Drupal Security

A High-Level Perspective on Web Vulnerabilities, Solutions and Tips for Securing Your Drupal Website

Pushpendra Kumar, *Drupal Technical Architect*
Dean Schneble, *Drupal Practice Leader*

Version: 2.5
December 3, 2015

[technology + passion] - risk

TABLE OF CONTENTS

Leading the way to a secure world.
Protecting our customer's brand.

Introduction

This paper provides an overview of the current high-level threats to web applications and how they can be addressed within Drupal. When evaluating Drupal for your use or when planning your Drupal project, stakeholders and technicians can use this document as part of your decision making process. If your Drupal project is already live, this document should also prove helpful for evaluating your project's security needs and for mitigating existing vulnerabilities. While the latest, stable version of Drupal (Drupal 7) is the primary focus of this document, many of the topics covered apply also to Drupal 6, as well as the recently released Drupal 8.

Overview of Drupal Security

Drupal is well-known as a very secure open source CMS and application framework and compares well to other open source CMSs as well as proprietary CMSs. Unlike proprietary tools, which are owned by companies with commercial reputations and certifications at stake, community driven open source projects, like Drupal, have no incentive to hide security vulnerabilities, whether the vulnerability is known to exist or just a suspected possibility. It's always in the best interest of the community to be up-front regarding all potential security issues. The Drupal Security Team–comprised of respected community volunteers–works to resolve security issues, review code for vulnerabilities, and regularly publishes Security Advisories back to the community. In contrast to this very transparent process, a company with a proprietary tool is often forced to first consider the commercial implications that might surround any potential vulnerability or threat to their application before deciding on their course of action.

We will have a closer look on top 10 Web security vulnerabilities defined by OWASP'13 and review how Drupal guards against those vulnerabilities. We'll also discuss some of the security tips and tricks for keeping your Drupal website secure. Finally, we'll have a look at the top contributed modules available for enhancing Drupal security.

Web Application Threats

The Open Web Application Security Project (OWASP) is a worldwide not-for-profit organization focused on improving the software security with a mission to make software security more transparent, so that individuals and organizations can make informed decisions about true software security risks. They periodically publish the OWASP Top 10, which represents a broad consensus as to what the most critical web application security flaws are. Drupal's API and default configurations are designed to be secure and to mitigate these common security risks.

THE OWASP 2013 TOP 10

1. Injection Flaws

"Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization."

- Query parameters are automatically sanitized to plug possible injection holes via Drupal's robust object-oriented database API.
- Drupal 7's new database API makes use of parameterized queries for built-in SQL injection attack prevention.
- Drupal provides a set of functions to process URLs and SQL arguments, making security an easy choice for developers.
- To prevent the execution of potentially dangerous files, Drupal's file system layer controls where files can be written and alters file extensions of executable files.

2. Broken Authentication

"Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities."

- User credentials are managed on the server side, not in the user's cookie. Passwords are never emailed.
- Authentication cookies are not modifiable by site users.
- The Drupal core manages user account authentication.
- User sessions (and related cookies) are completely destroyed and recreated on login and logout.
- Session cookies are named uniquely for each Drupal installation and strongly restricted by domain, limiting cross-site snooping.

3. Cross Site Scripting

"XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites."

- Drupal has a build-in system for filtering user-generated content for display. By default, untrusted user content is filtered to remove dangerous elements.
- Drupal has wide range of input filters that remove potential XSS exploits from user input.
  — e.g. , *Check_plain(), Check_markup(), Filter_xss_admin()*
- The "Form API" verifies using Form Token that a valid user loaded a form before submitting it. This verification makes effectively mitigates against XSS in Drupal sites.

4. Insecure Direct Object Reference

"A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data."

- Drupal's menu and form APIs encourage validating and sanitizing data submitted from users.
- When object references are passed through the Form API, Drupal core protects the values from tampering by site users.
- Drupal and PHP provide file and session APIs that allow convenient and secure object reference passing.
- Unauthorized requests are prevented by Drupal's permissions and access control system. They can be further obfuscated through configuration and community-contributed code.
- Protection against semantic forgery attacks and additional validation is implemented in the Drupal Form API.

5. Security Misconfiguration

"Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date."

- Drupal.org provides documentation for best practices for securing your configuration.
- There are also a number of contributed modules which can automate the review of your site configuration from a security perspective.

6. Sensitive Data Exposure

"Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser."

- Drupal contributed modules are available to encrypt sensitive data when it's being stored or transmitted.
- Passwords are stored using a one-way hash after salting. Even if an attacker is able to download a site database, recovering usable passwords is extremely difficult.

7. Missing Function Level Access Control

"Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization."

- Drupal uses an integrated URL/access control system.
- Every URL in the system must have access control configured, even if that access is "allow everyone."
- Drupal's powerful permission-based system controls function level access by checking for proper authorization before the action is taken.

8. Cross Site Request Forgery

"A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim."

- The Drupal, the Form API provides protection against CSRF using special tokens in the forms which are added automatically.
- If you discover that you have menu callbacks that are vulnerable to CSRF, the simplest solution may be to add a confirmation form via confirm_form() for each of your menu callbacks.
- drupal_valid_token() provided to generate/validate tokens for GET requests.

9. Using Components with Known Vulnerabilities

"Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts"

- There are several libraries and frameworks in Drupal core but they are system level and not significant risk to server of application vulnerability.

10. Unvalidated Redirects and Forwards

"Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages."

- Drupal's integrated menu and access control system cannot be circumvented with internal page requests.
- Drupal protects against auto redirects to links off-site which are often used in phishing attacks.

Secure Drupal Development

Before beginning your site build, carefully review all of the data your site needs to handle, define the types of users that will access your site, and the user and data interactions you need to support. Then determine –in advance–what your site's security needs are and how you plan to support them. The outcomes of this process should enable you to properly budget and schedule sufficient time in your project for security reviews. Don't underestimate the importance of this step, this upfront investment of time will pay dividends later in the project.

- Enable the Update manager core module - The list of available updates will indicate when new releases are ready for download, and you may configure various options.
- Apply every security patch in a timely manner - after backing up EVERYTHING (code & database).
- Always make regular backups of code, files & database.
- Subscribe to Drupal security notification lists from your Drupal user profile page.
- Have a testing environment ready to evaluate updates for deployment.
- Avoid storing ANYTHING other than the base Drupal install files in the web ROOT.
- Be careful when writing modules or make code changes to themes!
    - Separate/Comment out any changes to code.
    - Don't hack the Drupal core files! If you place custom modifications into your sites Drupal core, you will not be able to easily install Drupal updates and security patches.
- Conduct rigorous code reviews for your own work and all modified contributed modules.
- Don't forget that Drupal is just one important part of your application's overall platform. Apache, PHP, and the rest of your server hosting environment must also be kept up-to-date and secure. These important items are out of the scope of this document, but resources are available at Drupal.org and throughout the internet.

Managing Users and Access

ROLES AND PERMISSIONS

Carefully define the permissions that each type of administrative role will require for those users who need to perform site maintenance. Ensure their roles and permissions are defined distinctly from the roles of your sites non-admin users.

- It's typically more secure to have a greater number of roles defined, each with a narrower set of permissions, than to have fewer roles with broader permissions. You can always give your users multiple roles, and this way you can ensure each user has only the specific permissions they require.
- Only assign the Drupal "Administrator" role to developers or to a Super Admin who can fully understand the implications of the resulting permissions. It's easy, even for an experienced user, to make a critical mistake when they have these permissions.

USER MANAGEMENT

To control how new users can join your site, review the options for user validation and find the right balance of user convenience versus security for your application. If you can establish an authentication tool such as CAS, Shibboleth, OAuth or LDAP, you can often greatly simplify this process for your users and admins. Seriously consider implementing a validation module wherever anonymous users can input content or comments.

- CAPTCHA/reCAPTCHA is a very effective tool for preventing bots or other automated tools from creating spam by requiring a bit of human-like reasoning. The downside is that some users can find the hard-to-read images frustrating.
- Honeypot can also be used as a bot deterrent, and while not quite as effective as CAPTCHA, most users find it much less intrusive.
- Mollom, while more complicated to configure, allows ease of use while still being a very effective tool for preventing comment spam and can allow anonymous users to post comments.

CONTENT INPUT CONTROL

Users want and need sophisticated and powerful tools for adding content to your site, and it's important to preserve all (valid) user input. But at the same, you need to ensure that user input—either maliciously or accidentally – cannot break or breach the security of your site. Meeting this challenge requires carefully managing your sites user input options.

- Enable the Drupal Core Filter module, but carefully configure the text input formats and which specific user roles can use them.
- Implement WYSIWYG Text Editor settings very carefully, limiting the available buttons and features. Use discretion regarding who can use WYSIWIG to add files and images and which file extensions are allowed.
- Do NOT allow tags such as: SCRIPT, IMG, IFRAME, EMBED, OBJECT, INPUT, LINK, STYLE, META, FRAMESET, DIV, SPAN, BASE, TABLE, TR and TD. If possible, determine which specific HTML tags are needed and whitelist only those specific tags your users require.
- Disable the PHP Filter module. This module can cause security and performance issues as it allow users to execute PHP code on your site. There are better alternatives available that do not expose such vulnerabilities on your site.
- Never allow anonymous users to have access to the Full or Filtered HTML text input formats.
- Carefully scrub the Users table when cloning or backporting the database to other environments. Change all passwords or remove users as necessary.
- Always use placeholders in functions such as t(), l(), url(), db_query().
- Create unique names for every field that holds even remotely sensitive info. Why? Because permissions are by FIELD NAME regardless of content type.
    - Example: the field field_user_address, if used on 2 different forms, has the SAME permissions on both forms.
- Consider the types of permissions for fields and content types:
    - Create (Can be given to authenticated users)
    - edit OWN; view OWN (might be safe for most)
    - edit ANY; view ANY (editors or admins only)
    - delete OWN; delete ANY (be careful, admins only)

CONTRIBUTED MODULES

Carefully manage the use of contributed Drupal modules. While most popular contributed modules and very safe and secure, new modules are made available every day in which issues can arise. Historically, there are many more Security Advisories for contributed modules than for Drupal core modules.

- When evaluating a contributed module, thoroughly review the criteria below. Properly evaluating each module in advance will save you time and trouble.
  - — Is there a well-supported version to match your Drupal installation?
  - — Does the maintainer have a solid reputation?
  - — What is the total usage/downloads?
  - — Are there many of open issues? Will they affect your functionality?
  - — Is the usage changing over time? Are people no longer using it?
- If you have an existing site, disable or un-install modules you are not using (UI & Devel modules, like Masquerade). Regularly audit your site for unused modules.
- Never install non-recommended modules or libraries or themes. If you see a warning on Drupal.org, that component should not be trusted.

OTHER USEFUL SECURITY TIPS

- Disable the Tracker module as unauthorized users can often see the activity of others. If you need this type of functionality in your application, it's probably best to create a secure alternative solution.
- Turn off the display of error and warning messages on your production site. These messages can give a potential attacker clues as to where your site might be vulnerable.
- Keep the file system as private (admin/config/media/file-system) and always manage exceptions very tightly.
- Enable Syslog instead of default Database.
  - — By moving the log out of DB, we make sure that hacker is not able to access/delete logs by manipulating the DB even if Drupal site is compromised.
  - — The options to maintain or report logs with the OS are much larger than the default Drupal database logging system.
- If available, use HTTPS instead of HTTP.

Top Drupal Security Modules

Security Review

- The Security Review module automates testing for many of the easy-to-make mistakes that render your site insecure.

Login Security

- Login Security module improves the security options in the login operation of a Drupal site. By default, Drupal introduces only basic access control denying IP access to the full content of the site.
- With Login Security module, a site administrator may protect and restrict access by adding access control features to the login forms (default login form in /user and the block called "login form block").

Update Manager

- This module automatically monitors for new versions of the Drupal software and the contributed modules and themes. You can monitor the log to see what updates are available or to set notifications.

CAPTCHA/reCAPTCHA

- The purpose of CAPTCHA is to block form submissions by spambots, which are automated scripts that post spam content everywhere they can. The CAPTCHA module provides this feature to virtually any user facing web form on a Drupal site.

Content Access

- This module allows you to manage permissions for content types by role and author. It allows you to specify custom view, edit and delete permissions for each content type. Optionally you can enable per content access settings, so you can customize the access for each content node.

ACL

- The ACL module, short for Access Control Lists, is an API for other modules to create lists of users and give them access to nodes. It has no UI of its own and will not do anything by itself; install this module only if some other module tells you to.

SpamSpan filter

- The SpamSpan module obfuscates email addresses to help prevent spambots from collecting them. It implements the technique at the SpamSpan website.

OAuth (Open Authentication)

- This module implements the OAuth 1.0 standard for use with Drupal and acts as a support module for other modules that wish to use OAuth.

Password Policy

- This module provides a way to enforce restrictions on user passwords by defining password policies.

Security Kit

- Security Kit provides Drupal with various security-hardening options. This lets your mitigate the risks of exploitation of different web application vulnerabilities.

Menu Admin Per Menu

- To increase security measures, Drupal allows adding, modifying or deleting menu items only to users having administrator menu permission. If you want some users of the website to manage primary or secondary links, but not the navigation, Menu Admin per Menu comes to your help here.

REFERENCES

https://www.drupal.org/security - Maintained by the Drupal Security Team, this page details all current Drupal security advisories for Drupal Core and contributed modules, as well as security related announcements for things like Drupal security best practices.

https://www.owasp.org - The Open Web Application Security Project (OWASP) is a 501(c)(3) worldwide not-for-profit charitable organization focused on improving the security of software.

https://www.drupal.org

https://api.drupal.org/api/drupal

[technology + passion] - risk

About SDG Corporation

SDG is a leading provider of technology, consulting and risk management solutions to strengthen enterprisebusinesses while managing IT risk. We focus on six practices: Risk and Security; Identity and Access Governance; DigitalCollaboration; Quality Assurance; Mobility and Cloud. In addition we offer a GRC solution, called TruOps, to manageenterprise IT risk and compliance.

For over two decades, SDG has enabled enterprises to realize their dreams by helping them develop, manage and deploy solutions with acceptable risk. We combine technology, thought leadership and a relentless passion for customer success. SDG partners with enterprise brands, but we specificallyfocus on mitigating client IT risk. Our ultimate goal is to help enterprises realize the opportunity of technology, increase innovation, improvespeed-to-market and maximize returns on investment.

SDG Corporation
55 North Water Street
Norwalk, CT 06854

+1 (203) 866 8886

sdgc.com