Hack-proof Your Drupal App

Key Habits of Secure Drupal Coding
Introductions

• CommonPlaces
  – Erich Beyrent, V.P. of Engineering
  – Amanda Giovanni, Director of Enterprise Risk Management
Introductions

• WhiteHat Security
  – Arian Evans, Operations Director
Introductions

• Katalyst Strategies
  – Matthew J. Nash, Director of Cyber Risk Management
Security? Who cares?
Have you ever been hacked?

1337 h3xx0r

Drupal d00d

pwnage!
Web Application Security

• Analysts estimate that 75% of attacks against web servers enter at the application, not the network level
• As many as 15% of these attacks are due to poor coding practices
Case Study: Greenopolis.com
Case Study: Greenopolis.com

- Modules
- Theme
- Flash and Flex
- Security Audit Results
Greenopolis Modules

- Greenopolis currently uses 92 modules
Greenopolis Modules

- Greenopolis currently uses 92 modules
- 20% of those modules are custom
Greenopolis Theme

• Greenopolis uses a custom theme
• template.php, lots of individual templates
Flash and Flex

• Greenopolis has Flex widgets that can be embedded on any site, which get data via Services
Flash and Flex

• Greenopolis has Flex widgets that can be embedded on any site, which get data via Services
• Greenopolis has an interactive Flash map of the US that gets data from Services
Flash and Flex

• Greenopolis has Flex widgets that can be embedded on any site, which get data via Services

• Greenopolis has an interactive Flash map of the US that gets data from Services

• Greenopolis has a Flex TiltingPane widget that displays partner logos, populated via Services
Security Audit Results

• WhiteHat Security performed an audit of the application layer
120 vulnerabilities were discovered in the following categories:
- XSS
- CSRF
- SQL Injection
- Session fixation
- Insufficient Authorization
What We Learned:

- 90% of the vulnerabilities existed in the theme
Fixing The Problems:

- Completely reviewed the theme, implementing Drupal output filters
Fixing The Problems:

• Completely reviewed the theme, implementing Drupal output filters
• Code was audited to ensure sanitization of all user generated data
Fixing The Problems:

• Completely reviewed the theme, implementing Drupal output filters
• Code was audited to ensure sanitization of all user data
• Rewrote the search forms to sanitize user generated data
Fixing The Problems:

• Completely reviewed the theme, implementing Drupal output filters
• Code was audited to ensure sanitization of all user data
• Rewrote the search forms to sanitize user data
• Implemented web services proxy
Key Habits of Secure Drupal Coding
We are going to discuss:

• Sanitize Your Output
We are going to discuss:

• Sanitize Your Output
• Protect Your Database
We are going to discuss:

• Sanitize Your Output
• Protect Your Database
• User Input
We are going to discuss:

• Sanitize Your Output
• Protect Your Database
• User Input
• AJAX Risks
Sanitize Your Output:

- check_plain()
- check_markup()
- filter_xss()
- filter_xss_admin()
Sanitize Your Output:

Correct Usage:

```php
echo check_plain($node->field_fname[0]['value']);
echo content_format('field_fname', $node->field_fname[0]['value']);
echo t('%fname', array('%fname' => $node->field_fname[0]['value']));
echo t('@fname', array('@fname' => $node->field_fname[0]['value']));
```

Incorrect Usage:

```php
echo $node->field_fname[0]['value'];
```
Protect Your Database:

- `db_query()`
- `db_rewrite_sql()`
Protect Your Database:

Correct usage:
$result = db_query("SELECT foo FROM {bar} WHERE foo = '%s', $unsafe_var");

Incorrect usage:
$result = db_query("SELECT foo FROM {bar} WHERE foo = $unsafe_var");
User Input:

• Use the Form API
User Input:

- Use the Form API
  - Sanitize $_POST (and $_GET) data
Javascript and AJAX:

• Rule #1 – Javascript can be disabled
• Don’t assume data to AJAX postback functions are coming from your javascript
• Don’t assume AJAX transactions are private and cannot be observed by users
• Re-encode and escape all output
Don’t Trust User Input!
Things That Will Bite You:

- Outputting raw values
- Modifying data with $_GET
- Parameterized queries? WTF?
- Hacking core and killing kittens
Drupal Security Resources

• Drupal.org
  – http://drupal.org/writing-secure-code
  – http://drupal.org/node/28984

• Drupal Security Team
  – http://drupal.org/node/32750

• Pro Drupal Development book
  – http://drupalbook.com
Drupal Tools

- Update module [http://drupal.org/project/update_status](http://drupal.org/project/update_status)
- Coder module [http://drupal.org/project/coder](http://drupal.org/project/coder)
- Interactive debuggers (Zend, XDebug)
Participate!

• Subscribe to the Drupal Security RSS feed
• Send vulnerabilities and patches to the Drupal Security Team
Q & A

Erich Beyrent, erich@commonplaces.com
Arian Evans, arian.evans@whitehatsec.com