Drupal Audit Tasks (to be run for each site)

At each stage, the first step should be to create a forensic copy of code, database and files. This should be stored on **read-only media so it cannot be modified** (eg CD, DVD, read only file system). This is to be kept by Client.

Task	Responsible
Database Tasks	
### Make a forensic copy of the database	Client
### Truncate all sessions `TRUNCATE sessions;`	Client
### Check User Table Ensure that the list of users & email addresses make sense. Any incorrect or questionable account should either be deleted or set to disabled. `SELECT name, mail, init, status FROM users;` As you look at all the users on the site, be sure to look not only at their username, but also their email address. Does it make sense for that person? If you're not sure, block the account and double-check with the person. One trick you can use to see if the email has changed since the user registered is to look in the database and compare users.init with users.mail. **While you're at it:** audit the users and remove advanced roles from people who might not need them. Block users who are no longer involved in administering the site.	Client
### Update User passwords `update users set pass = concat('ZZZ', sha(concat(pass, md5(rand()))));`	Client

Alternatively, the Force Password Change module may be utilised to alter all user passwords.	

### Check User Roles 	
 	
Look at all the roles on your site. What permissions do they have? Which have advanced permissions that could be used to take over a site? Perhaps there is a new role you know you didn't make. Is there one user with that role? It's probably got an email and/or password controlled by the attacker. With your list of roles that have advanced permissions, take a look at the users on your site. Do you know every single one that has advanced permissions?
 	
Ensure that any roles in the database match what are expected and that none have been added.
 	
`SELECT name FROM role;`
 	
Also check that the users with administrative permissions match those expected.
 	
`SELECT u.uid, r.name, u.name`
`FROM users_roles ur`
`JOIN users u ON ur.uid=u.uid`
`JOIN role r ON ur.rid=r.rid`
`WHERE ur.rid IN (SELECT rid FROM role_permission WHERE permission IN ('administer users', 'administer permissions', 'administer content', 'administer content types'));`
	Client

### Truncate cache tables All cache tables can be truncated prior to export to both reduce the size of export as well as remove anything added outside of Drupal. Run the following for each cache table `TRUNCATE cache_<table name>;`	Client

Provide Acquia a DB Snapshot **Special note:** While you should look in the database to help make decisions, if your site has been well and truly compromised remember that the attacker may have modified any row or column in the database. Use known-good backup copies of the database (if there are any) to validate theories.	Client
Compare known good copies of the database against current exports Provided there is a recent known good copy of the database, certain tables can be compared to determine if unusual changes have been made.	Acquia
Check for menu router table Custom page callbacks can be created in the menu router table which allow execution of malicious files and may allow files to be added to the codebase. The menu router table will be checked to ensure these are not present.	Acquia
File Tasks	
Make a forensic copy of the filesystem	Client
Drupal files directory export Review the files in the "files" directory to ensure they are all appropriate. It may be helpful to review the combined metadata of owner, group, permissions and timestamps as a fingerprint of the files on the server. If most of the files have one fingerprint and a single other file has a different fingerprint (e.g. edited about when the attack started) that can help you understand what happened. Ensure copy of the files directories is created using the `'-p'` option in the `'cp'` command. This ensures that file permissions, timestamps and ownership remain intact and will aid in auditing.	Client

Antivirus scan of files directory Once the files directory and sub-directories have been copied, perform an antivirus scan on them.	Client
Document all non-Drupal pages in the codebase Any custom html/php pages should be documented to exclude them form suspicion. Include any PHP libraries, or external tools. Anything that is not Standard Drupal. It is important also to be aware of a known good state of these files to ensure that additions haven't been made to them.	Client
Ensure PHP files are not executable from the file system An .htaccess check will be added to ensure no PHP files are executable from public / private / temporary file systems.	Acquia
Scan file system for PHP/executable files	Acquia
Code Tasks	
Provide Acquia Drupal codebase snapshot: core, themes, modules, etc **Preferred option:** If using version control system (eg git, svn), a cloned version of it for each site or access to the code repository so that Acquia can check it out. **Secondary option:** Provide a copy of the codebase created using the `'-p'` option in the `'cp'` command. This ensures that file permissions, timestamps and ownership remain intact and will aid in auditing.	Client
Review all Drupal core/contrib code files Common hacks modify known files in the codebase. These will all be checked against published versions of Drupal core and contrib to ensure that nothing has been added which presents a vulnerability. Modifications to index.php or any code file in the site such as a template file are possibly vectors. The methods could include:	Acquia

• A virus on the computer used to administer the site which uses stored credentials in a FTP tool to edit and upload the files. • Arbitrary code execution on the server and loose server file permissions used to edit or overwrite a file. • Arbitrary file upload which was used to upload a command shell which was then used to modify the code. All code files will be compared to known good copies, either in the revision control system or from drupal.org Will use hacked! module, Security Review module and Drupalgeddon command	
Provide Acquia with patches applied to core/contrib modules If available. Any patches applied to core/contrib modules will be flagged as an alteration of the module. Patches applied should be made available or documented to ensure that false positives are flagged correctly.	Client
Provide Acquia a copy of the codebase prior to Oct 15th If available. This is to ensure that any custom modules have a known 'good' state. This is before the announcement of the Drupal SQL injection attack.	Client
Locate code files not associated with Drupal core/contrib Look for files on the server that are NOT part of the Drupal codebase, e.g. modules/system/qseboj.php	Acquia
Run standard Acquia developed automated tools A number of toolkits have been developed for automated testing of codebases. These will all be run against the sites as they are imported to ensure unexpected issues may be presented to Client and resolved.	Acquia

Provide Acquia with information on modules enabled Some modules may be present which increase the attack surface and make site content a possible area of exploit. PHP filter, Views PHP, Context PHP or any other module which allows site users to enter raw PHP is a potential target and Acquia should be made aware of where these are used. The PHP filter module in core is really handy for site admins to put a little snippet of code into the Drupal interface to make a tweak to the behavior of a node or block. It's also a great way for an attacker to run whatever code they want to.	Client
Check for PHP code inside your database With the above information about PHP modules in use, Acquia will be able to examine content added using these against known good copies of the database to ensure nothing new has been added and vulnerabilities in existing PHP if applicable. Examine blocks and views for PHP.	Acquia
Review the text filters in place Text filters should be reviewed to ensure that no changes have been made to the content filters. If a new filter has been added or existing filters changed then these will need to be reverted. Additionally, if unsafe filters are in place, content using them should be checked to ensure that unsafe tags are not in place.	Acquia
Post Auditing Tasks	
Review Server logs **Note**: this can be performed at any stage after Acquia has received all files and materials. Examine server logs for all dates subsequent to the release of the vulnerability. Check for unusual requests which may include SQL Injection code or non-403 page responses on admin paths from non-internal Client IP addresses or outside of regular work hours.	Client

Knowing the time and IP of your attacker may help you tie together more clues about the attacker from looking in the database at the watchdog.hostname or comment.hostname Useful logs to examine are Drupal watchdog log, Webserver (Apache/NGINX) logs, Load balancer logs (if applicable), server system logs.	

Some tasks based on information from:

- https://github.com/greggles/cracking-drupal/blob/master/after-an-exploit.md
- http://drupal.geek.nz/blog/your-drupal-website-has-backdoor