Cross-Site Scripting

Getting Developers to Take XSS Seriously

Use Social Engineering to Enhance Your Vulnerability Reporting
Contact Information

Jason Pubal

Website
www.intellavis.com/blog

E-mail
jpubal@gmail.com

Social
http://www.linkedin.com/in/pubal
http://www.twitter.com/pubal
Cross-Site Scripting

Outline

- What is XSS?
- XSS History
- Detecting XSS
- Preventing XSS
- Reporting Tricks
Cross-Site Scripting

Cross Site Scripting (XSS) is an attack against the user of a website. It is a technique that forces a website to display malicious code, which then executes in the user’s web browser. The attacker uses a vulnerable website to send malicious code to another end user of the site. The vulnerability arises when the website takes data in some way from a user and dynamically includes it in a web page without first validating that data.

• account hijacking
• rewrite portions of the page
• log keystrokes
• steal browser information
• Steal client machine data
• attack the user’s network
Persistent Cross-Site Scripting
Reflected Cross-Site Scripting
Websites with Cross-Site Scripting

WhiteHat Website Security Statistic Report, Winter 2011
Attacks Using Cross-Site Scripting

Web Hacking Incident Database

Top Attack Methods (All Entries)
Real World Examples

**Hacker Redirects Barack Obama’s site to hillaryclinton.com**
During the 2008 democratic primaries, XSS in Obama’s website was exploited to redirect visitors to Hillary Clinton’s website. Users who went to Obama’s community blog were instead taken to www.hillaryclinton.com.

**Apache.org hit by targeted XSS attack, passwords compromised**
A targeted attack against JIRA admins used XSS to steal administrative cookies. Using those privileges, they installed backdoors and scripts to collect passwords at login. Thanks to people’s tendency to use the same password on several websites and applications, the attacker was able to use those credentials get root access to other servers.

**New XSS Facebook Worm Allows Automatic Wall Posts**
An XSS in the Facebook’s mobile API allowed a maliciously prepared iframe element containing JavaScript to post to user’s walls.
History of Cross-Site Scripting
“I’m sorry MySpace and FOX. I love you guys, all the great things MySpace provides, and all the great shows FOX has, my favorite being Nip/Tuck. Oh wait, Nip/Tuck is FX? My bad, but FOX, I’m sure you still have some good stuff. But maybe you should start picking up Nip/Tuck reruns? Just a thought. I’m kidding! Please don’t sue me.”
Samy

Fastest Spreading Worm in History
JavaScript Malware

Cross Site Scripting (XSS) is an attack against the user of a website. It is a technique that forces a website to display malicious code, which then executes in the user’s web browser. The attacker uses a vulnerable website to send malicious code to another end user of the site. The vulnerability arises when the website takes data in some way from a user and dynamically includes it in a web page without first validating that data.

- account hijacking
- rewrite portions of the page
- log keystrokes
- steal browser information
- steal client machine data
- attack the user’s network

- ANYTHING A USER CAN DO OR ACCESS FROM THE BROWSER!
Manual Testing

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

XSS Cheat-Sheet: [http://ha.ckers.org/xss.html](http://ha.ckers.org/xss.html)

Browser Plugins
Web Application Vulnerability Scanners
Web Application Vulnerability Scanners
Preventing Cross-Site Scripting

https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

**Input Validation**

Accept known good (whitelist)
Reject known bad (blacklist)
Sanitize (change input to acceptable format)

**Output Encoding / Escaping**

Characters will still render in a browser correctly; escaping simply lets the interpreter know the data is not meant to be executed.

& → &amp;
< → &lt;
> → &gt;
" → &quot;
' → &apos;
/ → &#x2F;
Preventing Cross-Site Scripting

Use Libraries

ESAPI – https://www.owasp.org/index.php/ESAPI
MS Anti-XSS Library - http://wpl.codeplex.com
The value of the search_txt request parameter is copied into the value of an HTML tag attribute which is not encapsulated in any quotation marks. The payload `<script>alert(XSS)</script>` was submitted in the search_txt parameter. This input was echoed unmodified in the application's response.

This proof-of-concept attack demonstrates that it is possible to inject arbitrary JavaScript into the application's response.
Browser Exploitation Framework (BeEF)

http://beefproject.com/
Cross-Site Scripting Exploitation
Socially Engineering your Report

Exploit the Vulnerability! Report the Impact!
Socially Engineering your Report

Exploit the Vulnerability! Report the Impact!
Socially Engineering your Report

Exploit the Vulnerability! Report the Impact!
Copy Machine Experiment

The Power of “Because”

“May I use the Xerox machine?”
Giving no reason - 60%

“May I use the Xerox machine, because I have to make copies?”
Giving no real reason - 93%

“May I use the Xerox machine, because I’m in a rush?”
Giving a reason - 94%
Commander’s Intent

Give Them a Reason!

Everybody's got a plan until I hit them.

— Mike Tyson
Bystander Apathy

Assign a JIRA Ticket!
The Ponemon Institute puts the cost per record of a breach at $214, with an average cost of 7.2 million dollars. By contrast, a week of development time seems cheap.

Options
1 - $5,000  95% Effective
2 - $500  80% Effective
Herd Effect

You’re all sheep.

Best Practices

Amazon and Facebook employ CAPTCHA

93% of Websites in our Industry use Input Validation
Pygmalion Effect

Clearly Communicate Expectations
Metrics

If you want to improve something, measure it.

Measure to see if what you're doing is working. If not, try something else.
THANK YOU

Questions?!