September 28, 2016

DUBLIN
DRUPALCON

Cracking Drupal

Security concepts and pitfalls

Peter Wolanin
Moshe Weitzman

Track: PHP

https://events.drupal.org/dublin2016/sessions/cracking-drupal

Special thanks to Klaus Purer for creating the original talk and slides

Security strategies

- **Trust** - who can do what
- **Principle of least privilege** - each site user should have only the permissions necessary to do their job
- **Defense in depth** - multi layered protection to have fallbacks
- **Software updates** - rule out obvious exploits in Drupal, PHP, operating system, browser etc.

OWASP Top 10

- Open Web Application Security Project
- List of most critical security risks
- Assessment of attack vector, weakness and impact
- Updated every few years - 2017

https://www.owasp.org/index.php/Top_10_2013

1. Injection

Attacker's input is directly interpreted

SQL injection:

```php
<?php
db_query("SELECT uid FROM {users} u WHERE
           u.name = '" . $_GET['user'] . "'");
```

Remote code execution:

```php
<?php
eval($_POST['some_field']);
```

High impact! SA-CORE-2014-005 SQL injection

2. Authentication & sessions

- Choose good passwords, use TFA for admins
  - https://drupal.org/project/password_policy
  - https://drupal.org/project/tfa
- Hash your passwords (Drupal core covers this)
- Protect your session IDs
  Set up **HTTPS.** Do not send unencrypted session IDs.
  All HTTPS preferred. If you still need HTTP:
  - https://drupal.org/project/securelogin

3. Cross-Site Scripting (XSS)

- Attackers can inject Javascript tags
- All user input must be sanitized before printing to HTML
- (admin) user interaction is required - beware redirects

Reflected XSS example:

```
<?php
print 'You are on page number ' . $_GET['number'];
```

Penetration test: `<script>alert('XSS');</script>`

Persistent XSS

Attacker's Javascript is be stored in the database.

Vulnerable code, because of the node title:

```php
<?php
foreach ($nodes as $node) {
  $rows[] = array($node->nid, $node->title);
}
$render_array = array('#theme' => 'table','#rows' => $rows);
return $render_array;
```

Preventing XSS

Escape the user input:

```
<?php
foreach ($nodes as $node) {
  $rows[] = array($node->nid, check_plain($node->title));
}
$render_array = array('#theme' => 'table','#rows' => $rows);
return $render_array;
```

Handling text securely: https://drupal.org/node/28984

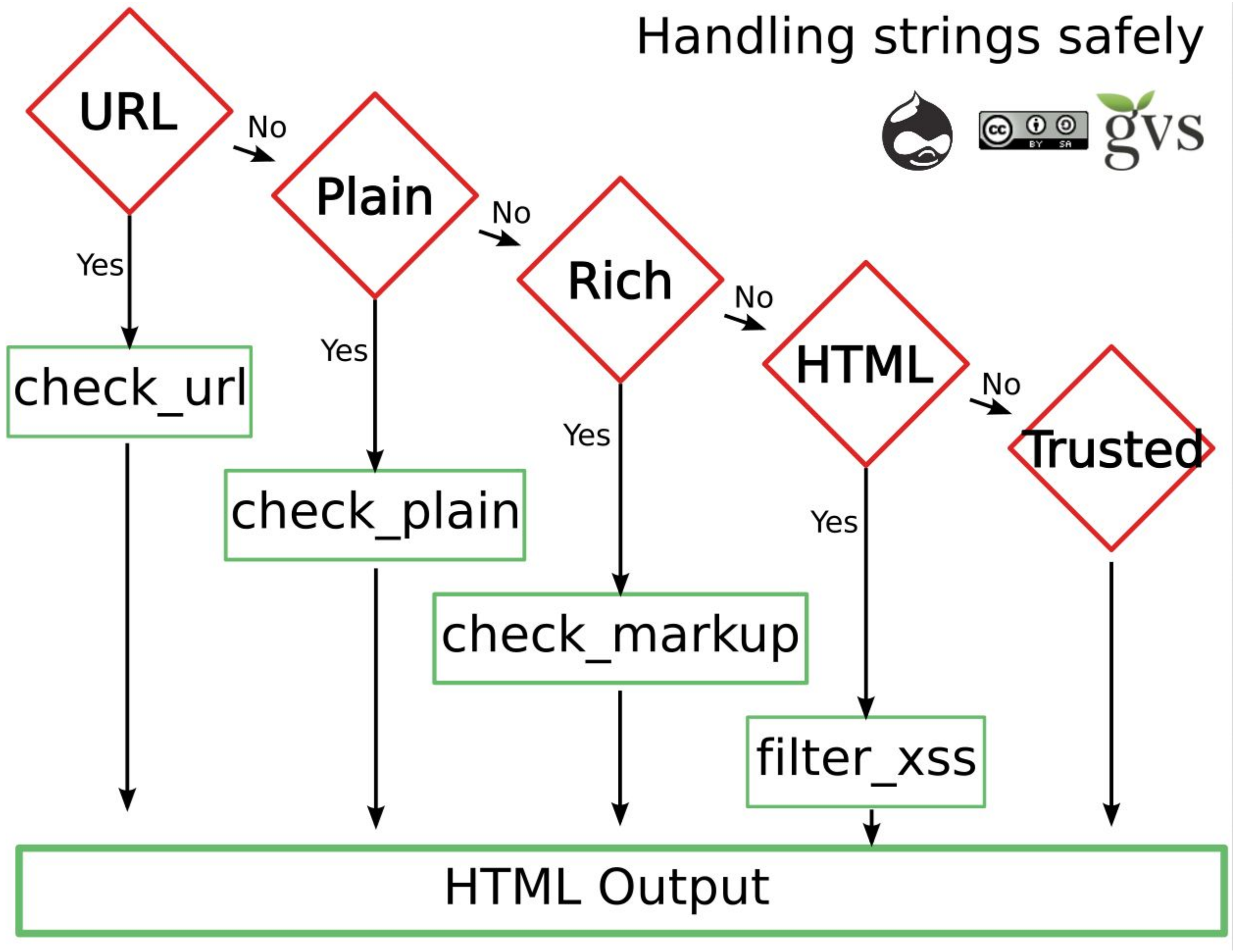
XSS is *Really* Dangerous

- Some people wrongly assume that the common test for XSS, an alert, is the actual attack. I.e. that it is at worst an annoyance or defacement.
- Anything that you as administrator can do, XSS can do also - change site settings, passwords, user roles, etc.

https://docs.acquia.com/articles/anything-you-can-do-xss-can-do-better

Filtering on output

When handling data, the golden rule is to store exactly what the user typed. When a user edits a post they created earlier, the form should contain the same things as it did when they first submitted it. This means that **conversions are performed when content is output**, not when saved to the database.



https://docs.acquia.com/articles/handling-strings-safely-drupal

Mitigating XSS

- What Drupal core does for us:
  - Drupal sets the HTTPOnly flag on session cookies to prevent cookie stealing in JS
  - User edit form: password change requires current password (since Drupal 7)
  - Text formats for different user roles
- Content Security Policy: W3C standard, no inline JS execution + JS domain whitelist
- We still need to rigorously escape user input.

4. Insecure Direct Object References

Category: Access bypass vulnerabilities

Happens rarely for Drupal, just use the user permission and access APIs.

Example - a custom page callback that displays a node without checking node access.

5. Security misconfiguration

- Display of PHP error reporting
  - Disable at /admin/config/development/logging
- PHP filter module, disable at /admin/modules
- PHP files writeable by the web server

Remove write permissions for www-data

```
-rw-r-----   1 deployer   www-data   index.php
drwxr-x--- 32 deployer   www-data   modules/
drwxrwx---   7 www-data   deployer   sites/default/files/
```

Docs: https://drupal.org/security/secure-configuration

Permissions

- Be careful with restricted, site-owning permissions (which roles do you trust?)
- Same for text formats (full HTML == XSS)
- Do not use the user 1 account in your daily work, it has all permissions - best practice block the account.
- User 1 name should not be “admin” or any other easily guessable name.

Private files configuration

Move the private files directory outside of the docroot to avoid direct downloads:

```
example.com
|+ conf
|- docroot
  |- index.php
  |- ... other Drupal files ...
|- private
  |- secret_picture.png
  |- ... other private files ...
|+
```

PHP file execution

- Drupal uses the front controller pattern: almost everything goes through **index.php**
- Disallow execution of PHP files in subfolders
- Prevents PHP execution in files directory

Apache example:

```
RewriteRule "^.+/.*\.php$" - [F]
```

Nginx example:

```
location ~* ^.+/.*\.php$ { deny all; }
```

Already present in .htaccess in Drupal 8

6. Sensitive Data Exposure

- **Encrypt sensitive data** such as credit card numbers in your database. Even better: don’t store them if you don’t have to (PCI, medical records, etc. compliance is hard).
- Again, use **HTTPS** for authenticated sessions (or better - all site traffic) to not transmit data in plain text.
- User **passwords** are properly hash-salted by Drupal 7.x+ core (use phpass for 6.x).

7. Missing Function Level Access Control

Access bypass in hook_menu():

```php
<?php
function mymodule_menu() {
  $items['admin/mymodule/settings'] = array(
    'title' => 'Admin configuration',
    'page callback' => 'drupal_get_form',
    'page arguments' => array('mymodule_admin_form'),
    'access callback' => TRUE,
  );
  return $items;
}
```

Using permissions

Protect your menu entries (routes):

```php
<?php
function mymodule_menu() {
  $items['admin/mymodule/settings'] = array(
    'title' => 'Admin configuration',
    'page callback' => 'drupal_get_form',
    'page arguments' => array('mymodule_admin_form'),
    'access arguments' => array('administer mymodule'),
  );
  return $items;
}
```

Correctly using node access

Limit the list of nodes with the node_access tag:

```
<?php
$records = db_select('node', 'n')
  ->fields('n')
  ->condition('type', 'expense_report')
  ->addTag('node_access')
  ->execute()
  ->fetchAll();
// ... load and render list of nodes somehow.
```

for 6.x: `db_rewrite_sql()`

8. Cross-Site Request Forgery (CSRF)

```
function mymodule_menu() {
  $items['mymodule/pants/%/delete'] = array(
    'title' => 'Delete pants',
    'page callback' => 'mymodule_delete_pants',
    'page arguments' => array(2),
    'access arguments' => array('delete pants objects'),
  ); return $items;
}
function mymodule_delete_pants($pants_id) {
  db_delete('mymodule_pants')
    ->condition('pants_id', $pants_id)->execute();
}
```

Example CSRF Exploit

- Attacker posts a comment somewhere:
  `<img src="http://example.com/mymodule/pants/1337/delete">`
- Chain of an attack:
  - Logged-in admin visits comment page
  - Browser fetches the image src and sends cookies along
  - Request is successfully authorized
  - Delete query is executed: pants 1337 is gone

http://epiqo.com/en/all-your-pants-are-danger-csrf-explained

Protecting against CSRF

- Write operations need to be protected. Use either:
  - Confirmation forms (use Form API)
  - Security tokens in the URL

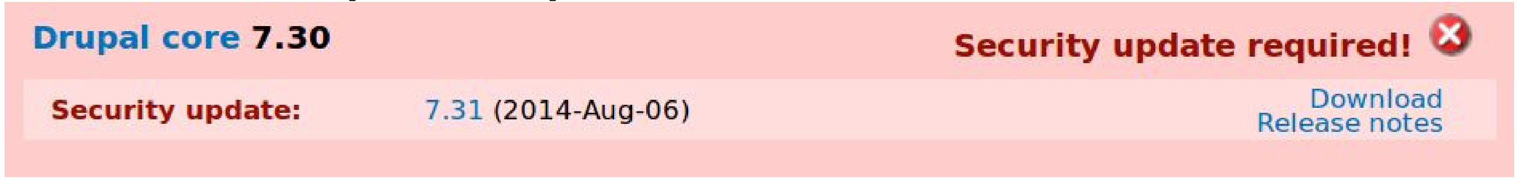
    ```
    http://example.com/mymodule/pants/1337/delete?token=tLBSLWTZVp
    Rmp1cD_I4hCKd2vS-dJbv6xxTICKr3DHM
    ```

- POST requests: always use the Form API! JavaScript can execute CSRF POST attacks, or you might submit a form on an malicious website.
- Docs: https://drupal.org/node/178896

9. Using Components with Known Vulnerabilities

Widespread attack vectors, often automated

- Update all server software regularly
- Monitor security mailing lists, RSS feeds etc.
- Enable Drupal’s update status notifications and emails

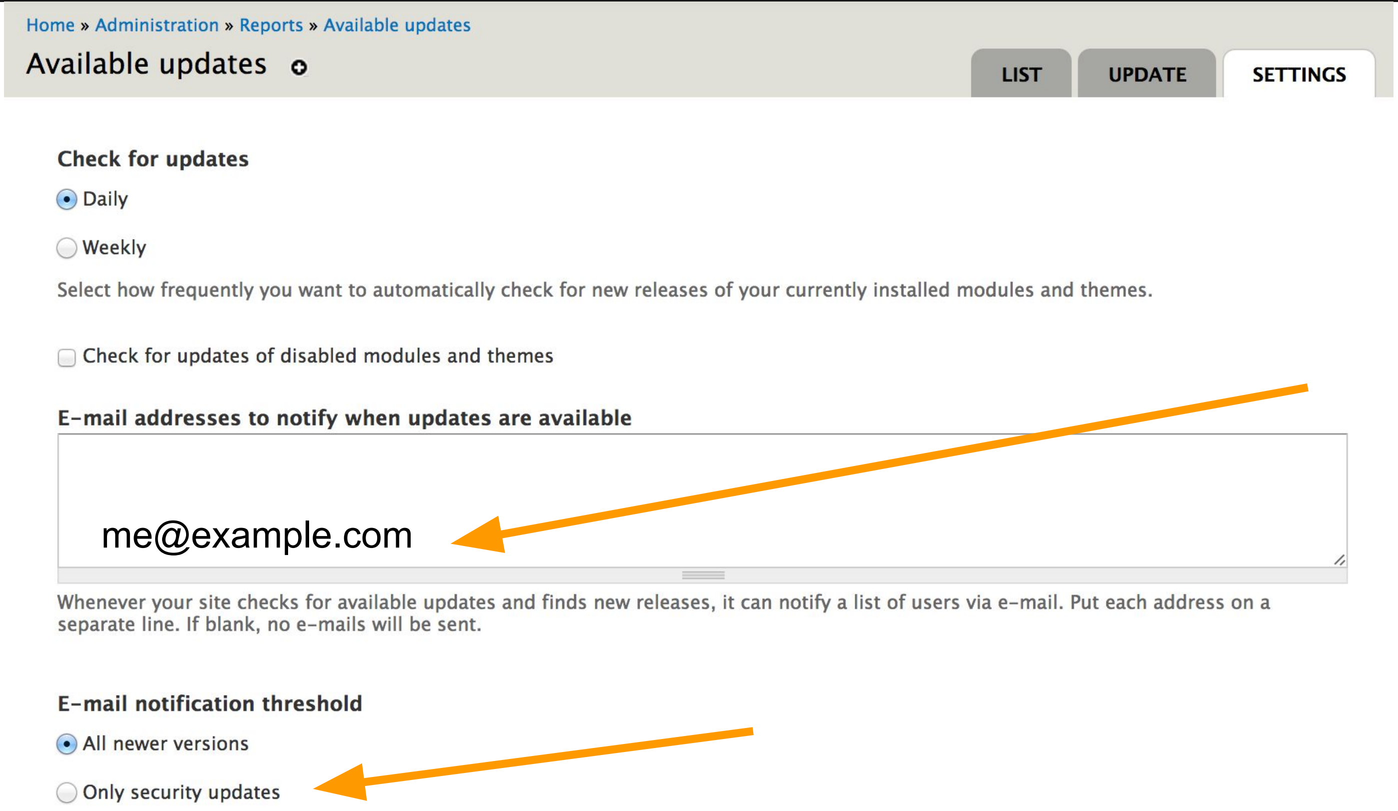


Drupal core 7.30 **Security update required!**

Security update: 7.31 (2014-Aug-06) Download Release notes

- Security advisories at https://drupal.org/security
- Disable software components (like modules) that are not used

Enabling Notifications: /admin/reports/updates/settings



Home » Administration » Reports » Available updates

Available updates

LIST | UPDATE | SETTINGS

Check for updates

[✓ Daily]

Weekly

Select how frequently you want to automatically check for new releases of your currently installed modules and themes.

☐ Check for updates of disabled modules and themes

E-mail addresses to notify when updates are available

me@example.com

Whenever your site checks for available updates and finds new releases, it can notify a list of users via e-mail. Put each address on a separate line. If blank, no e-mails will be sent.

E-mail notification threshold

[✓ All newer versions]

Only security updates

10. Unvalidated Redirects and Forwards

Vulnerability:

```
<?php
drupal_goto($_GET['target']);
```

Exploit example that redirects to evil.com:

```
http://example.com/cart?target=http%3A%2F%2Fevil.com
```

Perfect vector for phishing attacks. Correct:

```
<?php
if (!url_is_external($_GET['target'])) {
  drupal_goto($_GET['target']);
}
```

Do you see the pattern?

- Don’t trust any user provided data in the URL, the request, or content in the database
- Attackers use browser features to perform actions behind the user’s back (XSS, CSRF, open redirects)
- Attackers use known vulnerabilities and automated tools to mass-hijack sites

Be prepared for an attack

- Is your code in version control (git, svn, etc)?
- How often do you make full **backups**?
- Do you have separate login for each admin?
- If you are responsible for server (VPS) software do you keep it up to date?
- Do you have an out-of-band access method (e.g ssh + drush vs. web login)?
- Do you know where to find the Drupal watchdog log, web server log, syslog etc?

How to recover from an attack

- Determine what was compromised and when - after making a copy of the site
- Restore from backup
- Update code (and server software)
- Change all passwords and keys
- Audit your code (custom modules first!)
- Save and then scan logs for traces of the attacker (Drupal watchdog log, web server log, syslog etc.)

Useful security modules

- Security Review: check your site for misconfiguration https://drupal.org/project/security_review
- Paranoia: no PHP eval() from the web interface https://drupal.org/project/paranoia
- Seckit: Content Security Policy, Origin checks against CSRF, XSS https://drupal.org/project/seckit

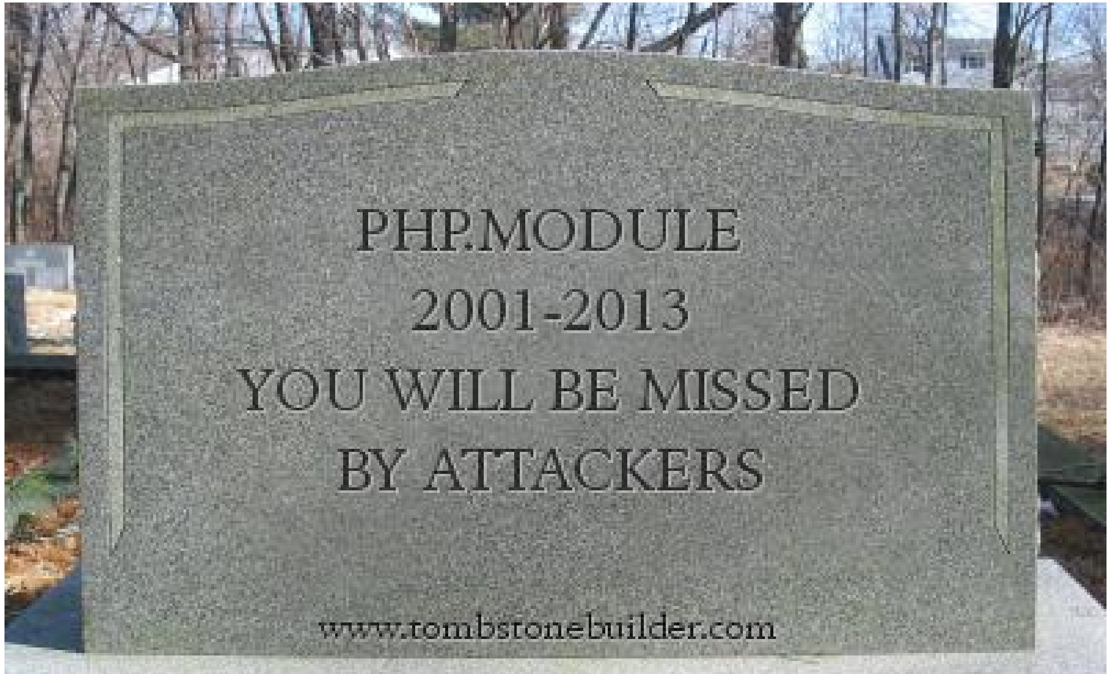
Security improvements in Drupal 8

- Twig auto-escape in templates
- Forbid PHP execution in subfolders in .htaccess
- CSRF token support in the routing system
- Hashed session IDs in the DB
- HTTPS peer verification in HTTP client (Guzzle)
- Permissions split up like “administer users”

https://dev.acquia.com/blog/drupal-8/10-ways-drupal-8-will-be-more-secure/27/08/2015/6621

Security improvements in Drupal 8

PHP module removed from core



Drupal Security Team

- https://www.drupal.org/security-team
- Coordinates security releases with maintainers
- Responsible disclosure: private issues at https://security.drupal.org/
- Defines security policies, risk levels

Resources

Security handbook: https://drupal.org/writing-secure-code

Secure configuration: https://drupal.org/security/secure-configuration

XSS: https://docs.acquia.com/articles/introduction-cross-site-scripting-xss-and-drupal

Security advisories: https://www.drupal.org/security

Site and book: http://crackingdrupal.com/

DUBLIN
DRUPALCON

THANK YOU! QUESTIONS?

BI<>RAFT
ENTERPRISE SAFETY, COMPLIANCE & TRAINING SOFTWARE

Peter Wolanin
drupal.org/user/49851 IRC: pwolanin

Moshe Weitzman
drupal.org/user/23 IRC: moshe

DUBLIN
DRUPALCON

JOIN US FOR CONTRIBUTION SPRINTS FRIDAY, SEPTEMBER 30

First Time Sprinter Workshop - 9:00-12:00 - Wicklow 2A

Mentored Core Sprint - 9:00-18:00 - Wicklow Hall 2B

General Sprints - 9:00 - 18:00 - Wicklow Hall 2A

DUBLIN
DRUPALCON

WHAT DID YOU THINK?

Evaluate This Session

events.drupal.org/node/13773

THANK YOU!