Drupal Security: The Cheat Sheet

(for Drupal 7.x)

Golden rules

- Input is the root of all evil.
- Drupal does not filter user input; it stores exactly what the user typed.
- It is your responsibility to filter user data when it is displayed.

Types of security issues

- **SQL injection**: Allows the modification of SQL queries to bypass access control, destroy data, etc. Use db_query with proper place-holders.

- **Access Bypass:** Proper access checking is not performed; allows unauthorized users access to administrative functions by manipulating URLs, for example. Remember to use user_access, node_access, etc. appropriately

- **Cross-Site Scripting (XSS)**: Allows the injection of unfiltered HTML, including JavaScript that can be used to steal cookie data, intercept user input, etc. Properly filter user input with check_plain and friends.

- **Arbitrary code execution**: The most critical of all; PHP code can be directly embedded by a malicious user. Destruction is limited only by the user under whom PHP is running. Properly filter user input with check_plain and friends

Writing safe SQL queries

Always use placeholders in queries and never insert variables directly.

Bad

```
db_query("SELECT column FROM {table} WHERE
string = '$string' and number = $number");
```

Good

```
db_query("SELECT column FROM {table} WHERE
title = :title and id = :id", array(
  ':title' => $string,
  ':id' => $number
));
```

Displaying user input

check_plain(*$text*)
Convert special characters to plain text.

```
print '<p>'. check_plain($node->title) .'</
p>';
```

check_markup(*$text*, *$format_id*, *$langcode*, *$cache*)
Run filters on a piece of text.

```
print check_markup($user->signature);
```

check_url(*$url*)
Strip out harmful protocols in URLs.

```
print '<a href="/'. check_url($url) .'">';
```

filter_xss_admin(*$string*)
Very permissive XSS/HTML filter for admin-only use.

```
print filter_xss_admin($forum->description);
```

Working with t() placeholders

@example
text is run through check_plain().

```
$output = t('There are currently @count
users online.', array('@count' =>
$countvar));
```

Outputs: There are currently 9 users online

%example
text is run through drupal_placeholder(), which in turn runs check_plain().

```
$output = t('The user %name has just
registered.', array('%name' => $user-
>name));
```

Outputs: The user fred has just registere

!example
text is displayed as-is with no filtering; normally you should **NOT** use this unless output is being used for e-mail *only*.`$output = t('View this full post at !url'), array('!url' => 'http://www.example.com/');`

Outputs: View this full post at http://www.example.com/

Resources

Writing secure code: http://drupal.org/writing-secure-code

XSS Cheat Sheet: http://ha.ckers.org/xss.html

Security announcements: http://drupal.org/security

OWASP Testing Project: http://www.owasp.org/index.php/OWASP_Testing_Project

Common pitfalls

Whenever you are displaying content from a query to a user, use a dynamic query and tag it with the appropriate tag to make sure access checking is done.

Bad

```
$result = db_query('SELECT title FROM {node}
n WHERE ...');
```

Good

```
$query = db_select('node', 'n');
$result = $query
  ->fields('n', array('title'))
  ->condition(...)
  ->addTag('node_access');
```

Guard session IDs as much as possible. Do not print them into a page or send them as part of an AJAX request.

Never rely only on client-side (JavaScript) validation; always have server-side code performing final checks.

Never pass in an array of input directly into a query. Let the database layer convert the array into placeholders for you:

Bad

```
db_query("SELECT t.s FROM {table} t WHERE
t.field IN (%s)", implode(',', $user_array));
```

Good

```
db_query('SELECT t.s FROM {table}
IN(:user_array)', array(':user_array' =>
array(1, 2, 3)));
```

Titles (node, block, page...), watchdog messages, form element titles and descriptions must be escaped.

Menu items, breadcrumbs, block descriptions, link titles passed into l(), and output from theme('username') are already escaped for you.

When in doubt, look at what core does. Or just escape regardless to be safe.

Lullabot loves you.