Drupal Security Best Practices

By Mike Gifford,
OpenConcept Consulting Inc.

Drupal Security Best Practices

A General Guide to Making your Drupal Site More Secure

By: OpenConcept Consulting Inc.,
originally written for Public Safety Canada

Main Author: Mike Gifford

Contributors

Mike Mallett (OpenConcept)
Xavier Landreville (OpenConcept)
Matt Parker
Michael Richardson
Colan Schwartz
Mack Hardy
Peter Cruickshank
David Norman
Lee Rowlands
David Timothy Strauss
Ben Hosmer
Ursula Pieper
Jonathan Marcil
Peter Wolanin

Editors

- Lee Hunter
- Xavier Landreville
- Mike Mallett

Version

1.3.0 (December 08, 2015)
This is a *living document*, sign up for updates.

Like this guide?

There are lots of ways to contribute to its further development. This is a community effort, but even if you aren't a security guru there are plenty of ways to help. If you have

editing skills, know that this guide could use your help.

We need people to spread the word about what you've found in this guide. It's available under an open Creative Commons license so please share it with people who you think would benefit. If this guide has been useful to you, please spread the word on your favourite social media platform.

Contents:

- Foreword
- Copyright
- About the Author
  - Mike Gifford, Principal Author
- A) Introduction
- B) Principles of Security
- C) Security Concerns for Managers
- D) Server Security
  - 1) Server Procurement
  - 2) Immediately After Receiving Root Access
  - 3) Create a baseline
  - 4) Limit Access from Outside
  - 5) Initial Installs
  - 6) Server Maintenance
  - 7) Managing Server Logs
  - 8) Rough Server Ecosystem Image
- E) Web Servers
  - 1) Restricting Access
  - 2) Removing Code
  - 3) HTTP Headers
  - 4) HTTP Basic Authentication
  - 5) Web Application Firewall
  - 6) Managing the .htaccess file
  - 7) MIME Problems
  - 8) Everything Else
- F) PHP
  - Setting ini Variables
- G) Database Layer
  - Database Access
  - PHPMyAdmin

- H) Drupal
  - 1) Files
  - 2) Drush
  - 3) Errors
  - 4) Core and Contrib Hacks
  - 5) Administration
  - 6) Modules to Consider
  - 7) Modules to Avoid on Shared Servers
  - 8) Drupal Distributions
  - 9) Choosing Modules & Themes
  - 10) Drupal Updates
  - 11) The settings.php
  - 12) Advantages of Drupal 8
  - 13) If You Find a Security Problem
  - 14) Miscellaneous
- I) Writing Secure Code
- J) Development, Staging and Production
- K) Regular Maintenance
- L) Points of Debate
  - Security by Obscurity
- M) Crackers
  - Pharma Attacks
  - Illegal Botnets
  - Private Information
  - Information Disclosure
- N) Regulations
  - Global
  - USA
  - Canada
  - Europe
- Additional Resources
  - General Guidelines
  - Videos
  - Third-party Tools
  - Books

Foreword

This document describes best practices for setting up and maintaining a Drupal site. It was initially written for the Government of Canada, but it is equally applicable to other organizations.

Drupal is a popular, open source Content Management System (CMS). It has a strong security model, but like any application, requires adherence to best practices. Furthermore, Drupal is only one piece of the software that is required to run your site, and one needs to consider the security of the entire set of software.

This is not a comprehensive document, as IT security is a complex field. We have tried to focus on fundamental principles that can improve security. For more information on web server security, see the reference links.

We do not believe that there will ever be a 100% secure system. There are always bugs in software and new exploits are being attempted all of the time. We are listing options to consider, but each organization will need to weigh which combination they are going to use.

Because there will always be new security exploits available, it is in your best interest to sign up to receive updates of this guide.

Copyright

About the Author

Mike Gifford, Principal Author

Mike Gifford is the founder and president of OpenConcept Consulting Inc., based in Ottawa, Canada.

Long a prominent contributor to the Drupal community, Mike has been a member of the international group of developers that contribute to Drupal Core. Mike is currently a Drupal 8 Core Accessibility Maintainer.

Led by Mike, OpenConcept contributed the first Drupal theme for the Government of Canada and has since been actively involved in the Drupal Web Experience Toolkit distribution within the Government of Canada.

To contact Mike:

email	mike@openconcept.ca
follow	@mgifford
call	1-613-686-6736
browse	https://openconcept.ca

A) Introduction

Drupal is a leading Content Management System (CMS) in many institutions around the world.

Governments are increasingly a target for cyber attacks, and yet the security culture is slow to change. Keeping up-to-date on best practices is critical to protect personal information and government assets.

This guide provides an overview of the principles, best practices and next steps in securing your Drupal site. We have provided, where possible, some detailed instructions. Managers should read sections *B) Principles of Security* and *C) Security Concerns for Managers*. System administrators will need to focus on sections *D) Server Security*, *E) Web Servers*, *F) PHP*, *G) Database Layer*, *I) Writing Secure Code*, *J) Development, Staging and Production*, and *K) Regular Maintenance*. Drupal developers can focus on section *H) Drupal* and *I) Writing Secure Code*, but should be familiar with the impact of the other sections too.

It should be clear that not all of the steps outlined here will need to be taken on all Drupal sites. The principles should be followed but not all of the security suggestions described will need to be followed by all organizations. Each practice or tool should be carefully evaluated to understand the potential costs, risks and benefits.

This document raises issues to consider before you procure a server and especially when you first gain access to your server. It provides suggestions on what additional software you can add to your site which can help improve its security. It also highlights configuration options that you can apply to Apache, PHP and MySQL to improve on the default settings. Finally we talk about things that you can do to enhance Drupal security.

The code snippets which are included are not always a comprehensive guide, but there are always links in the descriptive paragraph with more information which you should consult before installing programs on your production server. It is always worth consulting the community documentation on Drupal.org.

Because this document strongly recommends against the use of Microsoft Windows servers for Drupal sites, Windows security will not be addressed.

Security cannot be just a buzzword, it is a process that needs to be ingrained in the culture of an organization. There needs to be a clear understanding about lines of

responsibility and ultimately management needs to provide the budget required to ensure that systems can be maintained and regularly re-evaluated.

Eternal vigilance is important as those searching for your vulnerabilities are working around the clock and are well-financed. This document will, itself, need to evolve to keep pace with new vulnerabilities.

B) Principles of Security

There is safety in the herd:

Created by Steffen Nørgaard Andersen from the Noun Project

Leverage large, well maintained open-source libraries (packages) with a critical mass of users and developers.

Use compiled packages and check data integrity of downloaded code.

Start with a popular and well maintained Linux distribution (Debian/Ubuntu or Red Hat/CentOS).

Order matters:

Don't open up services to the Internet before your server is properly secured.

Limit exposure:

Only install and maintain what is necessary.

Reduce the amount of code installed.

Review server configuration regularly to see if it can be streamlined.

Deny access by default:

Only allow access where it is needed, and make all access policies deny by default.

Use well known security tools:

There are well supported libraries that limit exposure, and check for intrusion.

Many suggestions are provided in the *D) Server Security*, *F) PHP* & *H) Drupal* sections.

Avoid writing custom code:

Even large organizations find it difficult to invest properly in regular, ongoing code reviews.

Minimize the use of any custom code.

Contribute back: No software is ever perfect.

There is always room for improvement.

Make the code you use better and give it back to the community.

If you do it properly you won't have to rewrite your code with the next security release and you will get free peer review and ongoing maintenance.

Limit access:

There needs to be clear, documented roles of who has access to what.

Only use root access when required and do so through sudo so people are not actually logging in as root.

Isolate distinct roles where possible. Each person with access requires a separate account as shared accounts are inherently insecure.

Make your application happy:

When running smoothly your server should not be generating errors.

Monitor your server then investigate and resolve errors.

Document everything:

Make sure you have an overview of any customizations which may have been done or any additional software that may have been added.

Limit use of passwords:

Have sane organizational policies on password requirements.

Keep track of your passwords in controlled, encrypted programs.

Where possible use password-less approaches such as SSH key pairs which are more secure.

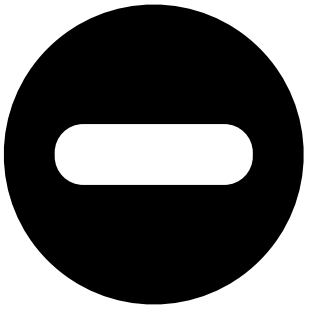
Don't trust your backup:

Define and review backup procedures and regularly test that you can restore your site.

Obscurity isn't security:

Organizations need to have their security policies well documented and internally transparent.

Section Security by Obscurity discusses this issue in detail.



Created by Mateo Zlatar from the Noun Project

Security is big:

It is a mistake to assume that one person can do it well in isolation.

Having access to a team (even outside of the organization) will help.

Remember, you're still not safe:

Have an audit trail stored on another system.

If your site is compromised, take the time to find out how.

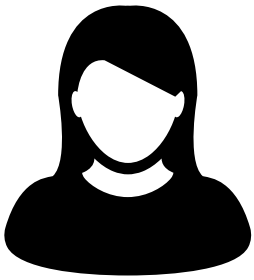
Use proper version control for all code and configuration.

Not just for techs:

Upper management needs to take the time to understand these general principles of IT security as they have profound implications for the whole organization.

C) Security Concerns for Managers

There are many assumptions about IT security that need to be fundamentally rethought in the era of the Internet. Organizations are struggling to come to terms with this at the same time as working to understand the implication of cloud-based services. What we can be certain of is that this field is accelerating and organizations need to keep up.



Created by Wilson Joseph from the Noun Project

The first principle is to understand that time corrodes security and on the Internet time moves very fast. You can't assume that any service you buy or develop is currently secure or will remain that way for long. It is critical to understand what investments have been made and how they are maintained.

Web hosting and application development are different fields and one cannot simply outsource security upgrades to someone else to do. No web hosting company can "take care" of your server security in isolation of the application that is running on it. Ultimately, someone familiar with your website and its content needs to be involved in performing upgrades.

Third-party agencies are ultimately going to be involved in supporting your site. The protocol for communicating with these agencies needs to be clear and well documented. This recently happened with a large Canadian municipal website that was redirected to an image of a dancing banana using an approach known as social engineering. By leveraging human vulnerability, crackers were able to gain control of critical infrastructure. Properly documented procedures are important, as third-party services can often be manipulated by fraudulent email or telephone requests.

It is also important to remember that one person working in isolation cannot be expected to be an expert in all aspects of Internet security. This is a vast area of expertise and it is changing quickly. It's important that your security person has ongoing training and is engaged with both the Drupal and wider security communities to keep up with the latest threats, vulnerabilities and mitigation strategies.

Schedule time for a skilled security expert outside the core team to double check the server/Drupal configuration every quarter. This does not have to be a consultant, but it should be someone outside of the website development team.

Everyone wants security to be simple, it isn't. It's a matter of determining, as an

organization, how much risk you want to be exposed to. You can invest as much or as little on security as you want, but the risks are generally inversely proportional to resources spent on tightening your system. Security has costs as well as benefits. Complex systems are usually less secure because it costs relatively so much more to secure them.

Many organizations have policies for Threat and Risk Assessments. However, as we've seen with the implementation of Healthcare.gov political pressures associated with large projects often push security concerns into a post-launch phase. It is highly recommended to go through a Application Threat Modeling process. Threat Risk Modeling is also a recommended process to help expand understanding of potential threats by using processes like STRIDE or DREAD. By identifying and classifying an organization's assets one can begin to prioritize where to focus resources.

Thinking through attack vectors and limiting exposure is really important. Many of the sites that were compromised by the Shellshock bash bug in that hit in September of 2014 simply hadn't disabled services that they weren't using. Bash is a command processor that runs in a terminal window, so for most Drupal sites there simply isn't a need to expose it to anyone other than properly authenticated Linux users.

As with most work, a great deal of security work lies in identifying and eliminating assumptions. Document what is done, and be transparent in your work so that your organization knows that it has the level of risk it wants to maintain.

Organizations should also consider if the software that they use is properly resourced. The Internet is built on free software, but much of it is backed by corporations who are also providing services built on the expertise that they have built by contributing to open-source software. The Heartbleed bug cost the economy billions, but was largely caused because the OpenSSL library was under-resourced. Although this is just one example, consider donating to project like the OpenSSL Software Foundation which supports the security infrastructure your organization depends on. Likewise consider supporting organizations who contribute to Drupal's security team.

Much security work should begin before anything is installed. Properly considering security before beginning a server implementation is important. Addressing security issues later in a project makes it impossible to do a security evaluation of the base system. When setup is rushed, bad practices are often used which then become patterns that are followed long after the site is launched.

Implement and enforce a policy of using very complex passwords and two-factor authentication for any critical service, like email. Proper use of a secure, redundant

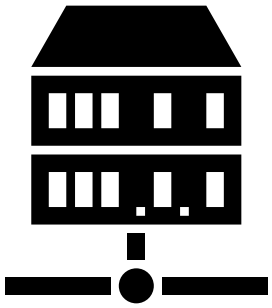
password manager is also something that should be key for all employees. If someone is able to hack into your Google Mail or GitHub account, they can often access much more than your communications. Most services on the Internet are keyed to email addresses and if access to that is compromised a user can get access to stored passwords or the ability to reset passwords. Identity theft online is a huge problem for institutions.

The UK's Government Service Design Manual is an excellent resource for any large institution and it has a great section that applies directly to web security, Security as enabler: Using technological change to build secure services. Of particular interest is the point that security shouldn't degrade user experience.

Don't ignore minor bugs. As Darren Mothersele mentions in his blog, it is possible for a number of minor vulnerabilities to be chained together in a way which can become a major exploit. Sites as large as GitHub have been successfully targeted this way. As he says, The cost of (in)security is high and "investment in security review and penetration testing is a Good Thing".

IBM suggests that organizations should spend between 6 and 14% of their IT budget on security. The less organizations spend on security, the greater the likelihood that there will be security breaches. How much of this should be spent on web security will depend on the relative number of assets and vulnerabilities which are contained in the website. Only with a good picture of your overall risk can you determine where it is best to invest in skills, processes and technologies to mitigate it. Think about what percentage of your budget you presently spend on web security, and if that is sufficient to cover the risks to what has become mission critical for most organizations.

D) Server Security



Created by herrreiprich from the Noun Project

Any website is a complex ecosystem of software. Each aspect can be tightened through proper configuration and through the addition of components beyond what is found in a default installation. This document provides some examples, but mostly relies on links so that you can read the specific details on how this should be done. There are other lists of considerations for server security, like Robert Hansen's list of 10 major tenets of a secure hosting model, but where possible we will be referring back to Section *B) Principles of Security*.

1) Server Procurement

Start server documentation with information from your server contract. There are often technical details and notes about who to contact when things go wrong.

It is important to determine that there is a strong security community behind the Operating System (OS) distribution you choose, and that you have the necessary human resources in your department to maintain it. Both Debian/Ubuntu and Red Hat Enterprise Linux (RHEL)/CentOS/Fedora can be considered solid. The advantage of a Debian or Red Hat based solution is that there is extensive documentation and large communities of users who've shared their experiences through forums, issue trackers, and blog posts. Ubuntu is based on Debian. CentOS is almost a copy of RHEL and Fedora is the community edition of RHEL. The Fedora edition often releases updates before RHEL which can be an advantage for some security fixes. Most references to one of these two groups of distributions should be interchangeable. Note that Ubuntu and Debian have a different development cycle and are not identical.

If you use a Red Hat Enterprise Linux (RHEL) system, you will need to subscribe to their service in order to apply security upgrades and install the additional packages mentioned in this document. Before procuring a Red Hat server, check that your package includes a subscription.

In our opinion, distributions of Linux like SUSE simply do not have a critical mass of users and developers (in the web server space) to maintain the code and documentation required for a secure environment. Microsoft Windows is not a standard platform for hosting Drupal and is generally not recommended since community support is much less robust. Furthermore, it is very difficult to limit exposure on a Windows

Server since there are many unneeded pieces of the operating system which you cannot easily uninstall.

If you are worried about the server's physical security, you can also set up an encrypted partition on your hard drive. If you do this, be aware that it may introduce performance issues which might cause problems for your server. This document will not be covering how to set up an encrypted drive but depending on the perceived threats, it may be worth implementing.

Special consideration should be taken when enabling HTTPS for encrypted traffic on "shared host"-type environments (any server hosting more than 1 domain). Typically, due to the nature of the protocol, only one HTTPS website (domain name) could be hosted per IP. However, with Server Name Indication (SNI), it is now possible to host multiple HTTPS domains with distinct TLS certificates on the same IP. It must be noted that SNI is dependent on both the client and the server supporting the TLS extension, but most do nowadays. Another option, which might be useful if your servers or clients do not support SNI, is using Subject Alternative Name on certificates (also known as Unified Communications Certificates). These certificates contain extra fields that list other common names (domain names) for which that certificate is valid.

Finally, don't get a server that comes with a server admin control panel. It promises to make managing your site easier but presents security problems. There are a number of commercial packages, like cPanel or PLESK, that do make it easier to change settings on your server. This seems particularly attractive if less technical users are responsible for server administration. In our recent experience with cPanel, it introduced many difficulties in applying many of the suggestions and recommendations described here. If you choose a server with one, you will need to experiment with which of the following suggestions you are able to implement. Some control panels are also known to overwrite settings when manual changes are made to configuration files. It is important to work to minimize the attack surface and since these dashboards are managed through the web, it is yet another point where your server can be compromised. Ultimately a control panel could prove convenient both for an administrator and for someone looking to hack into your system.

2) Immediately After Receiving Root Access

Hopefully the root password wasn't sent via an unencrypted email with the other login credentials. Very few people use GPG to encrypt emails because it is cumbersome, but confidential documents should be encoded/decoded with this type of protection. You can request that the password not be sent using the same medium so it will be difficult to intercept. At the very minimum, try to ensure passwords are sent separately from the

rest of the authentication data. This provides a slightly more obscure means to stop this information from being intercepted. Ideally credentials would be sent with a tool similar to NoteShred which can delete a password after it is viewed without leaving an email archive of the information. However it is sent, change your password immediately after receiving it.

Most web hosts send all of the credentials together, therefore, the first step after getting access is to log in and change the root password. Unencrypted email communications offers no security on the Internet and thus you must address this vulnerability immediately.

Update the list of available software and perform system software upgrades. Most web hosts will use a pre-packaged distribution and there will frequently be updates that need to be applied immediately. Make sure you've got the updates and that the new packages are running. If you update the Linux kernel you will have to reboot the server for it to be applied. If you update Apache, you will also need to restart it. Debian-style systems often restart the main daemon instances on package updates automatically, where RHEL-style systems treat daemon restarts as an administrator's responsibility.

```
# Ubuntu/Debian
$ apt-get update && apt-get upgrade

# CentOS
$ yum upgrade
```

You will inevitably have a number of passwords to maintain. We recommend storing these in a new KeePass or KeePassX Password database. It has a nice password generator which makes it very easy to generate long (25+ characters) and complex passwords and store them immediately. If you get any other passwords supplied via email, reset them immediately. Remember that your email address is also a point of vulnerability.

The most common account that crackers try to compromise is the root user, so disable root logins. Furthermore, set up user accounts with sudo access and use SSH keys so that nobody accessing the server is using a password. Note: the commands listed here assume you are using sudo access and but we have chosen not to explicitly prefix them with sudo.

Protect your SSH keys by ensuring that your private keys are password protected and using at least 2048-bits encryption. Some people now think that the default should be

4096-bits, while really paranoid people may choose to use 8192-bits. Although theoretically there is nothing wrong with doing this, in practice there are *limitations* to what browsers and hosting companies provide. At the present time, 2048-bits seems sufficient for most use cases.

By disabling the use of passwords for SSH user logins, a common server vulnerability is simply eliminated. When you turn off password logins, "script kiddies" cannot compromise your server with common dictionary or brute force attacks. There are explanations on how to effectively disable password logins but check that `/etc/ssh/sshd_config` has the text

```
PasswordAuthentication no
```

Remember that when downloading important files there is a possibility that they have been tampered with. Important security documents often come with a MD5 or Secure Hash Algorithm (SHA) code which allows a user to verify that the file on a server is identical to the file that they have downloaded. You can generate a checksum locally to determine equivalence using one of these:

```
$ shasum -a 256 -/DrupalSecurity.epub
$ md5sum -/DrupalSecurity.epub
$ openssl sha1 -/DrupalSecurity.epub
```

3) Create a baseline

Record a baseline of your server that you can review, knowing that this is the minimum number of processes which are running with a clean system. Likewise, record the baseline from a netstat report to see what ports are open:

```
$ ps afx
$ netstat -l -p -n
```

Record the list of installed packages on the server. Save this information in a text file in your management code repository. If your server is compromised it is useful to know what packages were installed and running when you started:

```
# Ubuntu/Debian
$ dpkg -l
```

```
# CentOS
$ yum list installed
```

Manage your ports through firewall settings: It is important to review and document the firewall settings - open ports - to see that they are properly restrictive. The firewall program for sysVinit OS versions (CentOS/RHEL <=6), iptables, is still available for systemd OS versions (CentOS/RHEL >=7), which by default uses firewalld.

Using iptables, the port settings can be listed from the command line with:

```
$ iptables -L -vn
```

You can load/save the iptables easily using the iptables-persistent package (installed on Debian/Ubuntu using `apt-get install iptables-persistent`). With that you can simply save the existing IP tables from the command line:

```
# Ubuntu/Debian
$ service iptables-persistent save

# CentOS
$ service iptables save
```

Install Rootkit Hunter (RKH) to help you “detect known rootkits, malware and signal general bad security practices”. You can set it up to send you email alerts, but can also do manual scans.

```
# Ubuntu/Debian
$ apt-get install rkhunter

# CentOS
$ yum install rkhunter
```

4) Limit Access from Outside

In general you will want to allow traffic for port 22 (for known IPs), 80, 443 and reject other ports. It can also be useful to use firewall rules to restrict outgoing connections from the Apache user. The possible exception to this is drupal.org’s IP address as you will want to regularly use Drush (Drupal’s command line shell and scripting interface) to update modules (see H2 below). You can easily see what ports are open by using a

port scanner such as nmap from an external machine:

```
$ nmap -sS SERVER_ADDRESS
```

We recommend running periodic TCP port scans on your server. MXToolbox offers an option to do this through their site, but you can also use tools like nmap which offer more fine-grained controls.

Many servers come with BIND on UDP port 53. This program can probably be removed in most instances or should be restricted with a firewall if required. There are some detailed instructions here on how to remove it, which are particularly important if you aren't sure if you need it or not. To check if bind is running, run this from the command line:

```
$ ps -Al | grep bind

# sysVinit
$ chkconfig grep bind

# upstart
$ service bind status

# systemd
$ systemctl is-enabled bind
```

You can obscure your SSH port by reassigning it to other than the default (22). This might fool a lazy cracker who isn't using a port scanner first, but won't stop the serious folks.

One of the best ways to limit SSH access to a server is to restrict access to a handful of known subnets (ie. 192.168.1.0/24) where administrators actually work. Don't be afraid to add to this list; make it easy for your people to work wherever they need to. Security is not the enemy.

You can also restrict who can SSH into the server to a limited number of IP addresses. Be very careful when configuring this as you don't want to block yourself from accessing the server.

Debian's admin documentation offers the following changes which can be made to the iptables firewall:

```
# All connections from address 1.2.3.4 to SSH (port 22)
$ iptables -A INPUT -p tcp -m state --state NEW --source 1.2.3.4 --dport
22 -j ACCEPT
# Deny all other SSH connections
$ iptables -A INPUT -p tcp --dport 22 -j DROP
```

There are many ways to do this. Debian uses ufw, the sysVinit releases of RHEL use system-config-firewall-tui, lokkit is coming along nicely and systemd releases RHEL 7 ship with FirewallD by default. Ultimately they all do the same thing slightly differently. Make sure you understand your configurations and review them regularly.

If you already have established a virtual private network (VPN) then you can restrict SSH access to within that private network. This way you need to first log in to the VPN before being able to access the port. Leveraging an existing VPN has some additional costs but also some security advantages. If an organization isn't already using a VPN however, then the usability problems with forcing people to use it may encourage developers to find ways to circumvent it. It is important to remember that a VPN is only as secure as the individual servers on the VPN. If the VPN is shared with systems out of your control, and the responsible Systems Administrators (Sysadmins) might be lax in security, then your servers should be hardened as if on the public network.

5) Initial Installs

There are some tools to harden your Linux system. The program grsecurity addresses a number of memory and permissions issues with the kernel.

BastilleLinux guides the administrator through an interactive process to limit access on the server.

Mandatory Access Controls (MAC) policies can be managed through programs like SELinux and AppArmour, for high security environments. With Ubuntu, use AppArmour as it comes installed by default. While AppArmour is often considered inferior and less flexible than SELinux, there is no need to uninstall it. AppArmour may impact other security tools and should not be used in conjunction with SELinux.

With other distributions it is recommended to use SELinux (examples for SELinux on Debian and SELinux on Red Hat) as its rules were initially developed to meet NSA policies.

```
# Debian (not Ubuntu)
```

```
$ apt-get install perl-tk bastille selinux-basics selinux-policy-default
auditd
```

Using a Host Based Intrusion Detection System (HIDS) such as the OSSEC HIDS is a good practice. You can find more information on the projects, including tutorials and how-tos at OSSEC's documentation. Tripwire and Snort are other IDSes which monitor the integrity of core files and will alert you to suspicious activity (see Tripwire on CentOS and Tripwire on Debian). With any HIDS you should make sure that secure IPs, such as your outgoing gateway, are whitelisted.

Drupal monitoring can be set up to work with OSSEC which would be more efficient than using Drupal's Login Security module (6/7/8) as it would allow you to use your existing HIDS infrastructure to alert you to these sorts of attacks.

Crackers will often try to use a brute force attack to guess usernames and passwords. Using a service like Fail2ban that can block IP addresses that are making an unreasonable number of login attempts. This won't prevent distributed attacks, but could be used in conjunction with OSSEC.

Fail2ban is also an effective measure for flood control and can stop most denial of service (DoS) attacks. Drupal also has some built in flood control options; The Flood Control module provides a UI to control them. Note that some crackers are throttling their dictionary attacks to avoid tools like Fail2ban, so it will still be important to use a monitoring tool like Nagios to monitor if someone is methodically trying to guess bad passwords on your server.

```
# Ubuntu/Debian
$ apt-get install fail2ban

# CentOS
$ yum install fail2ban
```

Distributed Denial of Service (DDoS) attacks are more difficult to address, but there's a great defence plan laid out on StackOverflow.

Place the /etc directory under version control so that you can easily track which configurations have changed. The program etckeeper automates this process nicely and hooks into your package manager and cron to do its work when your server is upgraded or new software is installed.

```
# Ubuntu/Debian
$ apt-get install etckeeper bzr && etckeeper init && etckeeper commit
"initial commit"

# CentOS
$ yum install etckeeper && etckeeper init && etckeeper commit "initial
commit"
```

Ubuntu comes with the Ubuntu Popularity Contest (popcon) to gather statistics about which packages are used in the community. Although this is anonymous, it can be a good idea to remove this package so that it is not a potential weak link. This is an optional package that can be easily removed without impacting your site's performance.

```
# Ubuntu
$ dpkg --purge popularity-contest ubuntu-standard
```

You will probably want to install an opcode cache and Memcache (or Redis) to ensure that your site is responding quickly. PHP 5.5+ now comes with built in opcode cache, earlier versions of PHP can add this using APC. Memcached is a general-purpose distributed memory caching system. Both work to make your server more responsive by minimizing the load on the server and improving caching. This will help when there is an unexpected server load.

Aside from the performance advantages, there can be security improvements by caching the public display. There are huge security advantages to restricting access to the rendering logic (Drupal's admin) so that the public is only interacting with a cache serving front end content. In using any caching however, it is critical that only anonymous data is cached. A mis-configured cache can easily expose personal data to the public. This needs to be carefully tested on sites which have private or confidential data.

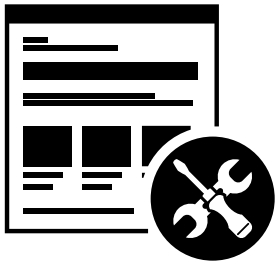
There are a number of ways to cache the public display, including leveraging Memcache and Nginx to extend Drupal's internal page cache. One of the most powerful tools is Varnish which can provide incredible performance enhancements. It can also be used effectively to deny all logins on your public site by being configured to deny cookies on port 80. This is a line that can be added to your Varnish vcl file in the *vcl_recv* subroutine to remove the cookies so that it becomes impossible to login to Drupal directly:

```
if (req.http.host == "example.com") { unset req.http.Cookie;}
```

If you have a site which has only a few users and doesn't have any online forms for anonymous users then you can configure Varnish to simply reject all HTTP POST requests. Then in Apache you can whitelist the IP address you want to have access to login to Drupal. Matt Korostoff documented this approach in his breakdown of the Drupalgeddon attacks that affected many Drupal 7 sites.

Shared server environments provide a number of security challenges. Do not expect it to be easy to securely host several sites on the same server with direct shell access to different clients. If you need to do this, it is worth investigating FastCGI which when used in conjunction with suexec or cgiwrap to isolate individual processes on a shared server. We expect most medium to large organizations to have access to either virtual (e.g. VMware, Xen, OpenVZ or KVM) or cloud-based (e.g. Amazon or Rackspace) servers. There is also significant movement in the Drupal community to use Linux Containers to more efficiently distribute processing power without compromising security.

6) Server Maintenance



Created by Yamini Ahluwalia from the Noun Project

Security requires constant vigilance. Someone should be tasked with ensuring that the server is kept up-to-date at least weekly. This isn't usually a complex task, but it does require that someone subscribe to the security update mailing list for the distribution (e.g. Ubuntu and CentOS), apply the updates, and review the logs to ensure everything is still running properly. Upgrades can be done with the following commands:

```
# Ubuntu/Debian
$ apt-get update && apt-get upgrade

# CentOS
$ yum upgrade
```

It is very useful to have a service like Nagios monitoring your production server to alert you if any problems arise. The configuration of Nagios can be quite complex, but it can be setup alongside your web application or on a dedicated monitoring server. You will need to grant access on your production environment to this server and you must enable CGI access on this server. Remember that if you enable this, you will also need

to consider the security implications that it presents. To get the server installed in your environment, execute the following from the command line:

```
# Ubuntu/Debian
$ apt-get install nagios3 nagios-nrpe-plugin
```

And for each server you wish to monitor with Nagios:

```
# Ubuntu/Debian
$ apt-get install nagios-nrpe-plugin
```

Munin can be run on the production environment to give you a sense of the relative load of various key elements over the past hour, day, week and month. This can be useful when debugging issues with your server.

```
# Ubuntu/Debian
$ apt-get install munin munin-node
```

Access to this information is available through your web server but you will want to configure your site to ensure that this data is not publicly available.

There are also many good reasons to use server configuration management software like Puppet or Chef. Initially, it will take you a lot more time to configure it this way, but it will make it much easier to restore your server when something does happen and and see you are back online quickly. It also codifies the process to ensure that you don't miss critical setup steps. This approach also makes it trivial to have essentially identical development, staging and production environments.

7) Managing Server Logs

Your web server is a complex environment involving thousands of software projects. Most of these will store log files in /var/log. If log files aren't properly rotated and compressed they can become unmanageably large. If your hard drive is filled up with old log files your site will simply stop functioning. Most distributions of Linux come with logrotate configured such that log files are segmented on a regular basis and the archive is compressed so that space isn't a problem.

Most Linux distributions also come with syslog built in, which is critical for doing security audits. You can also configure it to send emergency messages to a remote machine, or even send a duplicate of all log messages to an external loghost. An experienced

cracker will modify your log files to obscure their work, once they have compromised your server.

There is a discussion in the Drupal section later on about how to direct Watchdog messages to syslog. There are many tools to help system administrators more effectively monitor their log files, and regular log reviews can be an important part of early breach detection.

If your server is configured with a caching reverse proxy server or a load balancer such as Varnish, Nginx or haproxy then you should ensure that Drupal is made aware of the actual `REMOTE_IP`. The common solution requires configuring the `X-Forwarded-For` header in both Varnish and Apache, but as Jonathan Marcil's blog post points out, "X-Forwarded-For is actually a list that can be a chain of multiples proxies and not just a single IP address". To that effect, ensure that all IP addresses for your reverse proxies are identified in your settings.php file (configuration). Another solution would be to create a custom HTTP header such as `X-Remote-IP` and tell Drupal to use it using the configuration variable "reverse_proxy_header" in `settings.php` under "Reverse Proxy Configuration". Drupal itself will correctly manage a list of trusted reverse proxies with this header. This is useful if you want to correctly log IPs at a Web server, proxy or load balancer level. Note that the reverse proxy should append the IP address of the client issuing the request to this header.

```
$conf['reverse_proxy'] = TRUE;
$conf['reverse_proxy_addresses'] = array('127.0.0.1','192.168.0.2');
$conf['reverse_proxy_header'] = 'HTTP_X_REMOTE_IP';
```

Another approach to dealing with this is to simply use Apache's Reverse Proxy Add Forward (RPAF) module. As Khalid Baheyeldin writes in his blog, this Apache module can be used for both Reverse Proxy and/or a Content Delivery Network (CDN).

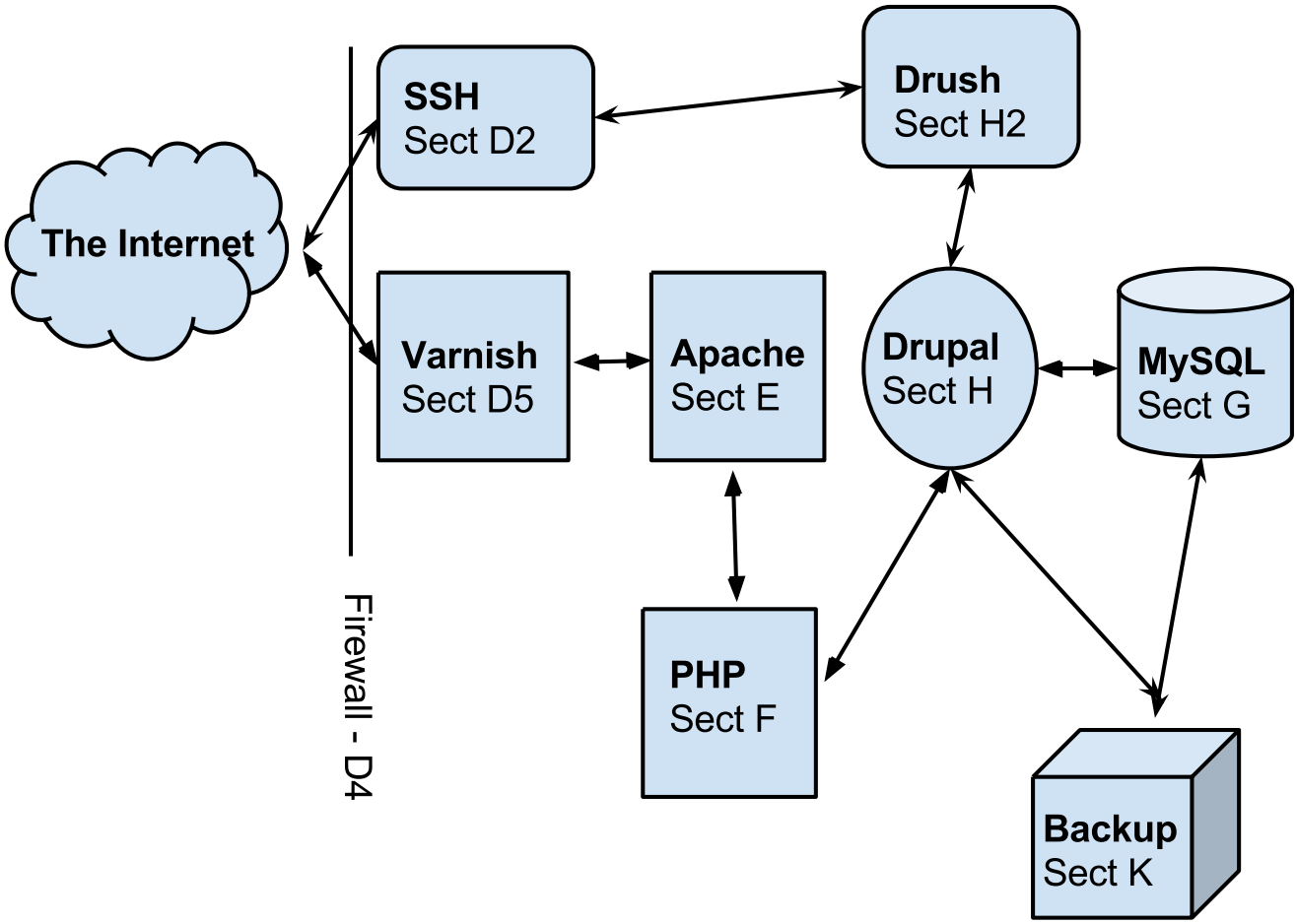
```
# Ubuntu/Debian
$ apt-get install libapache2-mod-rpaf
```

After editing `/etc/apache2/mods-enabled/rpaf.conf` and setting your proxy IP and restarting Apache, your access.log will show the real client IP rather than that of your proxy.

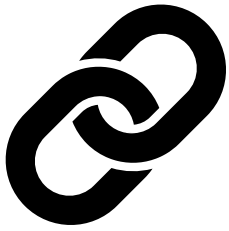
The most important server logs to monitor are Apache's. If there is more than one site on a given server, it is normal for each site to have its own log file rather than using the

default generic one. If you run more than one site or have multiple web servers, log centralization can allow you to get an overall view of site issues. Open-source tools such as logstash can be used to simplify the process of searching all of your log files.

8) Rough Server Ecosystem Image



E) Web Servers



All files and directories in your DocumentRoot should be editable by a non-root user and should also not be writable by the Apache user, except the Drupal files/ directory. Please refer to Drupal's Securing file permissions and ownership for the complete discussion.

PHP-FPM over FastCGI allows your server to have site-specific "pools" of PHP. By giving each site unique PHP permissions you can effectively "sandbox" a PHP application and simplify file/folder permissions by specifying the user and group for the process pool. This reduces the points of failure in a shared hosting environment, where the PHP on another site could be used to hijack the server. There are also real advantages to using PHP-FPM for managing server load as Apache's `mod_php` isn't very efficient.

Server or browser support for SSL versions 2 and 3 are not recommended. Despite this, as Google noted in their blog post about the POODLE Exploit, "SSL 3.0 is nearly 18 years old, but support for it remains widespread." Many older browsers still support this insecure version of SSL, but it is easy to test to ensure if your browsers are vulnerable and the number of exposed users is falling. Qualys SSL Labs also have a really great tool to evaluate if your server is still vulnerable.

On your web server, it is good to ensure that SSL configuration permits only TLS version 1.2. Unfortunately some common web browsers still do not support the latest version of TLS. Fortunately, as of September 2015, the latest version of all major web browsers support only secure TLS 1.0, 1.1, and 1.2 by default. Check if the SSL services employ only AES with key lengths 256 bits and higher. You can install GnuTLS from the command line to enable this:

```
# Ubuntu/Debian
$ apt-get install gnutls-bin
```

It is also recommended to disable SSLCompression in Apache. As stated in the Apache documentation "Enabling compression causes security issues in most setups (the so called CRIME attack)." This is the default for Apache version 2.4.4+.

The **HeartBleed security bug** did a lot of damage in 2014. The primary security practice we can recommend from this is to ensure that someone is always paying

attention to the security mailing lists for your operating system. By the time you hear it from the media it is probably too late. The other suggestion is one that is suggested by the EFF and others which includes implementing Perfect Forward Secrecy (PFS). Although we didn't explicitly refer to it as this in earlier versions of this document, the hardened SSL configuration we recommended in the fall implements this.

The duraconf configuration scripts for hardening SSL/TLS services provided by Jacob Appelbaum would have protected users from the HeartBleed bug. In the Apache config you can set hardened SSL configurations for the HTTPS protocol (note that we're now using Hynek Schlawack configuration rather than Ivan Ristic's because it is being updated more regularly) with:

```
SSLProtocol ALL -SSLv2 -SSLv3
SSLHonorCipherOrder On
SSLCipherSuite
ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:ECDH+3DES:D
```

After restarting Apache, you can check the SSL information in a browser by double clicking on the lock icon in the address bar on HTTPS sites to get information on the encryption channel and confirm it's using TLS.

There are other approaches like that suggested by Remy van Elst, but ultimately you need to test your SSL configuration through a tool like Qualys SSL Labs' Server Test. This is a free online service that performs a deep analysis of the configuration of any SSL web server on the public Internet. This will grade your SSL compliance and do things like confirm that you are using the latest version of TLS and verify that you are protected from BEAST attacks. You want straight A's!

On your staging/development server it is fine to provide a self signed SSL certificate to ensure that the traffic is encrypted. Setting up a third party verified SSL certificate on your production environment will be important as otherwise your users will be asked to verify the exception when accessing the HTTPS version of your site. A listing of certificate authorities is available at the bottom of this Wikipedia page. You can review the validity of your SSL certificate through a free SSL Test constructed by SSLLabs or with the following openssl command:

```
$ openssl s_client -connect SERVER:443
```

To check a specific protocol using openssl:

```
$ openssl s_client -connect SERVER:443 -ssl2
$ openssl s_client -connect SERVER:443 -ssl3
```

Note that SSL Certificate Authorities are deprecating the very popular SHA1 hashing function because of weakness in the algorithm. Qualys Labs recommends renewing with SHA256 as soon as possible.

1) Restricting Access

Another useful Apache module is mod_authz_host which can restrict access to specific pages such as `/user` and `/admin/*` - this can be useful if your site is used just as a CMS with no user interaction. The example below is more appropriate for sites which would have broader user authentication, but where users are restricted from editing nodes - `node/*/edit` - this type of approach can also be used to restrict access to non-production environments. If you have a multilingual site using a language prefix, you may also want to check that sensitive paths are restricted with the language prefix. In Canada, many sites would need to add `/fr/user` and `/en/user` . It is a best practice to secure all pages on non-production environments from search engines, but especially from crackers. The following are examples of how to do this with `mod_authz_host` and also `mod_rewrite`.

Example Apache configuration using `mod_authz_host`:

```
<Location - "/node/.*/edit">
  Order Deny,Allow
  Deny from all
  Allow from 192.0.2.1 192.0.2.25
</Location>
```

Example Apache configuration using `mod_rewrite`:

```
<IfModule mod_rewrite.c>
  RewriteEngine on
  # Allow only internal access to admin
  RewriteCond %{REMOTE_ADDR}
  !^(192\.0\.2\.1|192\.0\.2\.25)$
  RewriteRule ^admin/.* - [F]
```

```
</IfModule>
```

Drupal has a number of processes that can be triggered by URLs. You may wish to block some of these using Apache so that they cannot be loaded from the web browser. Common processes to secure are update, install and cron, tasks which can all be triggered using Drush:

```
RedirectMatch 403 "/(install|update|cron|xmlrpc|authorize).php"
```

2) Removing Code

CGIs have been used extensively in web development and there are a great many good server executables that you may want to consider running. However, many CGIs that may be installed on a server are not actually needed and expose you to an additional security risk. If you are not running any CGIs, you should disable CGI access by removing `LoadModule cgi_module` and `AddHandler cgi-script .cgi` from your Apache config. You can also do this from the command line with:

```
# Ubuntu/Debian
$ a2dismod cgi
```

If you don't need it, remove it. All software is a source of potential risk, so list all Apache modules and look for unneeded modules. There are some good discussions on drupal.org about which modules are necessary and which are not.

```
# Ubuntu/Debian
$ apache2ctl -t -D DUMP_MODULES

# CentOS
$ apachectl -t -D DUMP_MODULES
```

If you are using `mod_php` with Apache, it can be useful to install `php5-dev` so that you can build tools like PECL's uploadprogress. However, after you've done that you will want to remove the php5-dev module that you used to build it:

```
# Ubuntu/Debian
$ apt-get remove php5-dev
```

You can find other development packages on your server by:

```
# Ubuntu/Debian
$ apt-cache search ".-dev"
```

3) HTTP Headers

The Australian Government has produced an impressive report Information Security Advice for All Levels of Government which is sadly a bit out-dated as it hasn't been updated since early 2012. Most of that report is focused on content security policy, HTTP strict transport security and frame options.

The Security Kit Drupal module addresses many security problems associated with HTTP headers, but it is good to have them addressed at the Apache layer where possible.

The W3C is developing a standard content security policy (CSP) to provide security controls which can mitigate attacks such as Cross Site Scripting (XSS).

Mozilla has produced a good description of how to write a CSP and there are many commonalities with the Australian Government report above. To allow content from a trusted domain and all its subdomains, you can add the following to your Apache configuration:

```
Content-Security-Policy: default-src 'self' *.example.com
```

Your website and its visitors are going to be more secure if you use HTTPS to ensure that all information passing between the web server and the browser is encrypted. There is a growing movement encrypt all web traffic, even to brochure sites. Google announced in 2014 that HTTPS would be a ranking signal. Doing so will have minor performance implications as it does take some additional processing power. You certainly want to ensure that all authentication happens through a secure HTTPS connection so that usernames and passwords cannot be intercepted. Do ensure that all of your files are also being served from a HTTPS environment, as mixed-mode (HTTP and HTTPS) traffic introduces usability problems.

```
<VirtualHost *:80>
  ServerAlias *
  RewriteEngine On
  RewriteRule ^(.*)$ https://%{HTTP_HOST}$1[redirect=301]
```

```
</VirtualHost>
```

This can be further enhanced by opting into the HTTP Strict Transport Security (HSTS) enhancement which sends a special response header to the browser, which then prevents any communications from being sent over HTTP to the specified domain (see HSTS example):

```
Header set Strict-Transport-Security "max-age=16070400;
includeSubDomains"
```

You can also submit your site to the EFF's HTTPS Everywhere extension which will allows security conscious individuals to rewrite requests to these sites so that they use HTTPS by default. As part of this extension, you can submit new public rules for your site to ensure that it runs optimally with this browser extension.

With the use of Frame Options, users can be exposed to Clickjacking when an iframe is injected in your site. If you know that you aren't going to need to use iframes in your site you can disable it by modifying the Force X-Frame options in the Apache configuration. As usual, OWASP has an extremely useful guide on avoiding Clickjacking. You must have the mod-headers module enabled before adding this string to your Apache configuration but this is easy to add through the command line - a2enmod headers - afterwards you can add this to your configuration.

```
Header always append X-Frame-Options SAMEORIGIN
```

4) HTTP Basic Authentication

Most webservers provide a way to restrict access to a site using HTTP Basic Authentication — for example, using Apache HTTP Server's htpasswd files or Auth* directives, or nginx's ngx_http_auth_basic_module module.

While HTTP Basic Authentication is a good way to prevent search engines from indexing your testing and staging sites, it is inherently insecure: traffic between browsers and your site is not encrypted, and in fact if an attacker has the ability to sniff network traffic, this person can gain access to the site simply by copying the "Authorization" HTTP header sent by the browser.

Furthermore, the username and password used for HTTP Basic Authentication are not encrypted either (just base-64 encoded, which is trivial to decode), so do not re-use credentials used elsewhere (e.g.: each unique login should have its own unique

password).

It is strongly recommended to store the htpasswd file outside the document root and to set it with read only permissions (440).

5) Web Application Firewall

Web Application Firewalls (WAFs) can be used to provide additional protection over the Web server. It can be a standalone server that acts as a reverse proxy or installed as Web server modules.

Apache has a number of modules that can be installed to tighten security of the web server. We recommend installing ModSecurity and mod_evasive as a Web Application Firewall (WAF). This can be set to leverage the Open Web Application Security Project's (OWASP) ModSecurity Core Rule Set Project.

```
# Ubuntu/Debian
$ apt-get install libapache2-mod-evasive libapache2-modsecurity
$ a2enmod mod-security; a2enmod mod-evasive

# CentOS
$ yum install mod_evasive mod_security
```

To engage ModSecurity in your Apache, you'll need to set up the base files in your Apache configuration and then restart Apache. To set the chroot for Apache, using mod_security, make sure to define the directory you want to restrict.

```
SecChrootDir /chroot/apache
```

Using default generic configurations such as the OWASP Core Rule Set can impact the normal behaviour of Drupal and must be tested extensively before deployment. Usually some rules are breaking rich content edition or modules that behave differently than Drupal core. It is recommended to run the rules in a passive manner in order to identify false positives when in production. Default configuration of ModSecurity should do it with:

```
SecRuleEngine DetectionOnly
```

You can then set it to "On" whenever you are ready. A server restart is needed for changes to be effective. In that case the WAF will behave as a passive Web application

intrusion detection system and you can choose to never set it to “On” if you wish to use it only for that purpose. In any case, you’ll want to monitor the log files for alerts in order to detect malicious attempts and potential false positives.

WAF software needs maintenance as well and rules should be updated periodically. Tests for false positives should be made after each change of functionality within the Drupal site.

WAFs are a great solution for virtual patching and application flaw fixing, but they can be bypassed. It is discouraged to rely solely on that technology to keep up with security: fixing flaws and applying patches on the back-end applications should not be replaced with WAF utilization.

6) Managing the .htaccess file

Mike Carper has suggested a clean way of incorporating the *.htaccess file within the Apache config*. Using Apache includes to incorporate the .htaccess file provided by Drupal.org makes routine security upgrades much easier. When the .htaccess file changes, this will automatically be included by Apache. Unfortunately, Apache will need to be restarted before this will come into effect. There will be performance improvements by not loading the .htaccess file with every page load. You can also force some security rules right in the configuration.

7) MIME Problems

Some attacks are based on MIME-type confusion. You can help protect against this in some browsers if the server sends the response header

```
X-Content-Type-Options: nosniff;
```

In Chrome & Internet Explorer, script & stylesheet elements will reject responses when this is sent in the page header. There is an existing issue for this security feature in Firefox.

This can be set in Varnish with:

```
set beresp.http.X-Content-Type-Options = "nosniff";
```

“This reduces exposure to drive-by download attacks and sites serving user uploaded content that, by clever naming, could be treated by MSIE as executable or dynamic HTML files.” - OWASP List of Useful Headers

You can get Apache to force the MIME types for documents using mod_mime. The Apache MIME module maps file extensions to MIME-type (content-type) of documents.

```
# Ubuntu/Debian
$ a2enmod mime

# CentOS
$ yum install mod_mime
```

Unfortunately, any allowed file upload like a .jpg can actually be javascript content. This could contain an OBJECT tag on another domain which would enable CSRF and data hijacking.

You can force downloads for uploaded files like

```
Header set Content-Disposition "attachment"
```

but in most cases it will be more user friendly simply to use a different domain or a subdomain to avoid sending session cookies with uploaded files.

8) Everything Else

Modify the web server configuration to disable the TRACE/TRACK methods either by employing the TraceEnable directive or by adding the following lines to your Apache configuration:

```
RewriteEngine On
RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK)
RewriteRule .* - [F]
```

You should keep your server up-to date. Security by obscurity may delay some crackers, but not prevent them from accessing your system. Looking at the logs for any popular site, you will notice thousands of fruitless attempts at exploits that may not even exist (or have existed) on your system. Broadcasting information about your server environment isn't likely to cause any harm, but if you choose to disable it you can simply add this to your Apache configuration:

```
ServerSignature Off
ServerTokens ProductOnly
```

Denial of Service attacks can be mitigated by lowering the Timeout value from the default 300 seconds.

```
Timeout 45
```

Likewise the LimitRequestBody, LimitRequestFields, LimitRequestFieldSize and LimitRequestLine can be set to lower levels, however this can cause problems with file uploads though, so be careful about setting this value.

```
LimitRequestBody 1048576
```

One of the nice things about Ubuntu/Debian is that the Apache file structure is clean. By default it allows you to store a variety of different configurations for sites or modules that are stored in logical directories. That's not critical, but having a well defined Apache config file is. There should be inline comments about all changed variables explaining why they were added or modified.

It is possible to restrict the outgoing access of the web server by leveraging iptables' "–uid-owner" option on the OUTPUT table. This can also be done using containers and namespaces on modern Linux kernels. In most cases, if you are using containers, the UID of Apache will be the same inside the container as outside of it.

You should make note of the user/UID of your web server. This is dependent on the package installation order, but often this is "www-data" (uid 33) in Debian/Ubuntu and "nobody" (uid 65534) in CentOS. If you are using PHP-FPM, then you will need to search for the UID of that application rather than Apache's. Apache should never be run as your user login, although this is common in shared hosting environments.

Double check by viewing the output of:

```
# Ubuntu/Debian
$ ps aux | grep apache

# CentOS
$ ps aux | grep http
```

In order to restrict Apache to connect only to https://drupal.org (with IP addresses 140.211.10.62 and 140.211.10.16 at the time of writing) insert the following firewall rules:

```
iptables -A OUTPUT -m owner --uid-owner ${APACHE_UID}
-p udp --dport 53 -j ACCEPT

iptables -A OUTPUT -d 140.211.10.62/32 -p tcp -m
owner --uid-owner ${APACHE_UID} -m tcp --dport 443 -j ACCEPT

iptables -A OUTPUT -d 140.211.10.16/32 -p tcp -m
owner --uid-owner ${APACHE_UID} -m tcp --dport 443 -j ACCEPT

iptables -A OUTPUT -m owner --uid-owner ${APACHE_UID}
-m state --state NEW -j DROP
```

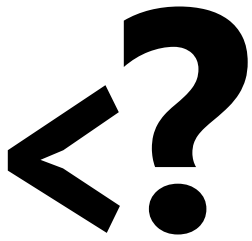
There are also Apache modules like Project Honey Pot that make it harder for people to hack your system. Honey Pot can also be installed on Drupal, but Apache is often more efficient at addressing attacks like this before it hits PHP:

```
# Ubuntu/Debian
$ a2enmod httpbl

# CentOS
$ yum install mod_httpbl
```

F) PHP

There are lots of good resources on how to tighten security for PHP. It is a very commonly used scripting language and it is running some of the biggest and most important sites on the Internet.



Created by icons.design from the Noun Project

We recommend installing a PHP hardening tool called Suhosin which tightens up PHP's existing configuration so that it is more robust. It is designed to protect servers and users from known and unknown flaws in PHP applications and the PHP core.

```
# Ubuntu:
# Enable "universe" repo
$ vi /etc/apt/sources.list
$ apt-get update ; apt-get install php5-suhosin

# Debian:
$ apt-get install php5-suhosin

# CentOS:
$ yum install php-suhosin
```

A good comprehensive list is from Justin C. Klein's blog post Hardening PHP from php.ini. Other than his comments on `safe_mode`, we think he's got it right. Drupal needs `safe_mode` disabled in PHP. PHP's Safe Mode isn't really considered much of a security enhancement and it has been removed in recent versions.

As with Apache modules, look for what you can remove. You can display a list of enabled PHP modules and look for those which can be removed. From the command line you can get a list of PHP modules with:

```
$ php -m
```

Setting ini Variables

Many PHP variables can be set via Apache as well as in the PHP configuration. We recommend keeping PHP-specific security configuration centrally located in the `php.ini` file.

Another exploit is Session fixation where a user's browser session can be hijacked by a third party.

OWASP goes into much more detail, but by using the HttpOnly flag when generating a cookie you can reduce the risk of an XSS attack by limiting access to protected cookies. It is advised to stop Javascript from accessing cookie data. Session information should only ever be passed to the server with the same domain. You can also set a secure cookie attribute and restrict all transmission of cookie data to an HTTPS connection to ensure that the cookie is less likely to be exposed to cookie theft via eavesdropping. Furthermore, you can control the hash algorithm used to generate the session ID and choose from a number of algorithms like the NSA's SHA-2 protocol or whirlpool. Add the following to your php.ini file:

```
session.cookie_httponly = 1
session.use_only_cookies = 1
session.cookie_secure = 1
session.hash_function = whirlpool
```

You can obtain a list of the available hash functions on your system by executing:

```
$ php -r 'print_r(hash_algos())'
```

Limit your exposure to only the system resources you want to make available to a PHP page. You can control your resources by limiting the `upload_max_filesize`, `max_execution_time`, `max_input_time`, `memory_limit` variables so that a script isn't as likely to monopolize resources:

```
memory_limit = 128M
max_input_time = 60
max_execution_time = 30
upload_max_filesize = 2M
```

By keeping up with security releases some will argue that there is no need to hide which version of PHP you are running. There is a broader discussion of this debate in Section L) Points of Debate under `Security by Obscurity`. In the PHP setting you can also limit information about PHP which is exposed by adding this to your `php.ini` file:

```
expose_php = Off
```

You can also explicitly disable PHP functions which allow scripts to reference other URLs:

```
allow_url_include = Off
allow_url_fopen = Off
```

You can also disable PHP functions which are considered dangerous. You will want to test to see that your Drupal install doesn't require any of these functions. You can grep from the Drupal root to find out if your site uses any of these functions. Drupal's PHP filter leverages the exec function, however there are lots of good reasons not to use the PHP filter. You can add this to your `php.ini` file:

```
disable_functions = php_uname, getmyuid, getmypid, passthru, leak,
listen, diskfreespace, tmpfile, link, ignore_user_abord, shell_exec, dl,
set_time_limit, exec, system, highlight_file, source, show_source,
fpaththru, virtual, posix_ctermid, posix_getcwd, posix_getegid,
posix_geteuid, posix_getgid, posix_getgrgid, posix_getgrnam,
posix_getgroups, posix_getlogin, posix_getpgid, posix_getpgrp,
posix_getpid, posix, _getppid, posix_getpwnam, posix_getpwuid,
posix_getrlimit, posix_getsid, posix_getuid, posix_isatty, posix_kill,
posix_mkfifo, posix_setegid, posix_seteuid, posix_setgid, posix_setpgid,
posix_setsid, posix_setuid, posix_times, posix_ttyname, posix_uname,
proc_open, proc_close, proc_get_status, proc_nice, proc_terminate, popen
```

Drupal's status page has a link to the output of phpinfo and you should decide whether or not you want to exclude that function in this list. You want to be able to limit what PHP has access to in the file system. Note that you may want to give slightly more access to PHP than just the Drupal root directory as it can be beneficial to put some files (like a salt.txt file) outside of the base directory. This can also be set in Apache, but you may wish to keep the PHP specific information inside the `php.ini` file:

```
open_basedir = /var/www
```

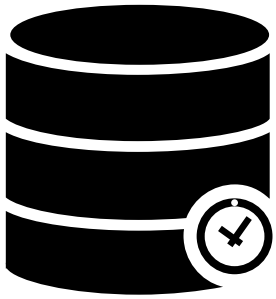
Make sure the session path is outside the root web directory and not readable or writable by any other system users. You will also want to set a temporary upload file directory that is outside of the web root. This can be specified in the php.ini file:

```
session.save_path = "/tmp"
```

```
upload_tmp_dir = "/tmp"
```

G) Database Layer

With Drupal's database abstraction layer you can now run it on MySQL, SQLite, or PostgreSQL out of the box. There are in fact a number of popular MySQL forks like MariaDB and Percona. Drupal can run with MSSQL too, but there will be more support for MySQL flavours of SQL. Oracle support for PHP is weak, so it is not recommended to use this database. We are not aware of any security advantages of one over the other.



Created by Mister Pixel

The database for Drupal can run on the same server, but for performance reasons it can be beneficial to set it up on another server. You want to ensure that your server environment is robust enough that it cannot be easily brought down by a Denial of Service (DoS) attack. There are a few server side tools to help with this, but mostly it's useful to have a buffer, even at your highest traffic times, so that your site is always responsive.

At the point where your server environment spreads on to more than one system, it begins to make sense to have a second network behind the web server, possibly including a VPN. It is quite likely that if the database is moved to an external server that there may soon be other servers including more than one front-end server too.

There is a lot that can be done to secure your database. Much of it comes down to reviewing access permissions for the Drupal user (set in Drupal's settings.php), the backup user (which has read only access to do regular backups) and the database's root user (which obviously has access to everything) and verifying that they all have complex passwords. These need to be unique passwords and the root password should not be stored on the server, but rather in your encrypted Keepass database.

If your server is running locally, you can disable access for MySQL to the network and force it to only use the internal IP address. If your web server and database are on different servers, you won't be able to do this, but you will be able to restrict what address MySQL will listen on. If your web server and database server share a LAN, bind MySQL only to the LAN IP address and not any Internet-facing ones. For a machine running both the web server and MySQL, you can add this to your my.conf file:

```
bind-address=127.0.0.1
```

Database Access

Be sure to review your databases, users and permissions to see that there are not any sample users or old databases still enabled on the server and that you are not giving greater access to a user than they need. You should also review the file system to see that the database files are restricted.

Drupal 7 and 8 require that the database have GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER, CREATE TEMPORARY TABLES permissions. This gives more permission than is generally required, but does cover both new installations and updates. There are a few options presented here for applying more fine grained permissions .

PHPMyAdmin

If you need a graphical tool like phpMyAdmin, disable it after use. Web applications like this can also be tightened down by placing them on a different port, firewall that port from other than 127.0.0.1, and always access it via SSH port forwarding. Access to these tools can also be limited to IP addresses for extra protection. Note that any software you use should be regularly updated to ensure that it receives any security enhancements, particularly if stored on the server. You can restrict access to phpMyAdmin via .htaccess or by configuring Apache to request an HTTP username/password login. They can also be restricted to only allow access from certain trusted IP addresses. This is an important vulnerability as it could give a cracker full access to your databases. It can be beneficial to put phpMyAdmin in its own VirtualHost and even run it on a non-standard port. Force HTTPS connections to phpMyAdmin - do not use regular HTTP. Also consider the implications of allowing database access via the web server: There is little benefit if you have restricted which interfaces MySQL will listen on, as described above, but then allow control of the database from an Internet-facing web page.

Drupal modules like Security Review can be useful to alert administrators if there are a large number of database errors. This is an indication of a possible SQL injection attack (SQLi attempt).

H) Drupal

1) Files

Verify Drupal file permissions on the server. You really need to restrict write access to the server and verify that the right users/groups have the access that they need for Drupal to operate effectively. One can set all of Drupal's files to be read only, only allowing for file uploads, cached files, sessions and temporary directories with write permissions. The major limitation of this approach is that it makes applying security upgrades more difficult.

By default, Drupal 7 disables execution of PHP in directories where you can upload PHP. Check the `.htaccess` file in the files directory. You can also add this to your Apache Virtual Host to ensure that no handlers have been overlooked.

```
<Directory /var/www/drupal/sites/default/files/>
 # Important for security, prevents someone from
 # uploading a malicious .htaccess
 AllowOverride None
 SetHandler none
 SetHandler default-handler
 Options -ExecCGI
 php_flag engine off
 RemoveHandler .cgi .php .php3 .php4 .php5 .phtml .pl .py .pyc .pyo

 <Files *>
  AllowOverride None
  SetHandler none
  SetHandler default-handler
  Options -ExecCGI
  php_flag engine off
  RemoveHandler .cgi .php .php3 .php4 .php5 .phtml .pl .py .pyc .pyo
 </Files>
</Directory>
```

Drupal needs to be able to write to the server to be able to perform certain tasks like

managing file uploads and compressing/caching CSS/JS files. Ensure that Apache has write access to /tmp and also to the public sites folder:

```
# Ubuntu/Debian:
$ chown -R www-data:www-data sites/default/files

# CentOS:
$ chown -R nobody:nobody sites/default/files
```

Make sure that you are only allowing users to upload file types that have limited security problems with them. Text and images are usually quite safe. There have been some exploits on PDF files, but they are quite rare. Microsoft Office documents should be scanned if they are going to be uploaded onto the server.

ClamAV can be incorporated into Drupal to scan uploaded files for viruses and other malicious code. Acquia recommends excluding the following file types: *flv*, *swf*, *exe*, *html*, *htm*, *php*, *phtml*, *py*, *js*, *vb*, *vbe*, *vbs*.

As Peter Wolanin pointed out in his Drupal New Jersey presentation , If you allow untrusted users to upload files there are additional issues you should be concerned about. The browser security model - same-origin policy - permits scripts contained in a first web page to access data in a second web page, but only if both web pages have the same origin. This is dependent on using session cookies to maintain authenticated user sessions.

On the Webserver's chapter under MIME Problems, we recommend a separate domain or subdomain to avoid MIME confusion problems with user contributed (untrusted) content. As Peter Wolanin notes, the CDN module or a small custom module could be used to :

- Serve uploaded files from a subdomain or different domain
- Use sites dir or redirect to prevent Drupal on the files domains
- Block public files on the Drupal domain

The use of a Content Delivery Network should also improve the load times of the site and reduce the threat of DOS Attacks.

Peter provides the following sample function:

```
function mymodule_file_url_alter(&$uri) {
  if (file_uri_scheme($uri) == 'public') {
```

```
    $wrapper = file_stream_wrapper_get_instance_by_scheme($scheme);
    $path = $wrapper->getDirectoryPath() . '/' . file_uri_target($uri);
    $uri = 'http://downloads.drupal-7.local:8083/' . $path;
  }
}
```

When you navigate to your site's File System settings page `/admin/config/media/file-system` and save the settings, Drupal will write a restrictive .htaccess file into your Public Files Directory which will limit exposure of the files contained there.

In Drupal 8 you will be able to specify the Public file base URL so that this will be easier to control.

2) Drush

Drush is a command line shell and scripting interface for Drupal. We strongly recommend using Drush on both staging and production servers because it simplifies development and maintenance. Note that the version of Drush packaged with your OS is likely to be extremely out of date.

It is recommended to install Drush with Composer (the dependency manager for PHP) but other options and details on the Drush git page.

There is a Security Check module available for Drupal which is a basic sanity test for your configuration. When the module is added, you can run this against your site from the site directory on the command line using:

```
$ drush secchk
```

As with the server configuration in general, document what you are using. Drush makes this fairly straightforward as you can simply export a list from the command line:

```
$ drush pm-list --type=Module --status=enabled
```

Cron is the Linux time-based job scheduler and it is used for a lot of key Drupal functions. Check to see that you are running cron several times a day. For Drupal 7 and above, if there is traffic to the site, cron jobs are run every 3 hours. The status page will tell you when the last time cron was run on the site. You may want to set up a Linux cron job using Drush if you have either a low traffic site or have special requirements.

To run cron on all of your sites in /home/drupal every half hour, from the command line enter `crontab -e` and then insert:

```
0,30 * * * * cd /home/drupal && drush @sites core-cron -y > /dev/null
```

You will need developer modules to help you build your site, but they are a security risk on your production site and need to be disabled. Many modules (such as Views) have separate administration screens that can also be disabled in a production environment. They are absolutely required when building the site, but can be disabled when they are not in use. It is always a good practice to see if there are any unnecessary modules that can be disabled on your site. This also offers performance benefits. Views is an incredibly powerful query building tool. Because of that, it is important that all Views have explicit access permissions set at `/admin/build/views` .

3) Errors

Check the Status Report and Watchdog pages regularly and resolve issues - Drupal should be happy! This needs to be done regularly, even after launch. Remember that you can more quickly scan your logs by filtering for PHP errors. With the Views Watchdog module you could also build custom reports to display on your website. On your production server, make sure to disable the display of PHP errors. These should be recorded to your logs, but not visible to your visitors. On your staging site you will want to see those errors to help you debug PHP problems, but it is a potential vulnerability to have those exposed. This won't catch all PHP errors however, and so it is also useful to review the error log of the web server itself.

Watchdog is a good tool, but is limited in a number of ways. Simply because it is database dependent, even having a lot of 404 errors can affect performance. Fortunately, logs can be easily directed to the server's syslog, with the Syslog Access module, which also allows you to leverage your favourite log management tool. The Drupal Handbook also has a great resource for how to send your logs to Syslog with integrated logging.

4) Core and Contrib Hacks

Before launching your site (and periodically afterwards) it is useful to run the Hacked! module to check what code differs from what was released on Drupal.org. Particularly when the diff module (6/7/8) is enabled, this is a powerful tool to evaluate your code. There are millions of lines of code in a given Drupal site, so Hacked! is a really valuable analysis tool. If you need to apply patches against the stable released version of the

code, the patch should be in a clearly documented directory. It is unfortunately a common practice for less experienced Drupal developers to cut corners and hack Drupal Core to provide some functionality that is required. There are lots of reasons why this is a bad idea and why responsible developers don't hack core. For the purposes of this document it is sufficient to say it makes it harder to secure. The same is true for contributed modules, you shouldn't have to alter the code to customize it most of the time. The Hacked! module is very useful in identifying when modules are no longer the same as their releases on Drupal.org. Being able to quickly scan through hundreds of thousands of lines of code and find differences against known releases is a huge security advantage.

You can also generate Drush make file from an existing Drupal site and then recreate a clean copy of the code-base which you can then diff (a command line comparison tool) to determine if your site has been hacked.

```
$ drush generate-makefile make-file.make
$ drush make make-file.make -y
```

It is recommended to run all modules you use through the Coder module (6/7/8), but especially any custom built modules and themes. This module can give you suggestions on how to follow the Drupal communities coding standards.

It can also help you identify other coding errors that may affect your site. Particularly when building custom modules the Coder module can help identify unsanitized user input, SQL injection vulnerabilities and Cross Site Request Forgery (CSRF) problems. It is unfortunately quite common for developers to extend Drupal by forking existing projects and not provide enhancements back to the community. Doing this breaks assumptions within the Update module but more importantly makes upgrades much more difficult. Even with a properly documented patch, it is a lot of work to upgrade, patch and re-write a function in a live website.

By contributing the improved code upstream, you can avoid that often painful process. The peer review that comes with contributing your code back to the community is a secondary benefit: your code base will become more robust because more people will understand it. Your bus factor (the number of people who can go missing from a project by either being hit by a bus or winning the lottery) will increase by releasing your code. Publishing the code elsewhere forces you to actually think about what is required. Further, if someone tries to install your code/system, they might notice missing parts or for that matter parts that might be confidential.

5) Administration

Drupal has a very fine grained and customizable permissions model. In its simplest form, users are assigned roles and each role is given permissions to various functions. Take the time to review roles with access to any of Administer filters, Administer users, Administer permissions, Administer content types, Administer site, Administer configuration, Administer views and translate interface. It is useful to review the permissions after upgrades to verify if any new permissions have been added.

Don't use *admin*, *root*, or simple variations of those as your user/1 admin name. They're the first ones that a cracker is going to try, so be a bit more unique. Obscurity isn't the same as security, but no need to give them their first guess when choosing user names. Another good practice with regards to user/1 is to completely disable the account. With the advent of Drupal 7 and Drush, user/1 is not required to administer Drupal websites anymore, and thus can be simply blocked. The account can be re-enabled as needed through Drush or directly in the database.

As with other server user accounts, you will want to restrict who has access to servers. Make sure to delete any test or developer accounts on the production server.

Another good practice concerning administative users within drupal is to automatically disable their account once a certain period of time has passed. Unused accounts are often a prime target for brute-forcing, as their password is most likely not being rotated, and their legitimate owner might not be watching for login attempts. It is also a PCI requirement that inactive administative accounts be locked-out after 90 days of inactivity. A login attempt does not count as activity, whereas a successful login or another active action does. Modules like the User Expire module can help meet that requirement by automatically expiring accounts with specific roles once they reach a certain inactivity limit.

Don't run Drupal without enabling the Update module that comes with core. Drupal core and contributed modules use a structured release process that allows your administrators to be proactively alerted when one of those modules has a security release. Any piece of code is susceptible to a security issue, and having a central repository that a Drupal site can compare against is key to the security paradigm. Aside from the releases that have fixes for known security problems, some modules (or a version of that module) may become unsupported. This is also a security problem, in that you will not receive updates if there are security problems that are identified with the module. The Update module also allows you to get a weekly email if there are security upgrades that need to be applied.

Drupal's input filters are very powerful, but can provide a vulnerability. **Don't enable the PHP filter** which is available in Drupal 7 Core. Installing the Paranoia module can really help enforce this practice. The PHP filter makes debugging more difficult and exposes your site to a greater risk than it is worth. This module has been removed from Drupal 8, but is available as a contributed module. All PHP code should be written to the file system and not stored in the database.

Another input filter that is problematic is Full HTML which should only be granted to administrator roles. Anyone with the Full HTML filter can craft malicious JavaScript and gain full admin access to any website on the same domain as the Drupal website. If needed, you can add some additional tags to the Filtered HTML input format but be cautious.

6) Modules to Consider

There are a lot of Drupal security modules. Depending on your needs you will want to add more or less than those listed here.

AES Encryption (6/7/8)
: Simple and easy to use encryption API.

Automated Logout (6/7/8)
: Provides the ability to log users out after a specified time of inactivity.

Clear Password Field
: Stops forms from pre-populating a password.

CDN
: Provides easy Content Delivery Network integration for Drupal sites.

Drupal Tiny-IDS
: An alternative to a server-based intrusion detection service.

Encrypt
: An API for performing two-way data encryption.

Local Image Input Filter
: Avoids CSRF attacks through external image references.

Login Security (6/7/8)
: Set access control to restrict access to login forms by IP address.

Paranoia
: Limits PHP functionality and other controls.

Password Policy (6/7/8)

Enforces your user password policy.

Password Strength

Provides realistic password strength measurement and server-side enforcement.

Permissions Lock

Provides more fine-grained control over what users with the permission ‘administer permissions’ can configure.

HTTP Strict Transport Security

To be used together with Secure Login, to prevent SSL strip attacks. Alternatively, directly enforce it through web-server settings.

Restrict IP

Restrict access to an administrator defined set of IP addresses.

Secure Pages

Manages mixed-mode (HTTPS and HTTP) authenticated sessions for enhanced security (note required core patches).

Secure Login (6/7/8)

Provides secure HTTPS access, without mixed-mode capability.

Secure Permissions

Disables the UI to set/change file permissions.

Security Kit

Hardens various pieces of Drupal.

Security Review (6/7/8)

Produces a quick but useful review of your site’s security configuration.

Session Limit

Limits the number of simultaneous sessions per user.

Settings Audit Log

Logs which users did what, when.

Shield

Protects your non-production environment from being accessed.

Site Audit (7/8)

A site analysis tool that generates reports with actionable best practice recommendations.

Two-factor Authentication (TFA)

Second-factor authentication for Drupal sites.

Username Enumeration Prevention (6/7/8)

Stop brute force attacks from leveraging discoverable usernames.

7) Modules to Avoid on Shared Servers

Many Drupal modules intended to help developers develop code also disclose sensitive information about Drupal and/or the web server, or allow users to perform dangerous operations (e.g.: run arbitrary PHP code or trigger long-running operations that could be used to deny service). These modules can be used to debug locally (and many are essential tools for Drupal developers), but should never be installed on a shared environment (e.g.: a production, staging, or testing server).

To limit the damage a malicious user can do if they gain privileged access to Drupal, it's not sufficient for a development module to be simply disabled: the files that make up the module should be removed from the file-system altogether. Doing so prevents a malicious user from enabling it and gaining more data about the system than they would be able to otherwise. Note that it is difficult to automatically enforce that these modules are not deployed to shared systems: developers need to understand why they should not commit these modules and take care to double-check what they're about to deploy.

Some popular development modules which should not be present on any shared website include:

Delete all

This module allows someone with sufficient privileges to delete all content and users on a site.

Devel (6/7/8)

Besides letting users run arbitrary PHP from any page, Devel can be configured to display backtraces, raw database queries and their results, display raw variables, and disable caching, among other things.

Drupal for Firebug

Drupal for Firebug outputs the contents of most variables, raw database queries and their results, display PHP source code, and can be used to run arbitrary PHP. Furthermore, it does all this by interfacing with browser developer tools, making it difficult to determine if this module is enabled by glancing at the site.

Theme Developer

This module, which depends on the Devel module mentioned earlier, is very useful for determining which theme files / functions are used to output a particular section

of the site, but it displays raw variables and slows down the site significantly.

Trace (6)

This module can be used to display backtraces and raw variables, among other things.

Note that most “normal” modules can be dangerous if a malicious user gains privileged access to Drupal. You should evaluate each new module you install to determine what it does and whether the features it brings are worth the risks. Some modules to take into special consideration are:

Backup and Migrate (6/7/8)

This module allows you to download a copy of the site’s database. If restrictions placed upon you by your hosting provider prevents you from being able to make backups, this module will allow you to do so; but a malicious user with privileged access would be able to download a copy of the whole Drupal database, including usernames, passwords, and depending on your site, access keys to the services you use.

Coder (6/7/8)

This module is very useful for ensuring your code conforms to coding standards but can be used to display the PHP that makes up modules.

8) Drupal Distributions

Drupal distributions provide turnkey installations that have been optimized for specific purposes, generally with a curated selection of modules and settings. There are now two distributions which have been specifically built for security, Guardr and Hardened Drupal. Guardr is built to follow the CIA information security triad: confidentiality, integrity and availability. It is worth watching the evolution of these distributions and installing them from time to time if only to have a comparison of modules and configuration options.

9) Choosing Modules & Themes

There are over 30,000 modules and 2,000 themes that have been contributed on Drupal.org. Unfortunately, not all of these modules are stable and secure enough to install in a production environment. When choosing projects to incorporate into your site consider:

- How many reported installs are there?
- What was the date of the last stable release?

- When was the last code commit to the repository?
- How many open bugs are there vs the total number of bugs?
- Do the maintainers also work on other projects?
- Is the project description useful and include screen-shots?
- What documentation is available?
- Is there a Drupal 8 stable or development release?
- How many maintainers are listed?
- Are translations available?

Note that these are just some issues to consider when choosing modules. Ultimately, having an experienced Drupal developer involved in a project is important when reviewing which projects to adopt.

10) Drupal Updates

Eventually, all software will need an update if it is going to continue to be useful. Most commonly they are feature releases and do not impact security. The available updates report will show you these when the Update manager is enabled. This report will also alert you when there are security updates available on projects that are enabled and hosted on Drupal.org. Core updates tend to be released on the 3rd week of the month.

The risk levels that the Drupal community has adopted are now based on the NIST Common Misuse Scoring System and converted into the following text descriptions: Not Critical, Less Critical, Moderately Critical, Critical and Highly Critical.

Sometimes a maintainer does not have the time to put out a full release, so will produce a development release, or simply post the code to the Git repository on Drupal.org. For now the Update manager does not have a means to track anything other than full stable releases. The Available update report will show you when a new release is available, but is geared to stable releases. If your site uses modules hosted on GitHub or other repositories, you will not have the benefit of the security alerts made by through Drupal.org.

Sometimes a module simply doesn't have an active maintainer or the maintainer is focused on the next major version of the code base. For instance, Drupal 6 is still officially supported, but there are very few maintainers actively addressing issues in this older code base. In these instances, a stable release can be removed because officially nobody is maintaining it. By definition, unmaintained code is a security problem.

Tools like Drop Guard are designed to make this easier for developers to keep track of.

11) The settings.php

After the initial install, make sure that write permission on the `settings.php` file has been removed.

In Drupal 7 you can set the Base URL which can be useful to block some phishing attempts. You can protect your users against HTTP HOST Header attacks by configuring the settings.php file:

```
$base_url = 'http://www.example.com';
```

In Drupal 8, this is now defined in the Trusted hosts pattern:

```
$settings['trusted_host_patterns'] = array('^www\.example\.com$');
```

There should be a salt in the settings.php so that there is some extra random data used when generating strings like one-time login links. This is added by default in Drupal 7 and 8, but is stored in the the settings.php file. You can store this value outside of the web document root for extra security:

In Drupal 7:

```
$drupal_hash_salt = file_get_contents('/home/example/salt.txt');
```

and Drupal 8:

```
$settings['hash_salt'] = file_get_contents('/home/example/salt.txt');
```

Drupal 8 has added a $config_directories array which specifies the location of file system directories used for configuration data:

```
On install, "active" and "staging" directories are created for
configuration.
The staging directory is used for configuration imports; the active
directory
is not used by default, since the default storage for active
configuration
is the database rather than the file system (this can be changed; see
"Active configuration settings" below).
```

By default this is done within a randomly-named directory, however for extra security, you can override these locations and put it outside of your document root:

```
$config_directories = array(
   CONFIG_ACTIVE_DIRECTORY => '/some/directory/outside/webroot',
   CONFIG_STAGING_DIRECTORY => '/another/directory/outside/webroot',
);
```

Set the $cookie_domain in settings.php, and if you allow the “www” prefix for your domain then ensure that you don’t use the bare domain.

12) Advantages of Drupal 8

Acquia has provided a great list of 10 Ways Drupal 8 Will Be More Secure some of which are mentioned elsewhere in this document. The use of Twig is a big one as it forces a harder separation between logic and presentation. It’s not terribly uncommon for an inexperienced developer to put a lot of PHP in the theme which introduces a lot of security problems down the line.

Another important security feature is that Drupal 8 has replaced a lot of its custom code with software that was Proudly-Found-Elsewhere which means that there is a broader pool of developers to look at to harden the code. Symfony, CKEditor, Composer, EasyRDF, Guzzle & Doctrine are just some of the examples of other open-source projects that have been incorporated.

The Configuration Management Initiative (CMI) and introduction of YAML files to control configuration will also allow administrators to have greater control of changes that are introduced. Simply the ability to track changes in configuration will help manage more secure, enterprise solutions.

By default in Drupal 8, PHP execution in subfolders is forbidden by the .htaccess file. This is beneficial as it protects against random PHP files from being executed deep within sub-folders.

You can set the public file base URL now making it easier to avoid MIME confusion attacks by allowing public files to be more easily stored on another domain or subdomain.

In Drupal 8 Cookie domains do not have www. stripped by default to stop session cookie authorization being provided to subdomains.

The adoption of CKEditor into Core also comes with an improvement in that core text filtering supports limiting the use of images local to the site. This helps prevent Cross-Site Request Forgery (CSRF).

Also mentioned in more detail in the Acquia article mentioned above, Drupal 8 also comes with:

- Hardened user session and session ID handling
- Automated CSRF token protection in route definitions
- PDO MySQL limited to executing single statements
- Clickjacking protection enabled by default
- Core JavaScript API Compatible with Content Security Policy W3C Standard

This is the first time that a cash bounty has been provided in the release cycle for discovering Drupal security issues. This is sure to motivate folks to look for and report issues that may have been overlooked in the process of building Drupal Core.

13) If You Find a Security Problem

The Drupal community takes security issues very seriously. If you do see something you think might be a security problem, there is a *full explanation* of what to do. The community needs to have these issues reported so that they can be fixed. For those who are more visual, there is a great infographic here describing the process of fixing security issues in Drupal projects.

14) Miscellaneous

Review the discussion in Section K and decide if you are going to remove the `CHANGELOG.txt` file. Ensure that you can keep up security upgrades on a weekly basis and **do not hack core**! If you plan to distribute your live site so that you can do testing or development outside of a controlled environment, consider building a sanitized version of the database. This is especially important if you have user information stored in the database. If absolutely all information on the site is public, this may not be necessary.

I) Writing Secure Code

There are lots of great resources about how to write secure code. The Drupal handbook page is a great place to start as it is focused directly on the best practices defined by this community.

Created by Wilson Joseph from the Noun Project

With Drupal 8, Twig has been adopted as the template engine, partly to make it harder for insecure code to be used in a site's theme. With prior versions of Drupal it is important to remember that the themes all have PHP in them and that this is a potential source of a security breach. It isn't uncommon for themers to put a lot of PHP inside of a theme rather than in a separate module. It's also more difficult to keep up with security releases in themes, as they are by their very nature customized.

OWASP has a good PHP Security Cheat Sheet with information on specific things to watch out for when running PHP applications. They also have a more generic document on Secure Coding and a more specific one on HTML5 Security.

There are also lots of good blogs on writing secure code, like this one, highlighting general approaches to PHP coding.

Code review can be used in order to find flaws in existing code and discover potential bugs. OWASP offer a free book that can guide you on that process. Basically, you can do it manually using the same guidelines as when writing secure code, but you can also use automated tools. These can be installed in a continuous integration system such as Jenkins in order to periodically checks for flaws. Common clarity and code style rules can spot weak code that is not directly tied to security but can be an indicator on code quality, thus giving a hint on security. Rare tools exists to find security related weaknesses in PHP frameworks and PHP such as phpcs-security-audit that even support Drupal, but like many tools they need manual follow up and tend to execute slowly.

Common Drupal secure coding practices are:

- Never trust user input.

  With sanitization functions like check_plain, filter_xss, and filter_xss_admin it is easy in Drupal to clean strings on output. Any variable in a template or HTML output should pass through one of these.

- Protect yourself from SQL Injection through leveraging Drupal's database abstraction layer.
- Use preg_replace_callback rather than preg_replace as the latter can allow matches to be evaluated as PHP code although the `\e` modifier that introduced this flaw is now deprecated in PHP.
- Always use the Drupal functions when possible instead of plain old PHP.
- Drupal comes with jQuery and leverages it extensively. However, do not rely on JavaScript for validation.
- If you are using eval or drupal_eval this is a potential security risk if PHP input provided contains malicious code.

  If you do this, you can add a new permission in your module so that an admin needs to explicitly assign permissions to a user role.
- Same precautions should be taken with functions such as exec, system, fopen, delete and others that execute external applications or interact directly with the file system.

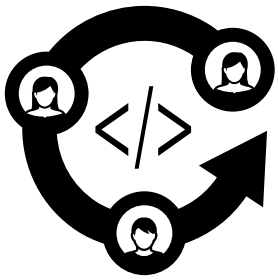
As David Strauss wrote recently, All Code is Debt. "All of the custom code you've written yesterday, rewritten today, and what you'll write tomorrow — you will be burdened with maintaining, forever." Code needs to be maintained on a regular basis to ensure that it is keeping up with the latest security best practices.

When writing code, testing is important and security testing should be part of the process. OWASP publishes a very complete Testing Guide as well as an Application Security Verification Standard that goes deep into details. The verification standard could also be used as a complete security requirement list when designing new modules for your Drupal site. Open-source tools such as OWASP ZAP and Subgraph Vega provide graphical user interfaces to perform dynamic scanning of Web sites. For complex Drupal sites they might have some difficulties but they can still be used as an intercepting web proxy in order to perform manual testing.

There are a lot of resources available to help understand Cross-Site Scripting (XSS) but Acquia has written a nice introduction to XSS

J) Development, Staging and Production

Any formalized development process should have three distinct server environments; Some organizations have more. The development environment can simply be a developer's computer (or perhaps several developers' computers). The staging and production servers should be essentially identical environments. The role of the staging server is to document and test the migration process to verify that the code and configuration can move onto the production server. For more information refer to OpenConcept's blog post on the path of code vs content.



Created by Richard Slater from the Noun Project

The code for your Drupal site should be stored in a central repository. The Drupal community has generally adopted Git, but there are other valid options for version control. A developer will pull/push/clone/branch to/from that repository. New code is committed and pushed from the development environment into the central repository, and can then be pulled onto the staging environment. If it passes testing there, it can then be pulled into production. The database on the staging server can simply be cloned from the production server using Drush. Assuming that the new code works well with the production database, you can be reasonably certain that you will be able to migrate that code and configuration to the production site. This is definitely more complicated, but both the staging and production environments will need to be accessible via Drush and the Git repository.

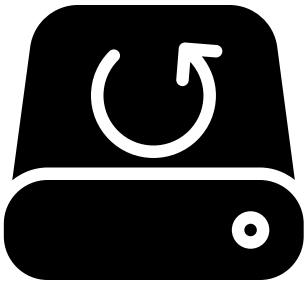
You will need to set up an SSH user with its own SSH keys to allow you to use Drush aliases to transfer databases between staging and production. You may also want to have another account to be able to transfer uploaded files which probably would not be managed under version control. Using an external site like GitHub for storing your repository provides access to some great additional tools like Travis and SauceLabs which can help you deliver a more reliable site. You can also set it up on your staging or development server.

Limit access between servers. There is a potential risk from having a semi-porous boundary between these environments, but the risks are far outweighed by the benefits. Having a central Git repository gives you control across all environments at one time. Being able to diff any change allows you to quickly identify where changes have been made and know why. Drush is certainly powerful, but only experienced users should have access to it. With a solid backup plan, even if this is compromised, it can be

quickly restored.

K) Regular Maintenance

No security plan is foolproof. You need regular backups to ensure that you can restore your system quickly if required. With both the database and file system it is important to have both local and remote backups. You want the local backup because that allows you to quickly restore the site if there is a problem. You want a remote backup in case of total system failure. There are many ways to setup and configure this. Some helpful backup solutions include:



Created by iconsmind.com from the Noun Project

- Bacula
- rsync/rsnapshot
- mysqldump
- xtrabackup

Remember that a backup is only good if it can be restored. It's a best practice to make use of RAID drives, but RAID should be used as a failsafe and not considered a backup strategy. Backups should be stored regularly locally, but there also need to be regular, long-term backups stored off-site. Make sure to evaluate your backup procedures and test your restores to verify that they are working effectively. Drupal.org releases security updates on Wednesdays when needed. These are broadcast through an email list, RSS feeds and Twitter. Subscribe to the security newsletter for updates (you will need a Drupal.org account and the instructions are on the sidebar of the previous link). It is also useful to check the Status page and Watchdog pages in your Drupal site.

SELinux provides auditing services which are worth monitoring. You should be watching your server logs, particularly your Apache error log:

```
$ tail -f /var/log/httpd/error_log
$ grep 'login.php' /var/log/httpd/error_log
$ egrep -i "denied|error|warn" /var/log/httpd/error_log
```

Security best practices are constantly changing. OWASP has released two Top 10 lists about the most critical web application security flaws. There are two descriptions of how the 2013 Top 10 applies to Drupal, the first is from the Drupal Security Whitepaper published in 2014 & the second in a blog post with a *short table view* which is a great summary. There is also an older comparison with the 2010 Top 10 which may also be useful for some users.

This needs to be updated, and reviewed, particularly if you are writing any custom code. It's a simple idea, but it can be good to search Google for test data that might have been left in development or exposed in an upgrade. Anything in a draft mode should never be exposed to the Internet.

Acquia's Insights provides a useful tool to get regular insights on how to improve your website. Their security section will be able to do a quick review of your website to check on a number of security related issues. They also address performance, best practices, SEO and code analysis. With the Acquia Network Connector (6/7/8) module, this can be easily and securely done on any website accessible to the Internet. Their dashboard provides an easy way for you to regularly monitor important elements of your site.

Qualys and Rapid7 both offer a number of other security monitoring and risk assessment services. These are included in Acquia's hosting.

L) Points of Debate

Security by Obscurity

There is a bit of a division within the security community as to whether one should expose information about what versions of software are being used.

Created by Jessica Lock from the Noun Project

Make it Obscure

Leaving a `CHANGELOG.txt` file visible does nothing to improve security, rather it only helps inform an attacker how to focus their research efforts to find a zero day attack, a contrib module vulnerability even faster, or just disable any scripted attacks that aren't relevant to your stack. Justin C. Klein Keane, in his blog Open Source Software Security, strongly recommends hiding both the Drupal and server identification.

Make it Transparent

In many cases where the `CHANGELOG.txt` has been removed, it is because the webmaster hasn't done a Drupal core upgrade and they are looking for a way to obscure that fact. By keeping the `CHANGELOG.txt` up-to-date at the very least it indicates that someone is paying attention to security updates. There are easy ways to fingerprint Drupal and the security team could hide access to this file in the .htaccess file that comes with Drupal core if they were concerned. By making it transparent, there is an additional reason for developers to make it a priority to upgrade core when there is a security release.

Be Consistent

While there is some discussion on the benefits of hiding `CHANGELOG.txt` there is agreement that when security releases are announced, that developers must apply them quickly so that the site cannot be compromised. By default, the Linux distribution, Apache and PHP also announce information that can be turned off in their configuration files. It is good to be consistent and have your reasoning documented so that it is clearly understood.

M) Crackers

There are a lot of reasons why someone might want to gain access to your website. It's possible that they don't like you or your organization. It could be that you are competition or it could even be a disgruntled employee. Some do it just for the challenge.

Unfortunately, there are also governments and corporations who are engaging in this type of activity. It is useful to highlight some of the actors who may want to compromise your site.

Pharma Attacks

One of the most common attacks is simply to increase traffic to a website. This site might include malware on it, but much of the time the goal is simply to increase page views either through actually redirecting your visitors or affecting the target site's Search Engine Optimizaiton.

One of the challenges about this type of activity is that you may not know that your site is compromised for quite some time. You really don't want to find out that your site has been compromised by reading a national newspaper .

Pharma hacks will likely maintain a backdoor that allows the cracker to insert files and modify the database. They may modify the database, add files to your server and modify existing files to insert malicious code.

There are a few descriptions on how to *diagnose and clean a pharma attack*, but if you've followed the guidelines outlined here, you should be fine. There are enough sites which aren't properly secured that it is likely that a cracker will give up and try to compromise another site.

Illegal Botnets

A compromised server could be used as part of a network of computers managed by the controller of the botnet who is then able to direct the activities of these compromised computers for various purposes. Criminals are able to use these servers as part of a lucrative Cyber-arms industry .

Private Information

A compromised server could lead to privacy disclosures revealing non-public information or information intended for another user. The Ashley Madison Data Breach is an example of the type of exposure which can be devistating for all involved. There is a growing list of organizations who have been compromised, if you want to track if your email is included in one of these breaches, sign up to Have I Been Pwned .

Information Disclosure

Sometimes what a cracker is after isn't the information on its own but gives the attacker more information to help find another vulnerability.

N) Regulations

Be aware of the regulations in your environment. Your country & industry may have its own standards. Acquia has compiled a nice list of regulations to to review as well.

Global

PCI - Payment Card Industry Data Security Standard. Make sure to also see the Drupal PCI Compliance Report for more information.

SOC1 & SOC2 - Service Organization Controls

SCADA (supervisory control and data acquisition)

USA

HIPAA - Health Insurance Portability and Accountability Act

FedRAMP (a backronym from Federal Risk and Authorization Management Program)

Federal Information Security Management Act (FISMA) of 2002

Canada

Personal Information Protection and Electronic Documents Act (PIPEDA)

Europe

Directive on Privacy and Electronic Communications

Additional Resources

General Guidelines

Drupal Security

- Standards, security and best practices - Drupal.org wiki
- Writing secure code - Drupal.org wiki
- Securing your site - Drupal.org wiki
- Drupal Security Group Discussion
- Drupal Security Report - Acquia
- Drupal Security - Acquia
- Security: How the world's largest open source CMS combines open and security - Acquia
- Drupal, SSL and Possible Solutions - Acquia
- Drupal Watchdog Magazine - Security Edition

Secure Hosting

- Linux: 25 PHP Security Best Practices For Sys Admins - Nixcraft
- Hardening an SSL server against the NSA - xin.at
- LinuxSecurity.com
- COTS Security Guidance (CSG) (CSG-09\G) Intrusion Prevention System (IPS) - CSEC
- COTS Security Guidance (CSG) (CSG-10\G) Overview of OS Security Features - CSEC
- How to Deploy HTTPS Correctly - EFF.org

Organizational Security

- Communities @ Risk: Targeted Digital Threats Against Civil Society
- Security in a Box - Tactical Technology Collective
- SocialEngineer.org
- Schneier on Security

Videos

- DrupalCon Barcelona: Drupal and Security: what you need to know
- DrupalCon Denver: Building and Securing Government Drupal Sites in the Cloud

- DrupalCon London: Doing Drupal Security Right
- Securing Drupal Sites for Government Agencies - Acquia
- Drupal Videos About Security on Archive.org
- Semantic Forgeries in Drupal's Form API - Greg Knaddison
- Drupal Security by Ben Jeavons - Acquia

Third-party Tools

- Retina Network Security Scanner - beyondtrust.com
- N-stalker Web Application Security Scanner
- Syhunt Web Security Audits
- Greensql Database Security

Books

- Cracking Drupal by Greg Knaddison
- O'Reilly.com's Linux Server Security by Michael D. Bauer
- Hacking Linux Exposed by Bri Hatch and James Lee
- Announcement of New Cyber Security Books published by scitech
- SELinux System Administration by Sven Vermeulen
- OWASP Application Security Guide For CISOs