Secure Drupal Development

Steven Van den Hout

Steven Van den Hout

@stevenvdhout

http://dgo.to/@svdhout

IS DRUPAL SECURE?

1

IS OPEN SOURCE SECURE?

MANY EYES MAKE FOR SECURE CODE

- ~~Security by obscurity~~
- Open code does not make it easier for hackers
- Open Source makes people look at it
- Popularity gets more eyes and more peer-reviews

- *Bad open-source software as bad*
- *as bad private software.*

OWASP

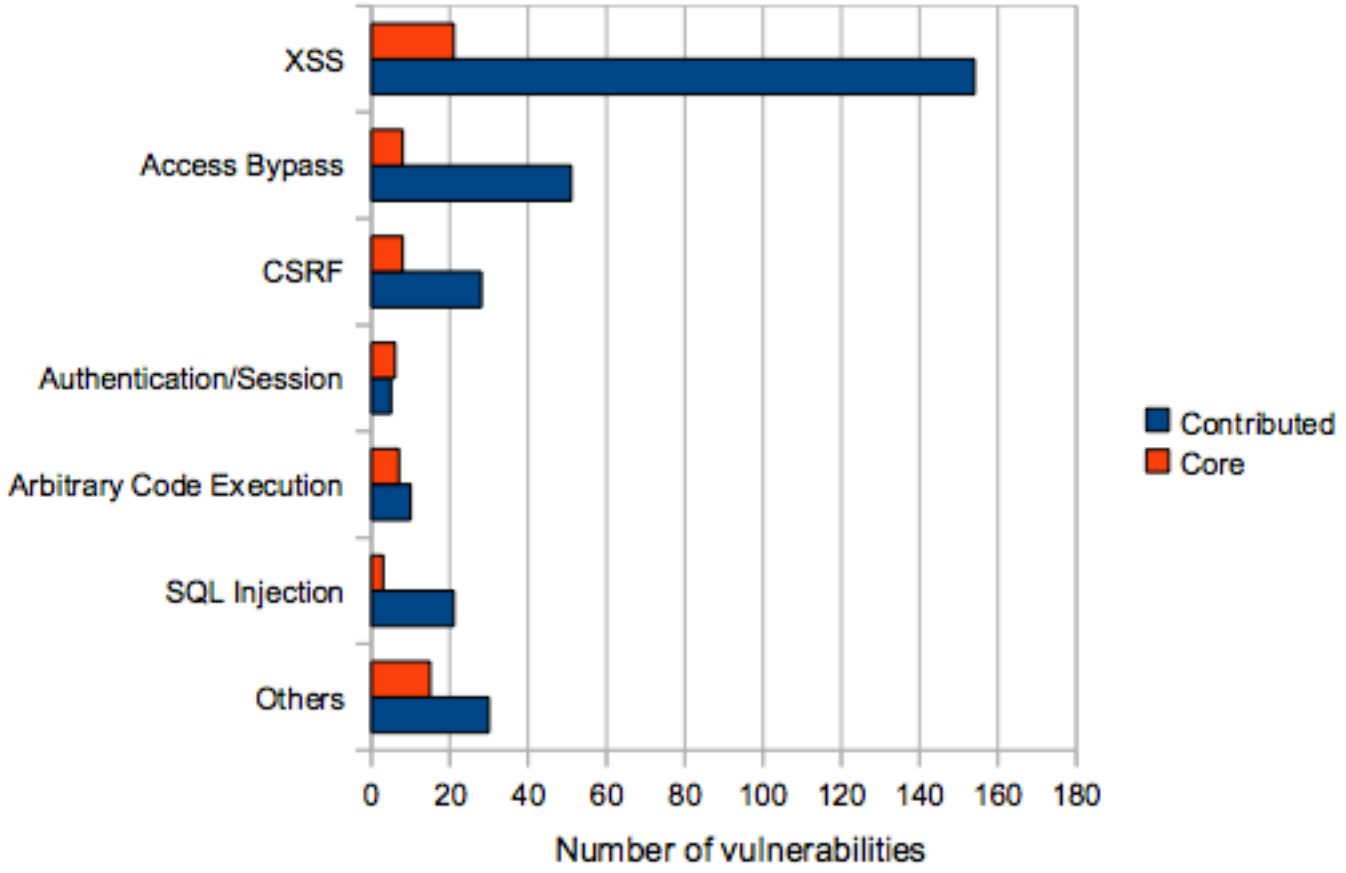
VULNERABILITIES

- Injection
- Cross Site Scripting - XSS
- Broken Authentication and Session Management
- Cross Site Request Forgery - CSRF
- Security Misconfguration
- Failure to Restrict URL Access
- Access bypas

OWASP
The Open Web Application Security Project

REPORTED VULNERABILITIES

Illustration 2: Number of vulnerabilities by type reported in SAs for Drupal core and contributed projects[4]



IS DRUPAL SECURE?

- Safe by design (Core and API)
- Security Team
- Highly organised
- Documented process for Security Advisories and Updates
- Thousands of maintainers, users and experts
- Support: Drupal 6/7, Core & Contributed Modules

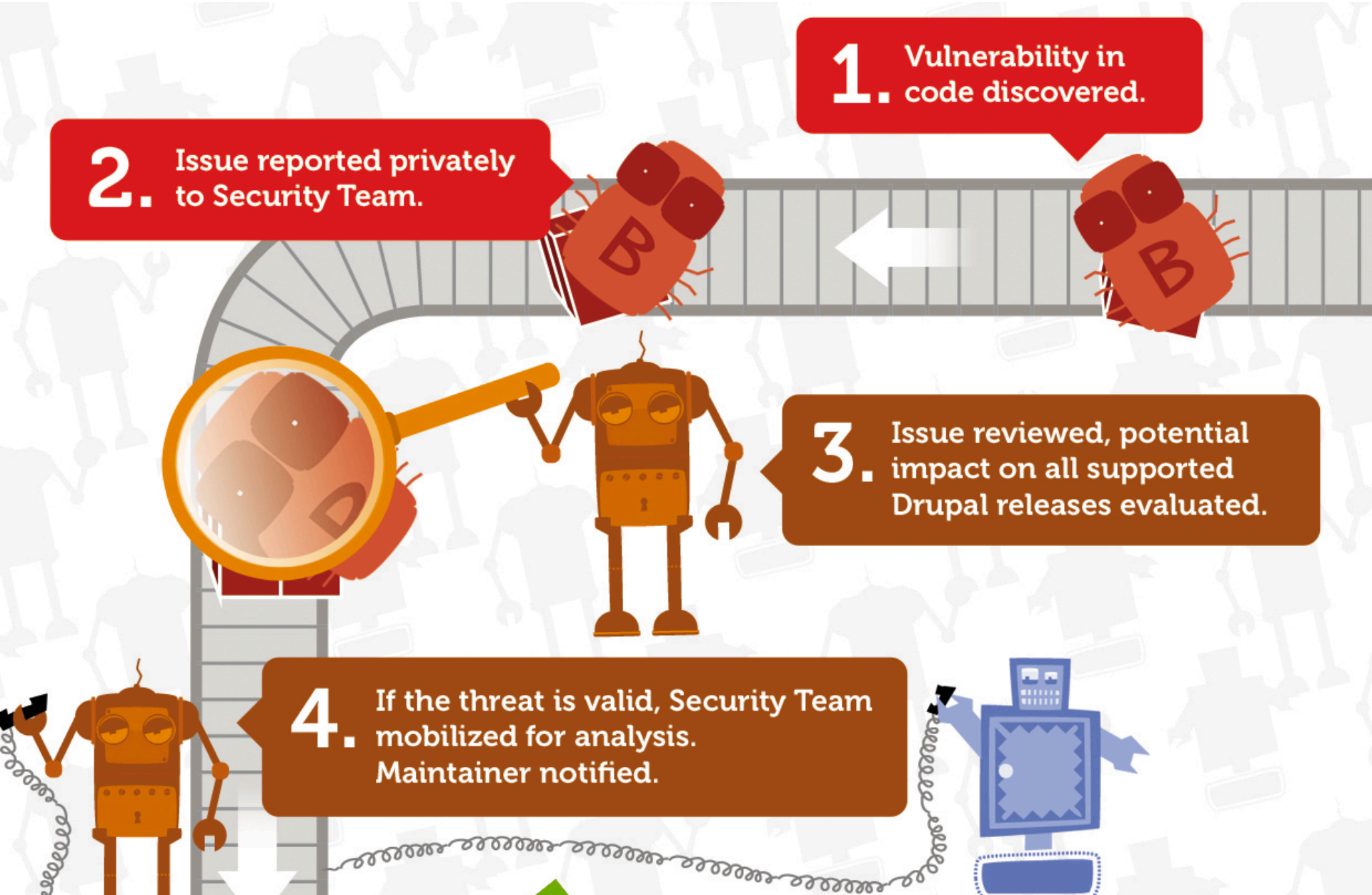
KEEP YOUR DRUPAL WEBSITE SECURE

2

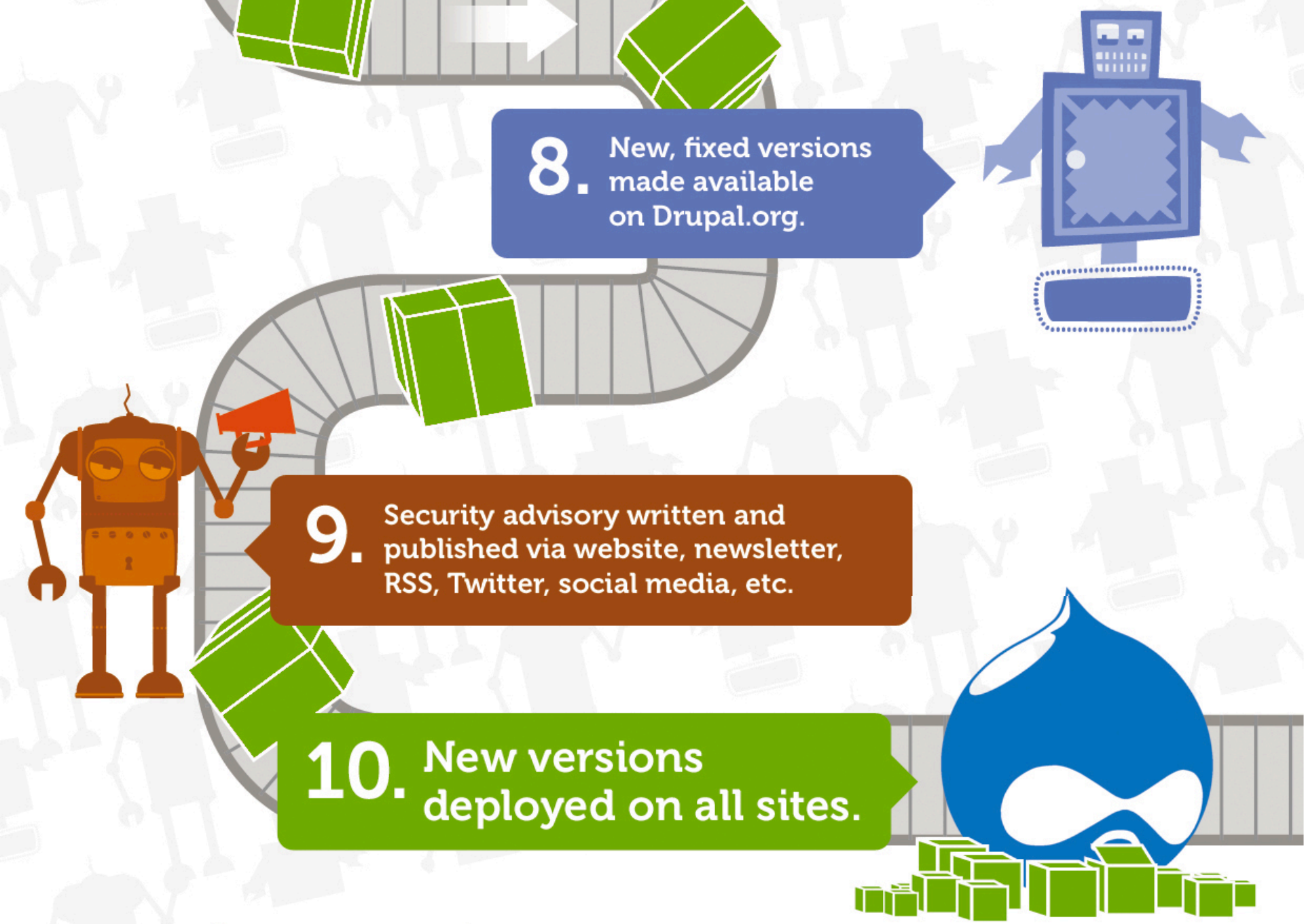
SECURITY IS A PROCESS

NOT AN EVENT

A DRUPAL SECURITY RELEASE




5.
Security Team provides support.
Maintainer fixes issue.
6.
Fixes reviewed and discussed.
7.
Code patches created and tested.



Sponsored by Acquia mogdesign.eu

For more information, go to drupalsecurityreport.org

PRIVATE DISCLOSURE

YOU'RE SAFE UNTIL RELEASE SECURITY UPDATE

UPDATES

Always stay up to date

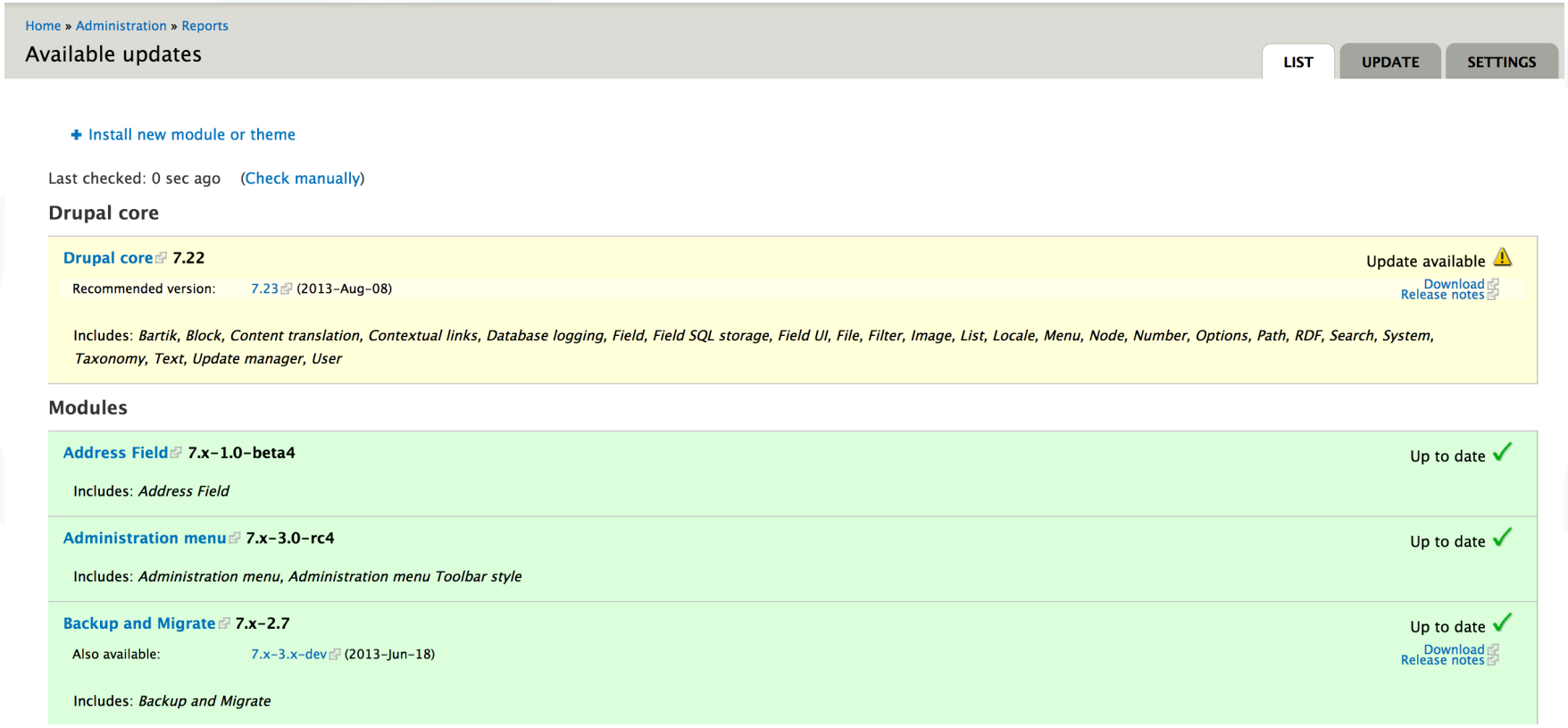
- Keep up with latest security releases

Update Workflow

- Hacked module + diff
- Drush up

UPDATE MANAGER

KNOW WHEN AN UPDATE IS NEEDED



STATUS MONITORING

INSIGHT INTO HEALTH OF YOUR DRUPAL WEBSITE

Tools

- Droptor.com (https://drupal.org/project/droptor)
- Acquia Insight (https://drupal.org/project/acquia_connector)
- Nagios (https://drupal.org/project/nagios)
- Drupalmonitor.com (https://drupal.org/project/drupalmonitor)
- ...

Subscription

calibrate

Site Visit manage

calibrate.be

- Dashboard
- Insight
  - Overview
  - Analysis
- SEO Grader
- Cloud
- Search

Last Data Update

Configuration check:
27 sec ago

View Full History

Overview

The Insight score for your site is representative of your security, performance, and best practices scores.

Insight Score

How is this calculated?

92%

Overview | Code | Server | Statistics

Analysis summary

Performance	18 of 22 issues resolved
Security	17 of 19 issues resolved
Best Practices	4 of 4 issues resolved

Insight Score

BUILD A SECURE DRUPAL WEBSITE

3

CONTRIBUTED MODULES

CONTRIBUTED MODULES

Quality assurance

- Usage
- Number of open issues
- Closed/Open ratio
- Response time

Good quality usually means good security

Manual code reviews for less used modules

UPDATES

Always stay up to date

- Keep up with latest security releases

Update Workflow

- Hacked module + diff
- Drush up

PATCHES

Contrib patches

Read the entire issue

Commit custom patches

Help out
Feedback from other users (maintainers)
Patch might get commited

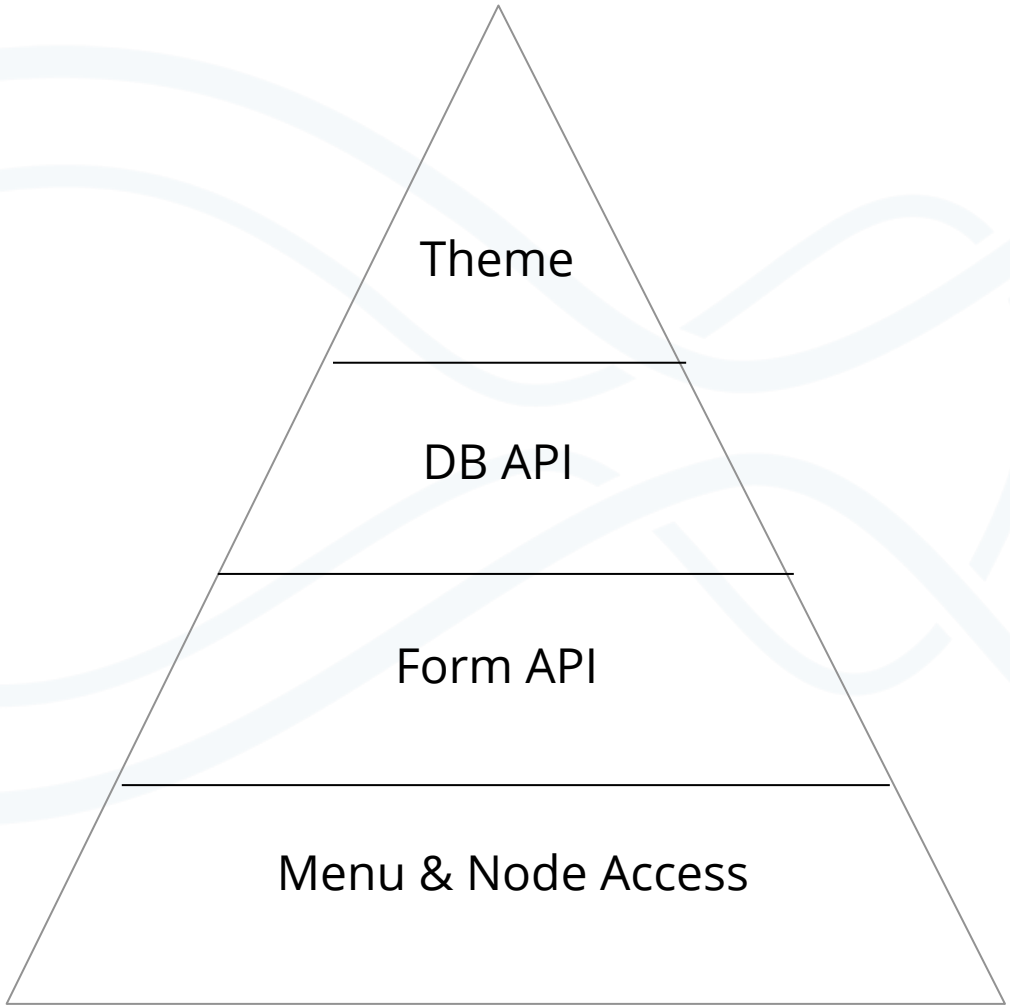
Patch management

Move module to patched
Create a patches.txt
Keep patches

CUSTOM

MODULES

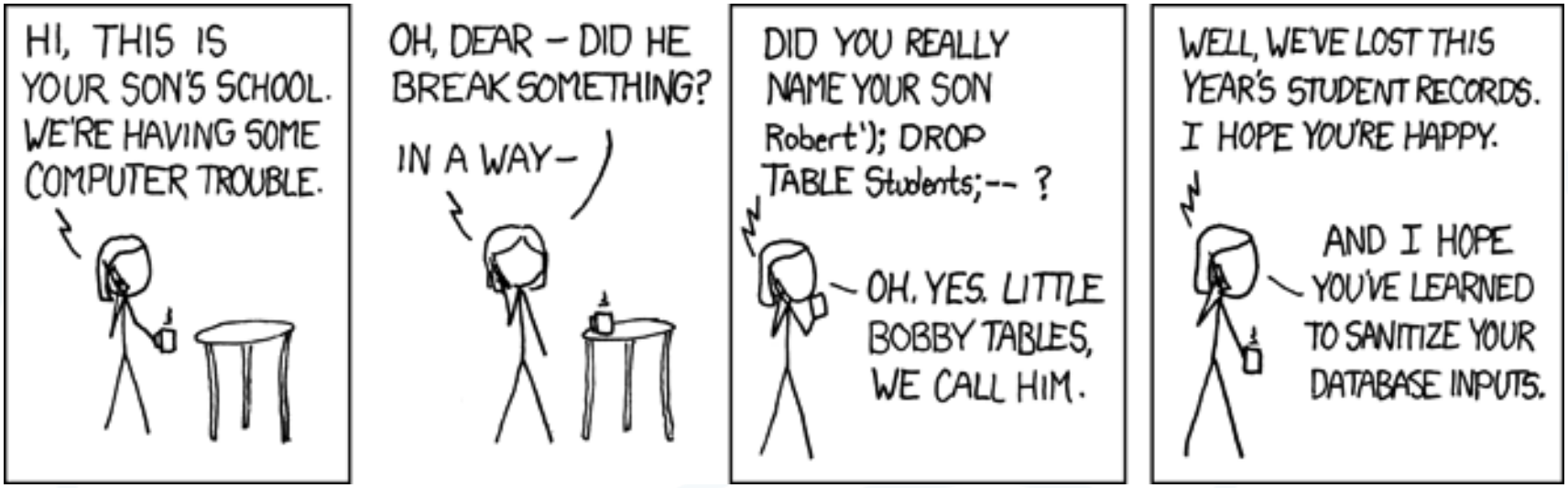
SECURITY PYRAMID



HACKS

AND HOW TO PREVENT THEM

SQL INJECTION



http://xkcd.com/327/

"SELECT * FROM user WHERE name = '$name'"

"SELECT * FROM user WHERE name = 'Robert'; DROP TABLE students;'"

SQL INJECTION

Placeholders

```
db_query(“SELECT * FROM users WHERE name = :user”, array(':user' => $user);
```

Dynamic Queries

```
$query = db_select('user', 'u')
        ->fields('u')
        ->where('name', $user)
        ->execute();
```

XSS (cross site scripting)

EXECUTING ABRITRARY JAVASCRIPT CODE ON THE PAGE

XSS (cross site scripting)

User Input

Title
Body
Log message
Url
Post
User-Agent
Headers

XSS (cross site scripting)

Validate forms

User input should never contain javascript

Form api

Never use $_POST variables
$form_state['values']

Form caching

XSS (cross site scripting)

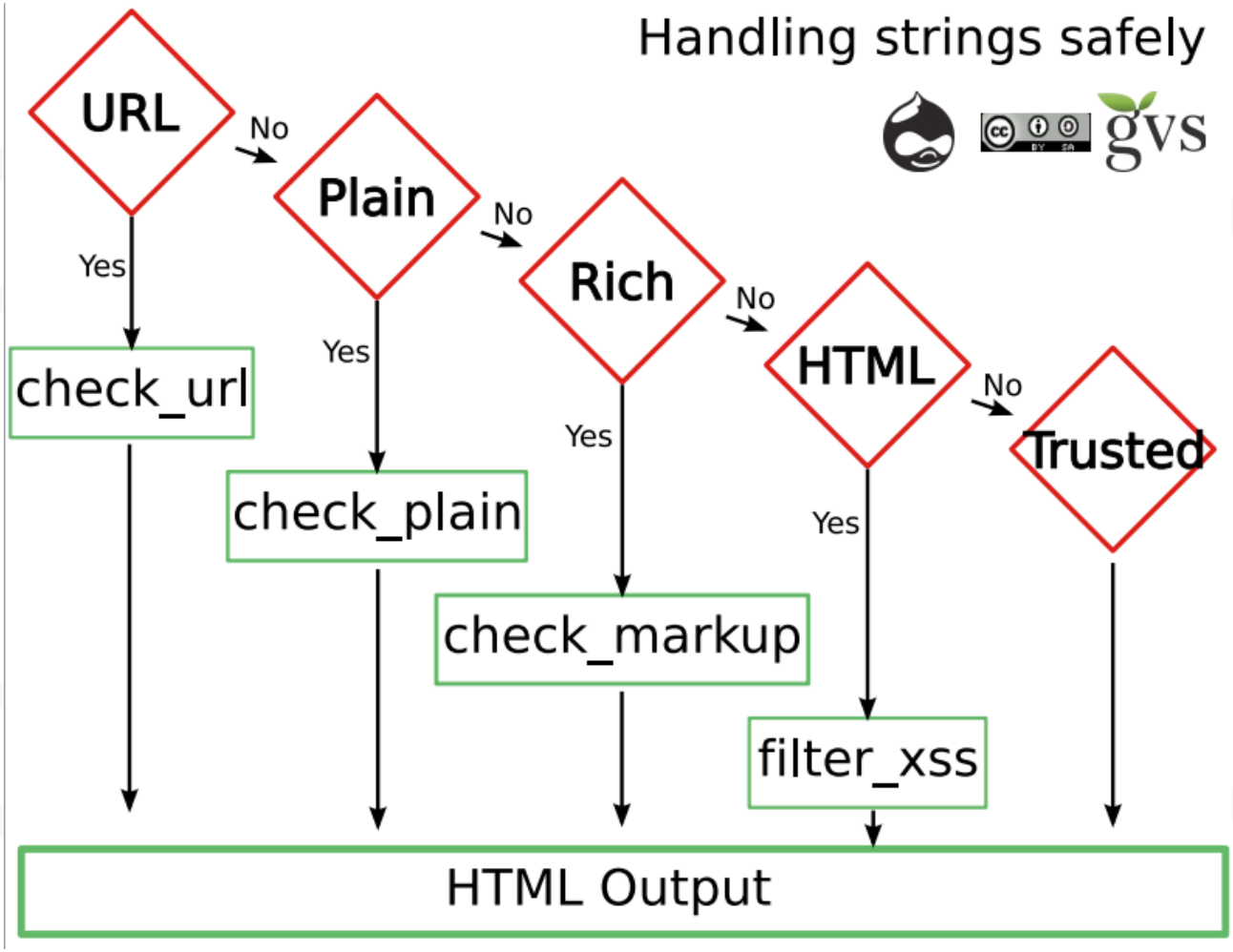
Input formats

Never use full_html

Filter Functions

check_url()
check_plain()
check_markup()
filter_xss()

XSS (cross site scripting)



http://drupalscout.com/knowledge-base/drupal-text-filtering-cheat-sheet-drupal-6

XSS (cross site scripting)

Functions

t()

l()

drupal_set_title()

@var => plain text
%var => plain text
!var => full html!

CSRF (cross site request forgery)

Taking action without confirming intent

```
<a href="/delete/user/1">Delete user 1</a>
```

Image Tag

```
<img src="/delete/user/1">
```

A hacker posts a comment to the administrator.
When the administrator views the image, user 1 gets deleted

CSRF (cross site request forgery)

Token (aka Nonce)

```
$token = drupal_get_token('foobar');
l('delete user 1', "delete/user/1/$token");

...

if (drupal_valid_token($received_token, 'foobar')) {
  // perform action
}
```

ACCESS BYPASS

VIEW CONTENT A USER IS NOT SUPPOSED TO

ACCESS BYPASS

View content a user is not supposed to

$query = db_select('node', 'n')->fields('n');
Also shows nodes that user doesn't have acces to

$query->addTag('node_access')

Rewrite the query based on the node_access table

ACCESS BYPASS

Bad custom caching

Administrator visits a block listing nodes.
The block gets cached

The cached block with all nodes is shown to the anonymous user

Add role id to custom caching

ACCESS BYPASS

Rabbit_hole module

Rabbit Hole is a module that adds the ability to control what should happen when an entity is being viewed at its own page.

Page manager can do the same.

Field access

```
$form['#access'] = custom_access_callback();
```

Menu access

```
$item['access callback'] = 'custom_access_callback',
```

CORRECT USE OF API

Form API

Validation
Form state
Drupal_valid_token

DB API

db_select, db_insert, placeholders
$query->addTag('node_access');

Filter

check_url, check_plain, check_markup, filter_xss, ...
t(), l(), drupal_set_title(), ...

THEMES

THEMES

Themer not responsible

Preprocess functions

CONFIGURATION

PERMISSIONS

Permission management

If Joe from advertising can give the full html filter format to anonymous user, don't bother to think about security

Split up permissions

The default permissions don't cover every use case

PERMISSIONS

```php
/**
 * Implements hook_permission().
 */
function security_permission() {
  return array(
    'ux view menu pages' =>  array(
      'title' => t('view the menu pages'),
      'description' => t('view the menu pages'),
    ),
    'ux manage user fields' => array(
      'title' => 'ux manage user fields',
    ),
  );
}

/**
 * Implements hook_menu_alter().
 */
function security_menu_alter(&$items) {
  $items['admin/structure/menu']['access arguments'] = array('ux view menu pages');
  $items['admin/config/people']['access arguments'] = array('ux manage user fields');
}
```

FILTER FORMATS

Never use full_html

Use filtered_html instead.

Never use phpfilter

Use a custom module for code
Versioning
Bad performance (eval)

CHECKLIST

CHECKLIST

Never use

full_html
Php filter

Permissions

Trusted users only
Split up permissions

API

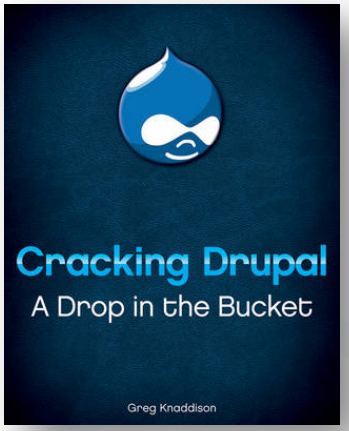
Preprocess functions
check_plain, filter_xss
DB API
Form API
Tokens
Menu/Node Access

GREAT

HOW ABOUT DRUPAL 8?

FURTHER READING

FURTHER READING



Books

Cracking Drupal !!
Pro Drupal Development

Online

https://drupal.org/writing-secure-code
https://drupal.org/node/360052
http://munich2012.drupal.org/program/sessions/think-hacker-secure-drupal-code.html
http://drupalscout.com/knowledge-base

Video

How to avoid All your base are belong to us (drupalcon Denver)