Secure web interface

- **Main gap:**

  Human generated versus sensors/database generated data
- **Standalone machine:**
  - **passwd protected**
  - **restricted permissions, ACL, etc...**
- **Web applications:**
  - **Public part (ex: index pages, register login, etc..)**
  - **Private part**
- **Safe/unsafe programming languages**
  - **Not really!**
- **Safe/unsafe programming styles ?**
  - **Yes**

Short list of threats

- XSS or Cross-Site Scripting
- Cross Site Request Forgeries (CSRF)
- SQL injection
- Information leakage
- Broken session management
- Good source of information

  Open Web Appli. Security Project (oswap.org)

XSS

- Malicious data sent to a client through the server
- Client forced to execute malicious scripts
- Example: forum with user comments
  - client side: HTML form with
    - Input type text with name=user
    - Input type textarea with name=comment
  - server side: script
    - Save the comment (file or database)
    - genenerate the comment on demand
    - Similar to `echo $comment`

XSS example

- **HTML code:**

```
<form action=writecomment.php method=POST>
Enter your name: <input type=text name=username><br/>
Enter your comment: <textarea name=comment></textarea><br/>
<input type=submit value=« add comment »>
</form>
```

- **PHP code: `writecomment.php`**

```
<?php >
$file=fopen(...,'a') ;
$comment= $_POST['comment'] ;
fwrite ($file, $comment)
?>
```

- **PHP code: displaycomment.php**

```
<?php > //get the comment from the DB or file
$file=fopen(...,'r') ; $line=fgets($file) ; echo $line....
?>
```

- **See live example;-)**

XSS solution???

- **Safe code**
  - **Use `htmlentities` to escape strings sent to the client**
  - **HttpOnly (see next slide)**
- **Safe server config (if you can)**
  - **`magic_quote` in php.ini** (to be avoided)
- **Safe client : forbid Javascript !**
- **Twitter webside ->Sept 2010**
- **Every day, new pb;-)**

HttpOnly flag

- Coming from Microsoft (2002)
- Target : cookie stolen via XSS
- Idea : add a Boolean field to a cookie
- HttpOnly = true/false
- Semantics :
  - JS readable/not readable
  - document.cookie does not work (empty string or encrypted)
- Default value : false;-)

HttpOnly howto

- with PHP : function setcookie

  setcookie($name,$value,time()+3600,$domain,true)
- Other option : PHP config file php.ini

  session.cookie_httponly = True
- PB1 : browser support ?
  - See www.browserscope.org
  - Otherwise addon to the browser
- PB2 : does not forbid XMLHttpRequest
  - XMLHttpRequest.getResponseHeader
  - XMLHttpRequest.getAllResponseHeaders
  - Fixed with last Firefox (I think;-)

SQL injection: idea

- Malicious data sent to the server by the client
- Server forced to execute malicious scripts
- Aug. 2007, UN web site was defaced using SQL injection
  hackademix.net/2007/08/12/united-nations-vs-sql-injections/
- Much more dangerous than XSS;-)
- Good tutorial:

  http://www.unixwiz.net/techtips/sql-injection.html

SQL injection: example

- **A standard pb: « SQL injection » (ex. with PHP)**

$q=‘SELECT * FROM users WHERE username=“’ . $_GET['username'] . ‘””
Username coming from a public HTML form
$_GET[‘username’] = ‘ richard”;DELETE * FROM users WHERE username!=“
$result = mysql_query($q) (or ‘ r”; DROP TABLE users‘ …) …DISASTER ;-)

Technique: check every combination till success!

Examples

- **Aug. 2007**, UN web site was defaced using SQL injection (hackademix.net/2007/08/12/united-nations-vs-sql-injections/)
- **August 17, 2009**, the United States Justice Department charged an American citizen Albert Gonzalez and two unnamed ...
- **December 2009**, an attacker breached a Rock You! plaintext ...

SQL injection: solution?

- **Programming side**
  - Secure programming
  - Parameter checking (string analysis, forbidden char)
  - `addslashes` function
- **Server admin side**
  - Restricted database privileges to limit damages
  - Magic quoting (magic_quotes_gpc = On within php.ini)
  - (With wamp php.ini in Apache2/bin folder)
  - Prevent multi-statements query

LAMP/WAMP

- **Linux/Windows, Apache, MySQL, PHP**
- **2006: 43% of frauds with PHP (NIST)**
- **3 important files:**
  - httpd.conf for Apache
  - my.ini for MySQL
  - php.ini for PHP
- **+ the config file for Linux servers;-)**
- **Show examples**

Conclusion

- Security for web interface...hot topic
- Main holes: human errors!
- WIFI network... worse !
- Convergence (GPRS/GSM, etc...)
- Internet TV

The never ending story!