DrupalCon
SEATTLE 2019
APRIL 8-12

Please open
http://vuln.rocks/crackdru

Cracking Drupal

Security concepts and pitfalls

Peter Wolanin
Michael Hess

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

About The Presenters

Peter Wolanin

- Drupal Security Team member since 2008
- Core contributor to 5,6,7,8 and module maintainer, but often distracted
- Thinks using the plugin system for menu links was a brilliant stroke...

Michael Hess

- Security Team member since 2011, team lead.
- Teaches and runs Drupal sites at the University of Michigan
- Has been known to kill a Drupal site just to watch it die...

Agenda

DrupalCon SEATTLE 2019

- Review the top 10 types of web vulnerabilities
- Learn some best practices
- Answer questions
- Have fun along the way

When you think of security what words come to mind?

http://vuln.rocks/crackdru

http://vuln.rocks/crackdru

CIA Triad

Confidentiality, integrity and availability, also known as the **CIA triad**, is a model designed to guide policies for information security within an organization. The model is also sometimes referred to as the AIC **triad** (availability, integrity and confidentiality) to avoid confusion with the Central Intelligence Agency.

OWASP Top 10

DrupalCon SEATTLE 2019

- Open Web Application Security Project
- List of most critical security risks
- Assessment of attack vector, weakness
- Updated every few years - 2017 is the Latest version.

owasp.org/index.php/Category:OWASP_Top_Ten_Project

What vulnerabilities have you heard of?

The OWASP Top 10

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XXE)
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging&Monitoring

1. Injection

Attacker's input is directly interpreted as code

SQL injection:

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

Remote code execution:

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

Highest Impact!

- Injection attacks can completely compromise a site and possibly also the underlying servers.
- SA-CORE-2014-005 SQL injection.
- SA-CORE-2018-002 & SA-CORE-2018-004 RCE via form API.
- SA-CORE-2019-002 phar file execution.
- SA-CORE-2019-003 RCE via unserialization.

SQL Injection question

2. Broken Authentication

- Choose good passwords, use TFA for admins (preferably all users)
  - 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 should be used for all sites now (http/2).

3. Sensitive Data Exposure

- **Encrypt sensitive data** such as credit card numbers in your database. Better: don’t store them if you don’t have to (PCI, HIPPA, etc. compliance is hard).
- Know your risk level
- Weak keys or poor key management can still expose.
- Use **HTTPS** for all traffic
- User **passwords** are properly hash-salted by Drupal 7.x+ core, but weak passwords can still be cracked.

http://vuln.rocks/crackdru

4. XML External Entities (XXE)

May be used to expose private or system file content, conduct a DoS attack, scan local networks, and more.

Affects SOAP, SAML, OPML feeds, or any other place XML is parsed.

XML parsers may allow external entities by default - beware any vendor libraries. Consider the source of any XML you are parsing.

5. Broken Access Control

Category: Access bypass vulnerabilities

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

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

Missing Access Control

Access bypass in hook_menu() (Drupal 7):

```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;
```

Missing Access Control

Access bypass in routing.yml (Drupal 8):

```
mymodule,admin_settings:
  path: '/admin/mymodule/settings'
  defaults:
    _form: '\Drupal\mymodule\Form\AdminSettingsForm'
    _title: 'Admin configuration'
  requirements:
    _access: 'TRUE'
```

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;
```

Using permissions

Protect your routes:

```
mymodule,admin_settings:
  path: '/admin/mymodule/settings'
  defaults:
    _form: '\Drupal\mymodule\Form\AdminSettingsForm'
    _title: 'Admin configuration'
  requirements:
    _permission: 'administer mymodule'
```

Correctly using node access

Limit the list of nodes with the node_access tag:

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

6. 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

Write permissions for www-data pose a risk

```
-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; }
```

7. Cross-Site Scripting (XSS)

- Attackers can inject Javascript tags
- All user input must be sanitized before printing 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
<?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

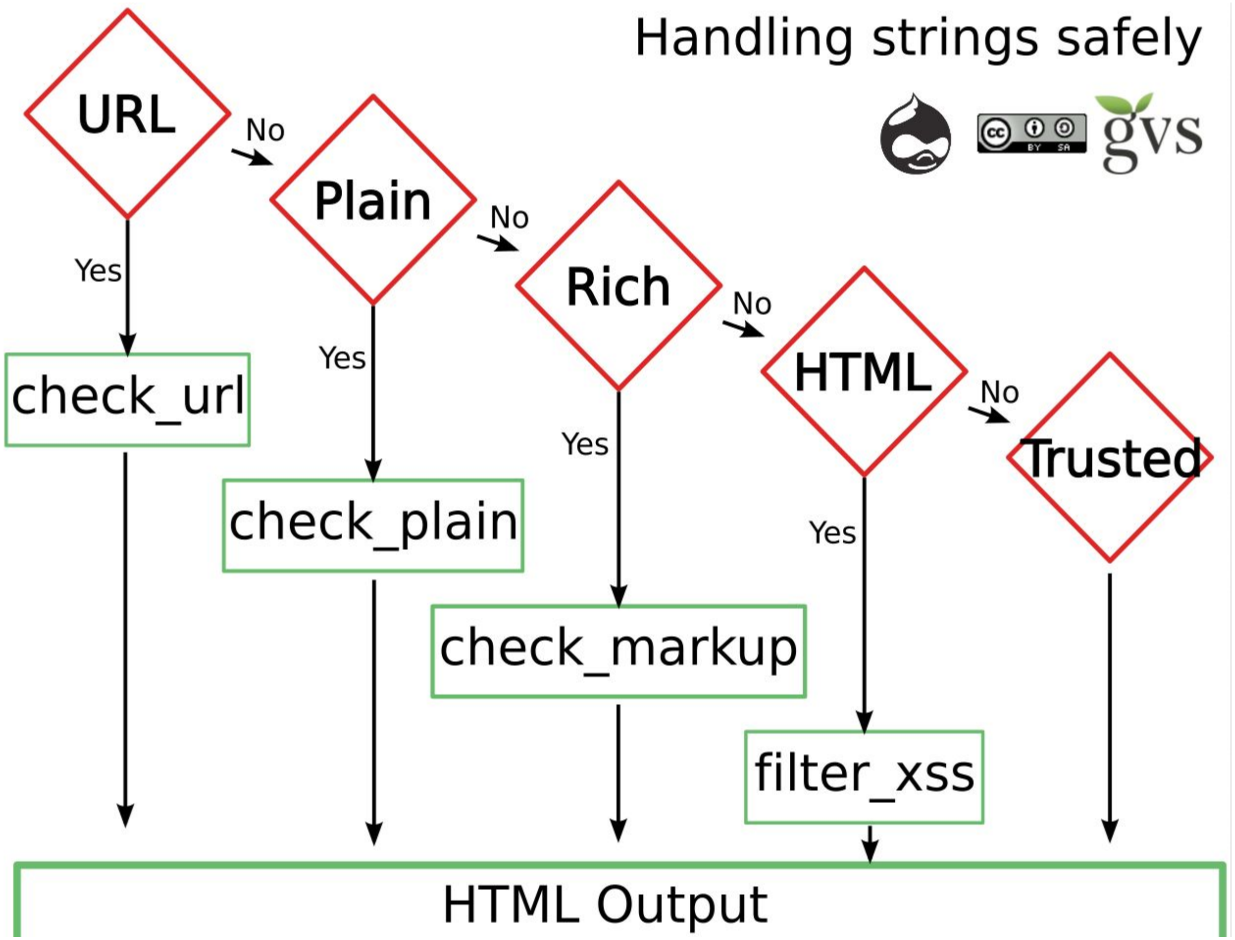
XSS is *Really* Dangerous

- 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://support.acquia.com/hc/en-us/articles/360005028694-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.



http://vuln.rocks/crackdru

Mitigating XSS

- What Drupal core does for us:
    - Sets HTTPOnly flag on session cookies to prevent JS
    - Password change requires current password
    - Text formats for different user roles
    - Autoescape in Drupal 8
- Content Security Policy: W3C standard, no inline JS execution + JS domain whitelist
- We still need to rigorously escape user input.

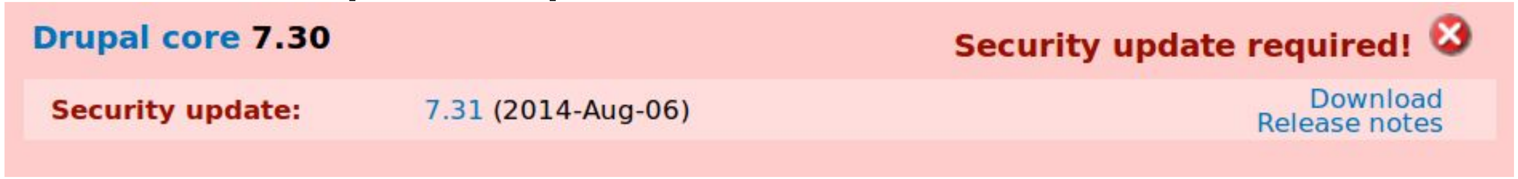
8. Insecure Deserialization

- Unserialization can be exploited in PHP via magic methods like __destruct() to delete files or even execute code.
- SA-CORE-2019-003 was a result of serialized strings being parsed for some fields as part of API calls.
- Never use PHP serialize format for cookies, form data, etc. - use a safe format like JSON.

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

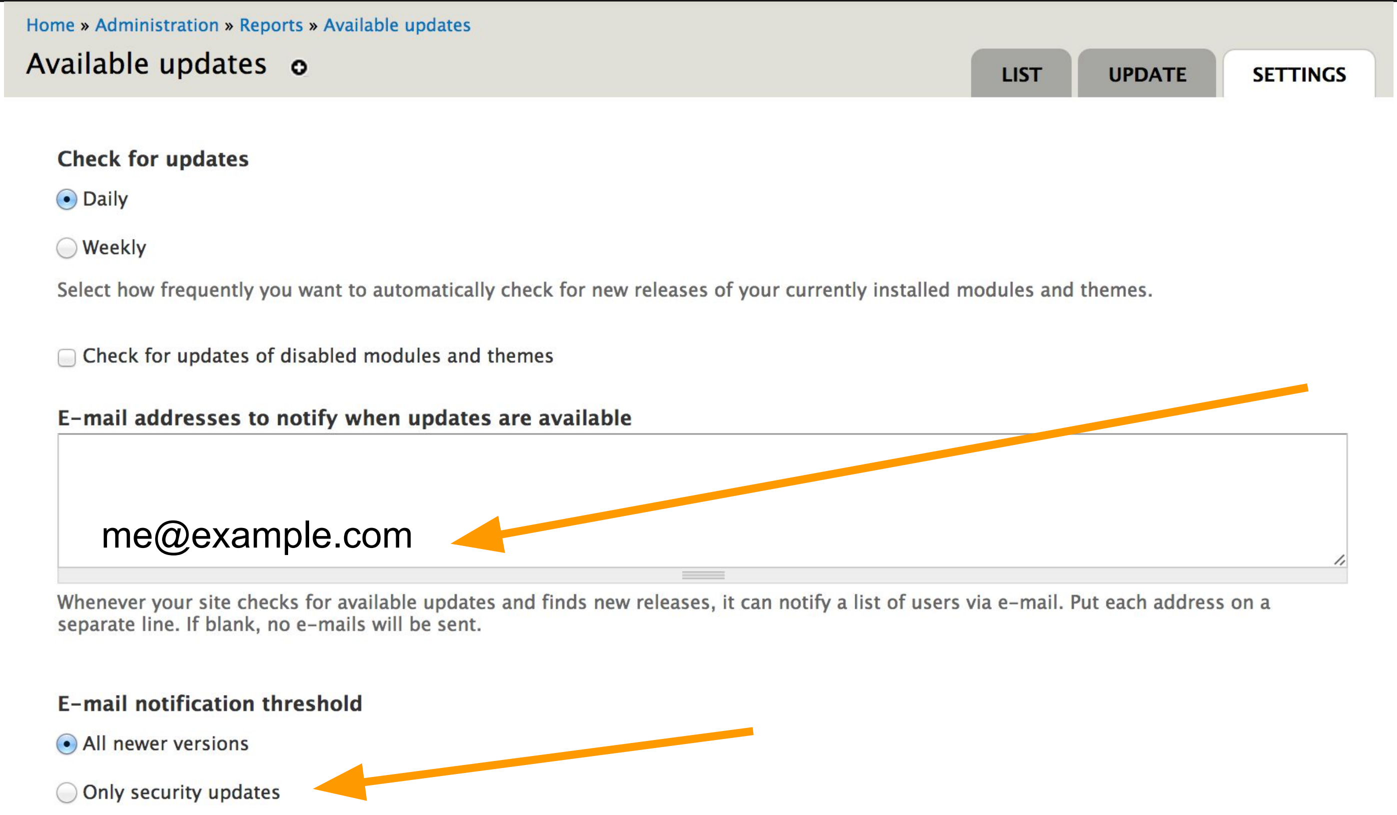


Drupal core 7.30 **Security update required!**

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

Drupal 7 will be EOL

Drupal 7 will be EOL in November of 2021.

(Drupal 8 will also be EOL in November of 2021, but the upgrade path is much easier)

10. Insufficient Logging & Monitoring

- **What is happening to your Drupal sites right now?** If you were experiencing unusual requests or logins would you know, or be able to find out later?
- If the Drupal or system logs were deleted do you have a central copy?
- Recent high-profile hacks were potentially going on for months before being detected.

Read your logs!

Use services that help with finding abnormalities.

Have centralized logging

Not top 10: 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

drupalsun.com/klausi/2013/02/26/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 (automated in Drupal 8)

    `http://example.com/mymodule/pants/1337/delete?token=tLBSLWTZVpRmp1cD_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

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

Check time!

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 (or VPS / VM) 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

DrupalCon SEATTLE 2019

- 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

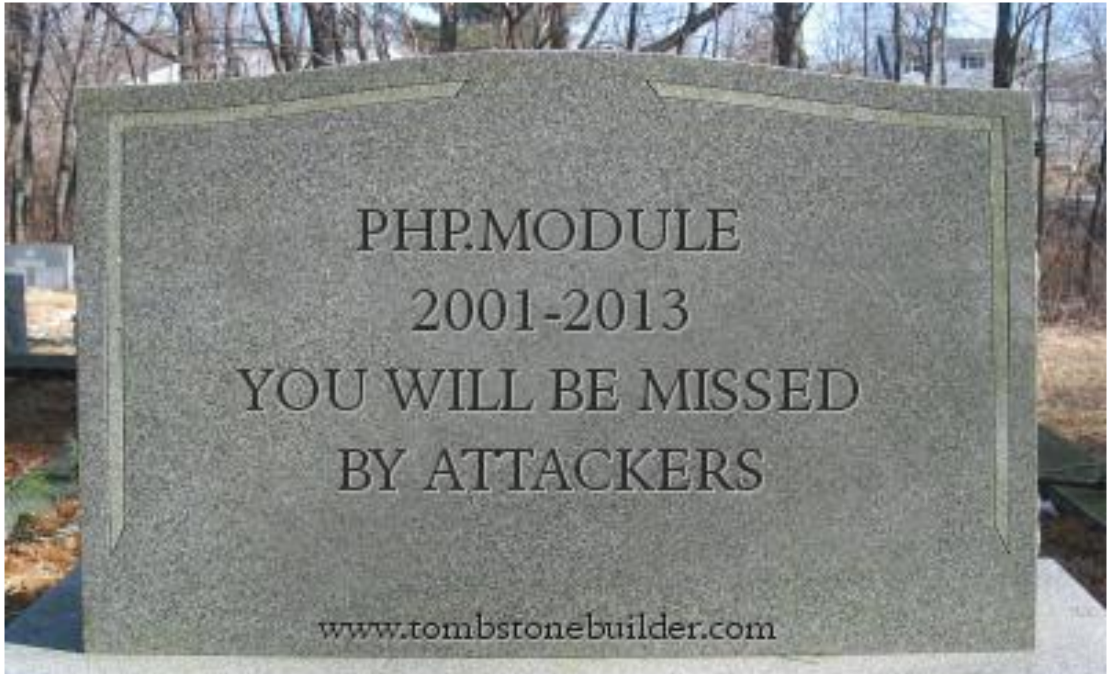
Security improvements in Drupal 8

- Twig auto-escape in templates
- Forbid PHP file 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

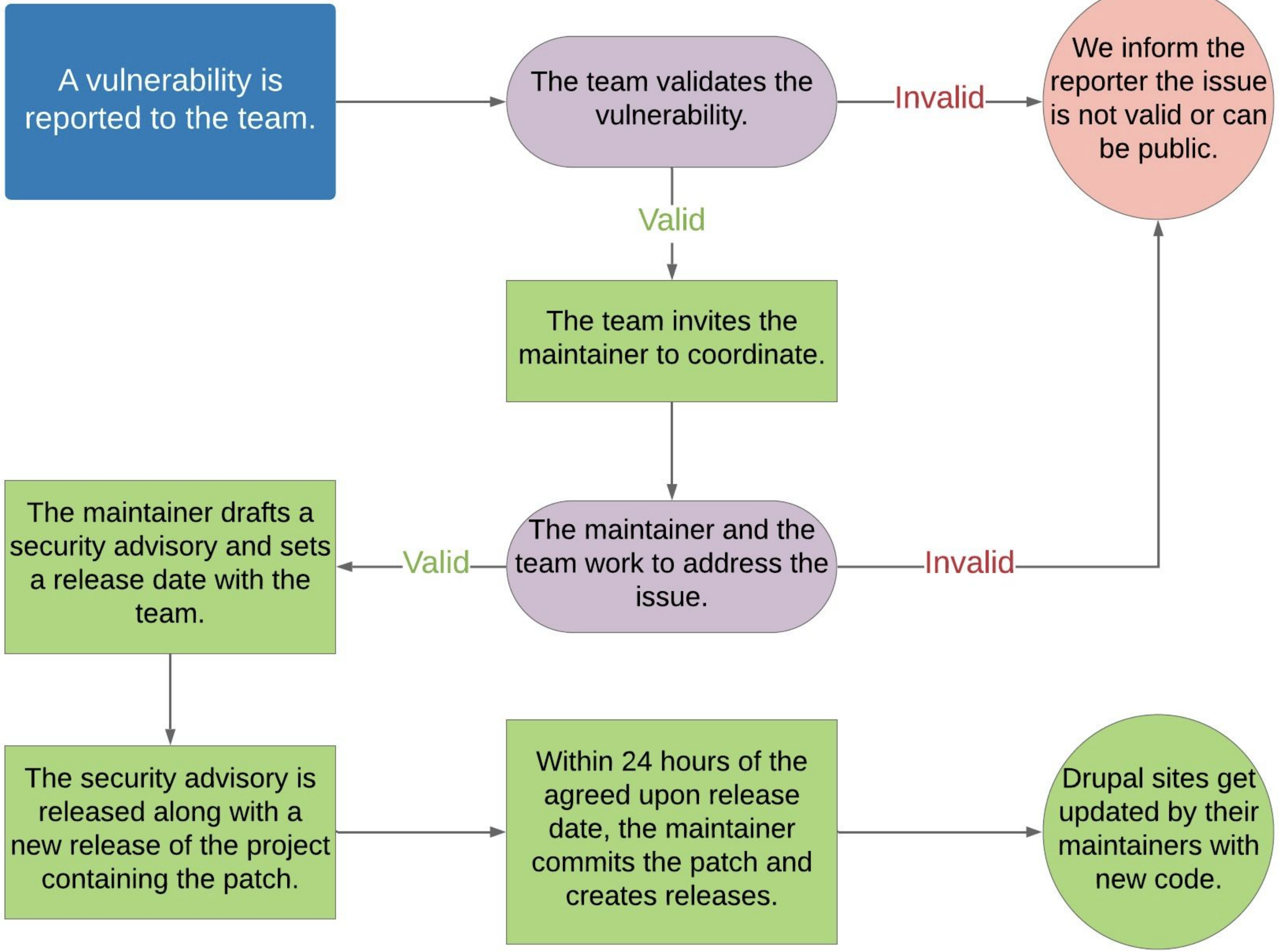
Drupal Security Team

- 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

Follow the Security Team

- On Twitter: *twitter.com/drupalsecurity*
- Via email: on your *drupal.org* user edit page under newsletters
- Via Web: *drupal.org/security* and *drupal.org/security/contrib*
- In Drupal Slack, the #annoucments channel and the #security-questions channel

Security Team General processes


A vulnerability is reported to the team.
The team validates the vulnerability.
Invalid
We inform the reporter the issue is not valid or can be public.
Valid
The team invites the maintainer to coordinate.
The maintainer and the team work to address the issue.
Invalid
Valid
The maintainer drafts a security advisory and sets a release date with the team.
The security advisory is released along with a new release of the project containing the patch.
Within 24 hours of the agreed upon release date, the maintainer commits the patch and creates releases.
Drupal sites get updated by their maintainers with new code.

http://vuln.rocks/crackdru

BEST PRACTICES

A quick summary

Best Practices

Best practices can guide you as to where to start with or invest in security.

Security is not a checkbox ✓, it has to be part of your workflow (and mindset).

openconcept.ca/drupal-security-best-practices-practical-guide

BRUSHING YOUR TEETH IS A BEST PRACTICE

- For security, you can't check a list and be done.
- You must keep working at it. It is a process, not a one-time task.

Your hosting matters

- Is your primary business hosting? If not, pay someone to host your site.
- Shared hosting normally runs the webserver as the owner of the file system (cpanel).
- Multiple sites on a server often use a common account for all sites (www-data, nobody, etc).

Unless you understand multisite, don't use it.

Multisite by default can be very insecure.

Unless you have a deep understanding of apache/nginx and file permissions, multisite is insecure.

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.

Resources

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

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

XSS:https://support.acquia.com/hc/en-us/articles/360004992074-Introduction-to-cross-site-scripting-XSS-

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

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

Drupal™ Steward

A new product
from the Drupal Association
and the Drupal Security Team

What is Drupal Steward

Peace of mind for Drupal Site Owners

A Web Application Firewall protecting sites from known vulnerabilities, **before** the vulnerability is disclosed and the update is released.

For more information:
drupal.org/blog/regarding-critical-security-patches-we-hear-your-pain

THANK YOU! QUESTIONS?

Peter Wolanin

drupal.org/u/pwolanin   slack: pwolanin

Michael Hess

drupal.org/u/mlhess   slack: mlhess

DrupalCon
SEATTLE 2019
APRIL 8-12

Join us for contribution opportunities

Friday, April 12, 2019

Mentored Core sprint	First time sprinter workshop	General sprint
9:00-18:00	9:00-12:00	9:00-18:00
Room: 602	Room: 606	Room: 6A

#DrupalContributions

DrupalCon
SEATTLE 2019
APRIL 8-12

What did you think?

Locate this session at the DrupalCon Seattle website:

https://events.drupal.org/node/22558

Take the Survey!

https://www.surveymonkey.com/r/DrupalConSeattle

Thank you!