryption for credentials and sensitive data.
- Homegrown cryptographic algorithms.
- Lacking or improper protection for data in transit or at rest.
Sensitive Data Exposure – Data at Rest

• Sensitive data stored in the database
  – Credit card numbers
  – Account passwords
  – Personally identifiable information (PII)
Sensitive Data Exposure – Data at Rest

- Sensitive data stored on the file system
  - Unencrypted database backups
  - Confidential files
Symmetric Encryption: is a class of encryption algorithms that uses the same encryption key for encryption and decryption.
Cryptography Principals - Symmetric Encryption

• Symmetric Encryption:
  – Pros:
    • Faster algorithm.
  – Cons
    • Key Management
  – Best Suited For:
    • Encrypting data at rest
  – Algorithms
    • AES
    • 3DES
Asymmetric Encryption: AKA public-key cryptography is a class of encryption algorithms that uses a public key for encryption and a private key for decryption.
Cryptography Principals - Asymmetric Encryption

• Asymmetric Encryption:
  – Pros:
    • Easier Key Management
  – Cons
    • Slower
  – Best Suited For:
    • Encrypting data in transit
  – Algorithms
    • RSA
Cryptography Principals - Hashing

A hash function is any algorithm or subroutine that maps large data sets of variable length, to smaller data sets of a fixed length.
Different Hash Functions

Different hashes calculated for “Hello There!”

<table>
<thead>
<tr>
<th>Original Text</th>
<th>Hello There!</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adler32</td>
<td>1bf4042e</td>
</tr>
<tr>
<td>CRC32</td>
<td>eaf06200</td>
</tr>
<tr>
<td>MD4</td>
<td>244ac66a6bf2cfc6f851f2c0d74a958c</td>
</tr>
<tr>
<td>MD5</td>
<td>4545d603b1d1eb9e7611d8897343ec29</td>
</tr>
<tr>
<td>SHA-1</td>
<td>85695bcbc0c4ccaf864347018d1a93021c6d2b5b</td>
</tr>
<tr>
<td>SHA-256</td>
<td>35e82dd5e8910e44e8c3274b8375cf17da2e8e89b76990125b20c b04472b6a3b7</td>
</tr>
<tr>
<td>SHA-384</td>
<td>3c93c726709596626d7dec8693368a93c7ac33653adf6a825422 98b10ff4d9a60358a3a5d2c88e5062116e6468e86da</td>
</tr>
<tr>
<td>SHA-512</td>
<td>856ab8ea61c622d37faa117c8fcd9b6933190be9669fd76653e2b 31b0ff99516d60a18efbf065039d0626ff5c2f35d93cb07e4bfda22 d7fb50edcde07bba1ed</td>
</tr>
</tbody>
</table>
A7. MISSING FUNCTION LEVEL ACCESS CONTROL
A7 – Missing Function Level Access Control

- Exploitability: EASY
- Prevalence: COMMON
- Detectability: AVERAGE
- Impact: MODERATE

- Occurs when page or function level access controls are missing
- This allows attackers to gain unauthorized access to unprotected functionality
A7. Missing Function Level Access Control

• **Missing Function Level Access Control:**
  Attacks can happen when the application does not perform server-side entitlement checks.

• **Caused By:**
  – Lack of access controls at the page level
  – Lack of access controls at the function level
  – Client-side only access controls
Lack of Access Control at The Page Level

Unprotected Functionalities
https://vulnerable.com/admin

Unprotected Pages

Unprotected Static Files
Client-side Access Controls

Consider the following example:

```html
1 <html>
2  <body>
3   <!-- User's Menu-->
4   <div name="customerDiv">
5      <a href="/viewaccount.php">View Your Account</a>
6      <a href="/paybill.php">Pay Your Bill</a>
7      <a href="/logout.php">Logout</a>
8   </div>
9   <?php if (user.isAdmin) {
10      $admin = "visible";
11     } else {
12        $admin = "hidden";
13     }
14   }
15 </body>
16 </html>
```

Attackers will find and attack these URLs!
A8. CROSS-SITE REQUEST FORGERY
A8 – Cross-Site Request Forgery

- Exploitability: AVERAGE
- Prevalence: WIDESPREAD
- Detectability: EASY
- Impact: MODERATE

- CSRF takes advantage of web applications that allow attackers to predict all the details of a particular action.
- Since browsers send credentials like session cookies automatically, attackers can create malicious web pages which generate forged requests that are indistinguishable from legitimate ones.
Cross-site Request Forgery

1. GET http://mybank.com/login/
2. HTTP/1.1 200 OK
   Set-Cookie: jsessionid=123456789
4. <html>
   <body>
   ...snip...
   <img src=http://mybank.com?transferTO=110998&amount=1000/>
   ...snip...
5. GET http://mybank.com?transferTO=110998&amount=1000
6. HTTP/1.1 200 OK
   Set-Cookie: jsessionid=123456789
Cross-Site Request Forgery

1. User visits their online banking site www.mybank.com
2. The user is logged in.
3. While being logged in, the user visits a malicious website.
4. The malicious website contains the following code.
   
   ```html
   <html><body>
   ...snip...
   <img src=http://mybank.com?transferTO=110998&amount=1000" />
   ...snip...
   ```

5. The website sends the HTML to the user’s browser.
6. The HTML is rendered and a request for the fund transfer goes to mybank.com
A9. USING COMPONENTS WITH KNOWN VULNERABILITIES
A9 – Using Components with Known Vulnerabilities

- Exploitability: AVERAGE
- Prevalence: WIDESPREAD
- Detectability: DIFFICULT
- Impact: MODERATE

- Occurs when the application uses components with unpatched vulnerabilities (allowing zero-day attacks)
- Attackers can access protected views by simply entering the target URL to access it directly.
Heartbleed

- A bug that allowed attackers to read the memory of the vulnerable server.
- 800,000 Servers were vulnerable.
- 30,000 X.509 Certificates had to be replaced
- CRA lost 900 taxpayers’ SINs
A10. UNVALIDATED REDIRECTS AND FORWARDS
A10 – Unvalidated Redirects and Forwards

- Exploitability: AVERAGE
- Prevalence: UNCOMMON
- Detectability: EASY
- Impact: MODERATE

- Applications frequently redirect users to other pages, or use internal forwards in a similar manner.
- Sometimes the target page is specified in an unvalidated parameter, allowing attackers to choose the destination page.
Unvalidated Redirects and Forwards - Example

• Consider the following example, what is the problem?

```csharp
string returnUrl = Request.Form["returnUrl"];
if (!String.IsNullOrEmpty(returnUrl)) {
    return Redirect(returnUrl);
} else {
    return RedirectToAction("Index", "Home");
}
```
Unvalidated Redirects and Forwards - Example


GET /welcomeredirect=www.attacker.com HTTP/1.1
Host: vulnerableapp.com
Exploiting Unvalidated Redirects and Forwards

From: eBay-Optima <checkout@eBay.com>
Subject: eBay Member: teccalep**Read Now**
Date: November 30, 2005 5:31:00 PM PST
To: Vaughn Aubuchon
Reply-To: checkout@eBay.com

Update Your Information

Dear eBay user,

During our regular update and verification of the accounts, we couldn't verify your current information.

Either your information has changed or it is incomplete.

Please click here update and verify your information by signing in your account below.

If the account information is not updated to current information within 5 days then, your access to EBay will be terminated.

This eBay notice was sent to brian.s.clifton@st.yahoo.com based on your eBay account preferences. Notification preferences for other types of communications, click here. If you would like to receive

As outlined in our User Agreement, eBay will periodically send you information about site changes. Policy and User Agreement if you have any questions.

Copyright 2003 eBay Inc. All Rights Reserved. Designated trademarks and brands are the property of their respective holders.