Rescue Me

Recovering a sad, broken Drupal

Matt Corks

http://acdrupal.evergreen.edu/_endoflife/
End Goal: A Known Good State

Drupal Problems I Have Known and Loathed

- Severe security incursion (eg Drupageddon)
- Rescue from an inexperienced prior vendor and/or multiple levels of outsourcing ($n > 3$)
Intrusion Recovery

https://www.flickr.com/photos/remydwd/312615249/
Why Me?

First, understand that it’s very rarely personal or directed

- Motivation for almost all attacks is financial - follow the money
- It’s almost always script kiddies copy/pasting exploit code from forums
- Primary Goal: Exploit your Google page rank to insert links to phishing sites
- Secondary Goal: Make money directly from your site (steal credit card numbers, email addresses, or other personal data; send spam email; or even mine bitcoins)

Very occasionally: goal is to deface homepage to gain bragging rights
How did they get in?

Almost always due to a publicly disclosed vulnerability in core, a contrib module, or a library.

The security team only supports stable contrib module releases; **not** for libraries and for dev, beta, & RC versions of modules without the shield icon.

Subscribe to the security announcement list and be aware of release windows.

Monitor security announcements for external libraries such as phpCAS.

Make sure OpenSSL, Apache/nginx, kernel, etc are up to date.
What did they do?

Take site offline as soon as you discover the problem; backup site and logs

Determine timestamp of attack & inspect all files changed since then

Inspect nodes, users, comments created or modified since attack

Review dblog & syslog

Note: Attacker could have changed any of these!

Use `file` to check file directory; search for PHP or JS in entire database
How do I recover user trust?

Regain user trust by informing them of disclosure of personal data.

Reset all passwords if accounts were compromised (also clear session & cache tables - after making a forensic copy of the menu_router table).

Mention time & date of infiltration and mitigation.

Disclose your PCI or HIPAA compliance issue as required.
How do I recover the database?

With backups: revert to last known good state, and manually inspect & re-insert valid new content created since then.

Without backups: remove spam, PHP, and JS from database; ensure PHP filter is still disabled; then start taking backups!

Refactor your site to use a payment gateway which doesn’t require you to handle credit card numbers (Paypal, Stripe, etc).
How do I recover user files?

Restore to last known good state, then bring forward missing resources after they pass inspection.

Without backups: use timestamp and *file* check to remove any spam or viruses from files directory, then start taking backups!

Don’t forget hidden files & directories beginning with . such as the classic . . .
How do I recover the codebase?

Revert to last known good state, apply all pending security patches

Without backups:

- Download latest stable version of Drupal core plus contrib & libraries
- Manually inspect custom modules, features, themes, then copy over
- Check for features overridden in the database:
  - for i in $(drush fl | egrep -i "over|need" | sed -e "s/ *Enabled.*//* s/.* //") ; do echo $i; drush fd $i > $i.diff; done
- The hacked module isn't enough: it doesn't report new files
How do I recover the server?

Where feasible, rebuild the server, virtual machine, or container

Otherwise, be aware of what the PHP process could have written and where; this could include backdoor for shell access (making recovery all but impossible)

The codebase itself should be read-only to PHP processes, which only need to write in the Drupal files directories (PHP files could be owned by a different user, or the files directories could be a different mount point)

If running a shared hosting environment, run PHP as a different user per client
How do I better prepare for this?

Apply security updates!

Have regularly-scheduled rolling backups (nightly, weekly, monthly) covering both the database and user-contributed files

Put your codebase into git or an equivalent system

Have a plan to recover anything else in any .gitignore file (eg. auto-generated passwords)

Use virtual machines or containers where possible
Other Resources

Security Team docs: Your Drupal site got hacked. Now what?

Security Review (D7 & dev release for D8)

Site Audit (D7 & D8, also includes security checks; this patch would add Security Review support)

Drupalgeddon (for an example of a process and a list of further resources)
Previous Vendor Recovery
What did they do in this patch?

Run **hacked** module; review and document patches to core or contrib

- Keep a record of all patches, either via composer or in a patches/ directory
- Rename them if they don't already include the module name & drupal.org issue number
- If they're legitimate but not already upstream, post them
What did they do in this custom code?

Read **all** their custom modules and themes

- Often you can just delete their code & replace it with an existing contrib module or API function
- Check the `.module` files for all features
- Organize modules into separate contrib, custom, and features directories
Typical mistakes: Reinventing the API

Often too much logic in the theme/template layer

Check for code inserted via `php`, `views_php`, or `computed_field` modules

- This should all be moved to functions in the filesystem (eg `computed_field_FIELD_NAME_compute()`) so that you can have revisioning, use the opcode cache, and get useful warning & error logs

Control block visibility via block hooks; for panel visibility call a function defined in a custom module
Drupal architecture disasters I have known

Custom patches to core or contrib not posted upstream, or only to vendor's github repo instead of drupal.org

Hacked core to enable inheritance of multiple install profiles in order to load modules from any of them as well as from sites/all or sites/domain.tld

Deployed the patches to core shipped with the uuid_features module for staging content instead of using deploy or migrate
Drupal architecture disasters I have known

All custom logic implemented in `views_php`, theme functions and/or templates

General-purpose `views`-like query builder and/or `t()` implemented in raw SQL queries in the theme layer

References in code to hardcoded node, user, or term id's

References to hardcoded titles or other user-generated strings

**Note:** The above problems typically imply that Drupal core & contrib will be badly out of date (see intrusion recovery, above)
Stakeholder expectation management

Explain that everything was in fact not working fine last week

Bid on an audit and complete that before committing to any more work

Be aware that the client is bad at working with and reviewing deliverables from a vendor, so budget for extra communication & coordination time

If you can't convince them how broken the site is, don't take the contract

Be very cautious if the client hasn't already figured out for themselves that they have a problem
Questions?

Also accepting:
- Advice
- Suggestions
- Anecdotes
- Cautionary tales
- Condolences

Matt Corks
matt@evolvingweb.ca
https://www.drupal.org/u/mvc