

Course 2: Programming Issues, Section 4

Pascal Meunier, Ph.D., M.Sc., CISSP

Updated February 28, 2005

Developed thanks to the support of Symantec Corporation,
NSF SFS Capacity Building Program (Award Number 0113725)
and the Purdue e-Enterprise Center

Copyright (2004) Purdue Research Foundation. All rights reserved.



Course 2 Learning Plan

- Buffer Overflows
- Format String Vulnerabilities
- Code Injection and Input Validation
- **Cross-site Scripting Vulnerabilities**
- Links and Race Conditions
- Temporary Files and Randomness
- Canonicalization and Directory Traversal

Learning objectives

- Understand the definition of a cross-site scripting vulnerability
- Know how they happen and why they are so hard to prevent
- Learn some ways to prevent them

Cross-Site Scripting: Outline

- Survey of client-side scripting technologies
- Definition
- Risks
- Security zones
- Examples
- Types of XSS
 - Without storage
 - With storage
- Other JavaScript vectors
- Lab: Explore the ubiquity of JavaScript
- Discussion

Client-side Scripting

- JavaScript Family
 - ECMAScript (ECMA-262 standard)
 - ❖ based on JavaScript 1.1
 - ❖ Third edition is now current
 - JavaScript (now at V. 1.5, compatible with ECMA 3rd Ed.)
 - JScript is Microsoft's implementation
- ActiveX Family
 - VBScript
 - ❖ Requires Internet Explorer on Windows
 - ActiveX controls
- Java Family
- ActionScript (Flash)

Cross-Site Scripting Vulnerabilities

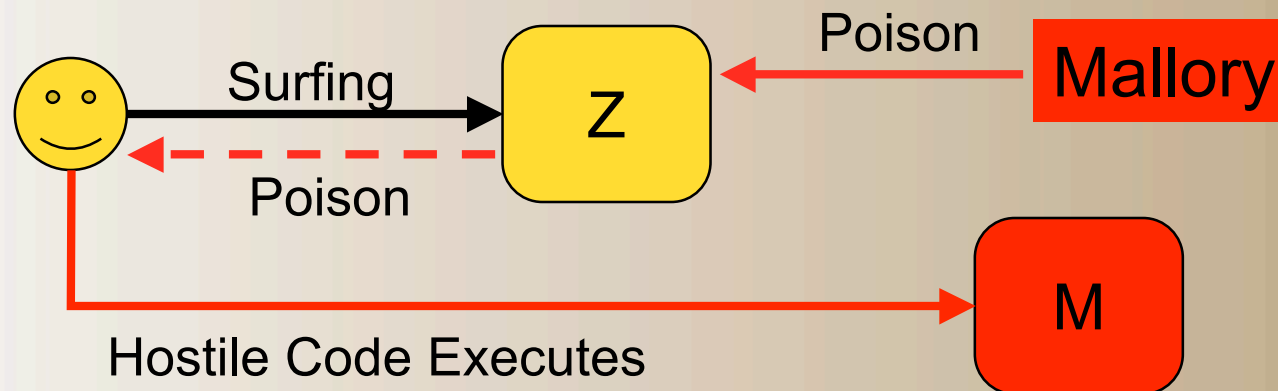
- A cross-site scripting vulnerability allows the introduction of malicious content (scripts) on a web site, that is then served to users (clients)
 - Malicious scripts get executed on clients that trust the web site
 - Problem with potentially *all* client-side scripting languages
- Use “XSS” to refer to these vulnerabilities, to avoid confusion with “CSS” (cascading style sheets)

XSS Concept

- Any way to fool a legitimate web site to send malicious code to a user's browser
- Almost always involves user content (third party)
 - Error messages
 - User comments
 - Links
- References
 - http://www.cert.org/archive/pdf/cross_site_scripting.pdf
(Jason Rafail, Nov. 2001)
 - <http://www.spidynamics.com/support/whitepapers/SPIcross-sitescripting.pdf>

Why the Name

- You think that you interact with site Z
- Site Z has been poisoned by attacker (Malory)
- The "poison" (e.g., JavaScript) is sent to you, along with legitimate content, and executes. It can exploit browser vulnerabilities, or contact site M and steal your cookies, usernames and passwords...



XSS Risks

- Theft of account credentials and services
- User tracking (stalking) and statistics
- Misinformation from a trusted site
- Denial of service
- Exploitation of web browser
 - Create phony user interface
 - Exploit a bug in the browser
 - Exploit a bug in a browser extension such as Flash or Java
- Etc.

XSS Risks -- Stolen Account Credentials

- With XSS, it may be possible for your credentials to be stolen and used by attacker
- Web sites requiring authentication need to use a technological solution to prevent continuously asking users for passwords.
 - Credentials have the form of a SessionID or nonce
 - ❖ Url encoding (GET method)
 - <http://www.site.com?ID=345390027644>
 - ❖ Cookies are commonly used to store credentials
 - These are usually accesible to client-side scripts

Cookie Mechanism and Vulnerabilities

- Used to store state on the client browser
- Access Control
 - Includes specification of which servers can access the cookie (a basic access control)
 - ❖ Including a path on the server
 - So cookie can be used to store secrets (sessionIDs or nonces)
- Side Note: Vulnerabilities in implementations
 - Cross-Domain Cookie Injection Vulnerability in IE 6.0.0, Firefox 0.9.2, Konqueror
 - ❖ <http://securityfocus.com/bid/11186>
 - ❖ CAN-2004-0746, CAN-2004-0866, CAN-2004-0867

XSS -- Point

- XSS vulnerabilities fool the access control mechanism for cookies
- The request for the cookie (by scripts) comes from the poisoned server, and so is honored by the client browser
 - No vulnerabilities needed in the client browser

XSS Risk -- Privacy and Misinformation

- Scripts can "spy" on what you do
 - Access history of sites visited
 - Track content you post to a web site
- Scripts can misinform
 - Modify the web page you are viewing
 - Modify content that you post
- Privacy ("I have nothing to hide")
 - Knowledge about you can be valuable and be used against you
 - ❖ Divorces, religion, hobbies, opinions
 - ❖ etc...

XSS Page Modification Example

- Cross-frame vulnerabilities, a.k.a. "Frame Injection"
 - A web page can modify a frame presented in another window
 - ❖ CAN-2004-0717 to -0721
 - Demo:
 - ❖ http://secunia.com/multiple_browsers_frame_injection_vulnerability_test/
- Impact: A malicious script running from one frame (e.g., from a previously visited site with XSS vulnerabilities) can modify subsequently visited sites in the other frame

XSS Risk -- Denial of Service

- Nasty JavaScripts can make your web site inaccessible
 - Make browsers crash or become inoperable
 - Redirect browsers to other web sites
- See: "Nasty, Malicious and Dangerous JavaScripts"
 - <http://www.vipro.de/javascripts> (in German)
 - ❖ Use <http://translate.google.com> to get it in English
 - ❖ Scroll to the bottom
 - Several scripts implement DoS attack on browser
 - ❖ Need to force-quit or kill browser!

XSS Risk -- Silent Install

- Exploitation of browser vulnerabilities
 - JavaScript, ActiveX, etc... allow the exploitation of browser vulnerabilities
 - ❖ Run locally on your machine
 - ❖ User security confirmation bypass vulnerability in Microsoft Internet Explorer 6.0 SP2:
 - <http://securityfocus.com/bid/11200>
 - Allows malicious users to trivially bypass the requirement for user confirmation to load JavaScript or ActiveX.
 - Installation of malicious code
 - Installation of user interfaces
 - ❖ Mozilla/FireFox XUL Interface spoofing vulnerability
 - CAN-2004-0764
 - Secunia Advisory SA12188
 - <http://securityfocus.com/bid/10832>

XSS Risk -- Phishing

- User Interface Modifications
 - Present fake authentication dialogs, capture information, then perhaps redirect user to real web site
 - Replace location toolbar to make user think they are visiting a certain web site
- Phishing Scenario
 - ❖ Victim logs into a web site
 - ❖ Attacker has spread "mines" using an XSS vulnerability
 - ❖ Victim stumbles upon an XSS mine
 - ❖ Victim gets a message saying that their session has expired, and they need to authenticate again
 - ❖ Victim's username and password are sent to attacker

Security Zones Model

- Internet Explorer
 - Local, Trusted, Internet, Restricted



- Scenario:
 - Trusted sites are allowed to run scripts
 - One of the trusted sites has a XSS vulnerability
 - A malicious script is planted on it
 - The script is trusted and run, and so can steal usernames, passwords, session cookies, etc...
 - ❖ stolen values can be sent as part of a contacted url (GET: url?v=value)

Accountability

- Accountability normally restrains the maliciousness of scripts on web sites.
- This is broken by XSS vulnerabilities; there is no limit to the maliciousness of a script.
 - Authors are not accountable because they are unidentified

History of Malicious Scripts

- 2000: Microsoft forced to shut down Hotmail
 - Script intercepted Hotmail authentication cookies and took over users' accounts
 - ❖ Javascript forwarded cookies to another site
- 2000: Zkey.com JavaScript exploit
 - XSS vulnerability allowed hacker to capture usernames and passwords
 - ❖ Social engineering aspects (phishing); Javascript mimicked the Zkey.com login dialog box ("please re-login")
 - ❖ See Rothermel, D. (2000)

Other Malicious Scripts

- 2001: Japanese auction web site "Price Loto" disseminated a malicious script that "altered the configuration of users' PCs" (users even had trouble shutting down the computer). The web site closed temporarily.
 - Miyake K., IDG News Service
- 2002: VBScript changes favorites and home page
 - JS.IEStart, a.k.a. FunChina, VBS.Passon (CA), VBS.PassOn (NAV) VBS/IEstart.gen.
 - Alters registry
key:HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main\Start Page

VBscripts that change Registry Keys

- 10/2003: QHosts-1 Exploits an Internet Explorer vulnerability
- Creates a new registry key, and modifies 6 others
- Distributed by getting people to visit an infected web site
- Performs man-in-the-middle attack on DNS
- Many more examples of scripts changing registry keys

XSS Vulnerability: Reflection

- A vulnerable web site is one that "reflects" or echoes data back to a user
 - No storage needed on the vulnerable web site itself

```
<?php
echo $input
?>
```
- The attacker creates an html link with some script in it as input to vulnerable web site. This may be in an email, or Malory's own web site.
 - `<A HREF='http://vulnerable.com?input=<malicious code'>Click here for free stuff!`
- What happens when Alice clicks on the link?

Results

- Alice clicks on link
- Alice is taken to the correct site
- Malory's code is echoed by the vulnerable site and executed by Alice's browser ***in the context of the vulnerable site***
 - sends Alice's cookies, visited urls, etc. to Malory's computer
- Variations: error or status messages that quote the malicious code
- Example: VBulletin forum
 - CAN-2004-0091
 - <http://www.securityfocus.com/archive/1/353673>

XSS Vulnerability: Stored

- Malory enters comments or text that contains an embedded script, in a forum, newsgroup, feedback section of a web site, etc...
- The malicious code is stored by the vulnerable site, and presented to visitors. Each instance can be thought of as a "mine".
- Alice reads the comments. Malory's code is executed on Alice's computer...
- Example: CAN-2003-1031
 - XSS vulnerability in register.php for vBulletin 3.0 Beta 2 allows remote attackers to inject arbitrary HTML or web script via optional fields such as (1) "Interests-Hobbies", (2) "Biography", or (3) "Occupation."

JavaScript urls

- JavaScript urls have the format "javascript:code"
 - An example JavaScript url is
 - ❖ `javascript:alert("Hello World")`
 - Type it in your browser's address bar, watch the alert window popup
 - Works also in <A> HTML links
 - ❖ `"javascript:alert(document.cookie)"`
 - ❖ JavaScript urls could be injected into the history list and then executed in the local machine zone or some other zone
 - CAN-2003-1026
 - CAN-2003-0816 (several injection methods)
 - ❖ JavaScript url in a frame (Opera <= 6.01; CAN-2002-0783) was executed in the context of other sites

Indirect Ways to Inject Code

- ActionScript (Flash) can load a JavaScript script from a url
 - Flash objects can be specified with the <embed> tag
 - ❖ ActionScript allows the getURL("url") function call
 - ❖ The url can be a JavaScript url!
- Forums that allow Flash content are vulnerable
 - People viewing the Flash content get a trojan JavaScript
- See <http://www.cgisecurity.com/lib/flash-xss.htm>

Spreading the Fame

- RSS (Real Simple Syndication) is a data feed from some web sites, to be displayed on someone else's web site
- Example: <http://slashdot.org/index.rss>
- What if the data feed contains malicious code?

Lab

- Goal: Get an insight on how hard it is to block JavaScript while allowing as much HTML functionality as possible
- You'll just need a web browser and a text editor

Lab Step 1

- Create a cookie in your web browser
- Load <http://www.cerias.purdue.edu/secprog/class2/XSS/step1.html>
 - ❖ `<title>Set the cookie!</title>`
`<script>`
`var curCookie = "XSS_test_cookie =" + "This`
`is my test cookie" + "; path=/";`
`document.cookie = curCookie;`
`</script>`
- The cookie should now have been set; look at your cookies

Lab Step 2

- Try to execute JavaScript in a title tag
 - Load <http://www.cerias.purdue.edu/secprog/class2/XSS/step2.html>
 - ```
<html><head>
<title>XSS tests <script>alert('cookie:'
 +document.cookie)</script>
</title></head>
<body>
Test for XSS in title.
</body></html>
```
- What happens?
- Try a different browser

## Comments on Step 2

- Doesn't work in Internet Explorer, Safari
- Worked in Mozilla (<1.3) and others
- So, don't assume that browser behavior is homogeneous and that it's safe to ignore something because one browser does.

## Lab Step 3: The <script> Tag

- Is the <script> tag necessary to execute JavaScript?
  - Load <http://www.cerias.purdue.edu/secprog/class22/XSS/step3.html>
    - ❖ 

```
<html><head>
 <title>XSS test Step 3</title>
</head>
<body>
Test for script embedded in html link

<a href="X"
onmouseover="alert ('cookie='+document.cookie
) ">X
```
- What happens when you move the cursor over the link?

## Comments

- You didn't need to click on anything to get the script executed
- There was no `<script>` tag
- What other events are there?
  - mousedown, mouseup
  - click
  - dblclick
  - mousemove
  - mouseover, mouseout
  - mouseenter, mouseleave
- and many more, see <http://www.gatescript.com/events.html>

## Where Else?

- HotMail vulnerabilities:
  - JavaScript in <header>
  - JavaScript in <style> tags
- Which other tags?

## Example: PHPNuke 6.0

- October 2002 advisory with 7 XSS vulnerabilities
- Popular web portal software
- News module
- PayPal module
- Open source (free)

# PHPNuke XSS

- PHPNuke aggregates the RSS feeds from many different sites.
- XSS #1: PHPNuke accepts html tags and presents them to users
- ```
<item rdf: about="http://www.somesite.dom">  
<title>  
<script>  
alert(`cookie:`+document.cookie)  
</script>  
</title>  
</item>
```

 - (adapted from genhex.org advisory)

PHPNuke XSS #2

- Strips `<script>` tags
- Forgot events on `<a href>` tags
- `X`
- Works in IE, etc... (Demo)
 - (Adapted from genhex.org advisory)

PHPNuke Other XSS

- XSS #3: User Info
- Tags in name and email fields not stripped at all
- Name and email listed in list of users scripts executed automatically!
- and so on for #4-7
 - (Adapted from genhex.org advisory)

Question

Why is it hard to perform input validation on submitted html data so as to remove all JavaScript? Choose two correct answers.

- a) JavaScript in certain locations may be executed by some browsers and not others, in non-standard and unexpected ways
- b) JavaScript looks like html, and most web pages require JavaScript
- c) JavaScript can be specified and embedded in various ways inside html tags

Discussion

How can you prevent cross-site scripting vulnerabilities? How about:

- a) Disabling scripting (which?) on client browsers
- b) As web masters, not require JavaScript
- c) Transform all inputs with equivalent (harmless) html encodings
 - "<" becomes <
 - etc...
- d) Build a model of valid HTML without scripting, then filter out what doesn't match
- e) Using a filtering proxy

So You Disabled Scripting...

- What if the browser doesn't respect your wishes?
- Scripts can be embedded inside xml stylesheets
- Executed regardless of settings for Active Scripting
- IE/Outlook Express, etc...
- Fixed now, but may reoccur
 - Guninski April 2001
- What if the browser, or a plugin, has a vulnerability that allows re-enabling scripting?

Questions or Comments?

About These Slides

- You are free to copy, distribute, display, and perform the work; and to make derivative works, under the following conditions.
 - You must give the original author and other contributors credit
 - The work will be used for personal or non-commercial educational uses only, and not for commercial activities and purposes
 - For any reuse or distribution, you must make clear to others the terms of use for this work
 - Derivative works must retain and be subject to the same conditions, and contain a note identifying the new contributor(s) and date of modification
 - For other uses please contact the Purdue Office of Technology Commercialization.
- Developed thanks to the support of Symantec Corporation



Pascal Meunier **pmeunier@purdue.edu**

Contributors (in no particular order):

Jennifer Richardson, Jared Robinson, Alan Krassowski,
Craig Ozancin, Tim Brown, Wes Higaki, Melissa Dark, Chris
Clifton, Gustavo Rodriguez-Rivera

