# Week 3 - lolphp

Introduction to Offensive Security

# PHP?

- One of the (if not the) most used languages for back-end webdev
- *Tries* to be very… robust
  - Fails "gracefully" in a lot of situations

# PHP Overview

- C-like language
- Enclosed by `<?php … ?>` (sometimes just `<? … ?>`)
  - Inlined into HTML
- Variables start with $
  - `$name`
  - Variable variables… `$$name`
- Request-specific dictionaries: `$_GET`, `$_POST`, `$_SERVER`

# PHP Overview

- Keep the php docs open. Function names are strange
  - Length of function name used to be key in internal dictionary, so function names were shortened/lengthened to make the lookup faster

# PHP Example

```
<?php
    if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['email']) && isset($_POST['password'])) {
        $db = new mysqli('127.0.0.1', 'cs3284', 'cs3284', 'logmein');
        $email = $_POST['email'];
        $password = sha1($_POST['password']);
        $res = $db->query("SELECT * FROM users WHERE email = '$email' AND password = '$password'");
        if ($row = $res->fetch_assoc()) {
            $_SESSION['id'] = $row['id'];
            header('Location: index.php');
            die();
        }
    }
?>
<html>…
```

Automatic string interpolation

# Type Juggling

- PHP will do just about anything to match with a loose comparison (==)
  - Things can be equal (==) or **really** equal (===)
- Implicit int parsing strings is the root cause of a lot of issues

# Examples

- 0 == "0"
- 0 == null
- "0e1234" == "0e4321"
  - Big issue when comparing hashes…
  - "1e1234" != "1e4321" though
- 0 == "asdf"
- $arr = array(); $arr[doesntexist] == null
- $arr = array(); md5($arr) === md5("Array")

# Demo

# File Inclusion

- PHP has multiple ways to include other source files
  - `require`
  - `require_once`
  - `include`
- These can take a dynamic string
  - `require $_GET['page'] . ".php";`
  - Usually seen in templating

# Demo

# PHP Stream Filters

- PHP has its own URL scheme: php://…
- Main purpose is to filter output automatically
  - Automatically remove certain HTML tags
  - Base64 encode

# PHP Stream Filters

```
$fp = fopen('php://output', 'w');
stream_filter_append(
       $fp,
       'string.strip_tags',
       STREAM_FILTER_WRITE,
       array('b','i','u'));

fwrite($fp, "<b>bolded text</b> enlarged to a <h1>level 1 heading</h1>\n");

/* <b>bolded text</b> enlarged to a level 1 heading   */
```

# PHP Stream Filters

- These filters can also be used on input
  - `php://filter/convert.base64-encode/resource={file}`
- `include, file_get_contents(),` etc. support URLs
  - … including PHP stream filter URLs
- `include` normally evaluates any PHP code (in tags) it finds
  - But if its base64 encoded…
  - Useful way to leak source

# Demo