Greg Knaddison - Security on Drupal

http://cardcorp.github.io/drupal-security-intro

http://groups.drupal.org/security

Backups - typically are leaked through sharing, for development etc., so sanitize by anonymizing sensitive data
- sanitizing backups (are there common scripts out there for that? > use drush sql-sanitize commands)
are there common scripts out there for that? (See link from the slides on sanitizing backups - goes to Acquia article)
- drush sql-sanitize has some hooks in contributed modules and people can/should add more for other contributed modules

Roles and permissions
Periodically audit permissions and roles eg. use masquerade to check a role's abilities directly
Permissions - any automated testing for that?
Text formats -
PHP module - use it for development and migrate that code into a module, disable on production sites

Security modules

Paranoia - block PHP from being run (eg. disables "use PHP for block visibility"); prevents itself being disabled
Security Review - review text formats, automated tests for red flags (eg. XSS), has a drush interface (can send email alerts)
Permissions Lock - disables the permissions page (requires permissions to be moved through code, so changes must be intentional and less error-prone in the UI, blocks potential UI access to intruders)
Two Factor Authentication - used by Acquia, adds another layer of auth. pluggable, extensible
Hacked! - ensures that no .php files have been modified in your Drupal site (ie. no core hacks) - can also use an automated script with git status
Pasword policy -
Password Strength2 (zxcvbn-based) - work on the entropy of the 'word'(dictionaries words or repeating characters)

Greg recommends:
- Password Strength2 and Two Factor Authentication
- Paranoia and Security Review on EVERY single site
- Keep code updates up to date

Greg Knaddison "these modules make it harder to make security mistakes"

Hosting - platforms provide good security by default
- Pantheon
- Acquia Cloud - has compliance with ?? standard
see http://drupal.org/hosts

"You have to budget for security"

Drupal Security Team
looking for new members - don't need to be an expert, but if you want to learn more about security and are detail-oriented

MOST COMMON SECURITY ISSUES

1. XSS (cross-site scripting)
2. Access bypass
3. CSRF (Cross Site Request Forgery)

1. XSS
- tricks browser into executing external commands (eg. a malicious application of AJAX)
- usually exploited via Javascript, also Java applet, PDF etc.

TESTING FOR XSS
These catch 90%
1. <script>alert('title');</script>
2. <img src="a" onerror="alert('title');">
more examples - html5sec.org - http://html5sec.org/
nb. be aware of browser specific quirks

FIXING XSS
filter text on output to browser as late as possible
eg drupal_set_title() or t() function when using the @ or % placeholders
-depends on type of data:
-- URL > check_url()
-- Plain text (ie. no markup) > check_plain()
-- Rich text > check_markup() - [input field with a text format, can specify the format to filter against]
-- HTML > filter_xss()

Examples of XSS
1. Alert
- Anon user posting a comment has permission to post Full HTML format
- Post comment with title field containing a <script> tag with Javascript
- JS then executes on page load

2. see greggles' Gist on Github https://gist.github.com/greggles/1368537

2. Access Bypass
- where user can access something they don't have permissions for
- enforced in many different systems (eg. field, entity etc.), so hard to get right

TESTING ACCESS BYPASS
- manually check URLs as an anonymous user, make sure they're secured (eg. node/NID)
- check out Behat (automated behavior driven development/testing) http://groups.drupal.org/behat

FIXES for ACCESS BYPASS
- proper access checks in custom code

- hash in URL is an effective strategy generally to protect URLs without requiring authentication, there is a module that implements this well (name?)

3. CSRF

FINDING CSRF

1. Look for: use of $_GET, $_POST with no drupal_get_token
2. Look for: URL arguments that execute actions without permission validation

FIXING CSRF
In general - make sure the user's intent is confirmed or validated
1. Use FAPI (form token) sending and validating tokens
    drupal_get_token - to add a token
    drupal_validate_token
2. Tutorials at http://drupalscout.com/tags/csrf

CSRF Examples
1. <img src="/user/logout"> entered in a comment by anon user, then an admin who goes to edit the comment executes the logout request on page load, and are logged out
2. Unpublish all content etc.

DARWIN AWARDS for DRUPAL Security
Google search for: "you may enter PHP code" -
https://www.google.com/search?q=%E2%80%9Cyou+may+enter+PHP+code%E2%80%9D&oq=%E2%80%9Cyou+may+enter+PHP+code%E2%80%9D

Drupal 6
- use ?? module to implement stronger D7 password hashing
- backport for login flood control

Drupal 7
- stronger password hashing/salt
- login flood control
- protected cron
- update manager module ?
-- if it makes module update process easier, then good
-- **but** UI access to install any tarball as active module code
-- > usually, disable by default

Drupal 8
- TWIG template engine automatically sanitizes strings in output
- No PHP in TWIG templates - prevents PHP mistakes by frontend themers, or less experienced developers
- WYSIWYG in core
    - streamlined filter
    - in sync
    - local image filter
- Removed PHP module

- built-in CSRF tokens via l() function

RESOURCES
- drupalsecurityreport.org - whitepaper on drupal security
- definitive guide to drupal 7

Security of Drupal vs other CMSes
- open source is now the default solution (CIO), proprietary must be justified
- size of community may affect how many security issues are discovered, so low number of reported issues (eg. Plone) may not be an indicator of good security

QUESTIONS TO ASK GREG

1. [**answered**] Automated testing of permissions?
2. [**answered**] Small number of users logging in
    one case: two Drupal sites used with one database. one site is for editing content, another used for viewing with two different database users
    another case: Paranoia and Security Review on EVERY single site

3. [**answered]**Transitioning between major Drupal versions, major investment so it's a key blocker to keeping codebase secure - will there be a Drupal 6 LTS?

4. Is blocking user/1 recommended?
    - In theory, user 1 is no more powerful than another random user on your site who has advanced permissions (e.g. "administer users" permission). In practice, I do like blocking uid 1 in production because it is such a focus point for someone who is attacking your site.

5. Spam solutions
    - There are a few I like: Honeypot, Mollom

6. Upstream DoS / attack filtering - eg. Cloud Flare - is it effective or recommended?
    - I don't personally have much experience with DDoS, but people I trust have recommended using mod_security (which requires a lot of time to make it work well, but it is open source) or CloudFlare or http://www.dosarrest.com/

7. [**answered]** How does Drupal stack up against other CMS frameworks in terms of security?
    -

8. Get links for: slides + Gist javascript for XSS
slides: http://cardcorp.github.io/drupal-security-intro/
gist: https://gist.github.com/greggles/1368537

9. [Are md5 hashes (or some other random string) in URLs effective against Access Bypass?] yes, generally
I couldn't find the code for this, but it was from Heine Deelstra and might be on familiedeelstra.com.

10. Is there a way to scan an existing site for PHP in the database?
There's some discussion of this topic at https://www.drupal.org/node/1203886#comment-8935001

WORKING GROUP IDEAS

1. Dealing with PHP module.
    use cases:
        1. Views PHP
2. Work through CSRF examples
3. Review xss checking with t() function and @/% placeholders
4. Spin up on Behat - there's some slides at www.slideshare.net/gregknaddison/only-test-the-features-you-want-to-keep if you want a little more on Behat and you like my (Greg's) sense of humor ;)