Drupal Security White Paper

Authors
Benjamin James Jeavons

Gregory James Knaddison

v1.3
11 March 2014

Introduction

This paper provides an analysis of the current state of Drupal security. Decision makers evaluating Drupal for use as a content management system or framework solution are encouraged to use this document in their decision process. The analysis includes historical vulnerability data with respect to mitigation techniques, common and critical security risks, and the community-driven procedures unique to Drupal. The latest, stable version of Drupal (Drupal 7, at the time of this writing) is the focus of the architecture, APIs, and methods discussed in this paper along with analysis of Drupal project Security Advisories from June 2005 to October 2013.

Executive Summary

Drupal is a mature open-source CMS and framework powering hundreds of thousands of sites on the web. Through peer-review and a growing community of driven experts and enthusiasts, Drupal's core systems have been strengthened to mitigate common vulnerabilities. Drupal addresses the critical security risks, including the Top 10 identified by The Open Web Application Security Project (OWASP), with professionally audited methods. Drupal has proven to be a secure and strong solution for enterprise needs.

Backed by the world-wide Drupal community, the Drupal Security Team resolves security issues found in code hosted on drupal.org, including Drupal core and thousands of community-contributed modules and themes. The published Security Advisories disclose vulnerabilities and weaknesses in core and contributed code, separately, and provide mitigation solutions.

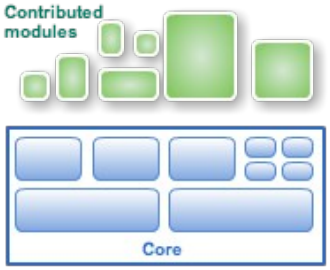
Incorrect use of core APIs and individual site misconfiguration is the cause of most vulnerabilities and weaknesses. Organizations should insist on developer training and employ security testing before moving code and configuration to production. As a process, a secure approach and method must be used during the entire software development life cycle and for all layers of the site and its related infrastructure.

Overview of Drupal and Drupal Security

Drupal is a mature, open-source, PHP-based CMS and web application framework. The Drupal project is composed of two principal bodies of work: (1) the Drupal core codebase, and (2) thousands of contributed modules and themes hosted on drupal.org. Drupal core is a foundational set of APIs and basic features with a small set of optional modules that provide a completely operable web application. Contributed modules and themes extend Drupal core.

Live Drupal installations are rarely composed of only Drupal core; most sites use some amount of contributed modules or have deployed their own custom code. Increasingly, Drupal is being packaged with non-drupal.org code and made available as a separate package. Commonly known as distributions, these projects often, but not always, go by a unique name but tout the fact they are built upon Drupal. Occasionally these projects include modifications to Drupal core for improved performance, extra security, or to fix specific bugs.

Illustration 1: Depiction of separation between core and contributed modules



Drupal is, of course, only one part of the software stack that powers a website and Drupal is commonly deployed on other open-source technologies like Linux and Apache. LAMP (Linux, Apache, MySQL, and PHP) is heavily used on the web and is the common stack components for using Drupal. A good security focus must take into consideration the risks at all layers of the software and hardware stack. Many of the principles expressed in this document, such as running the latest secure release and consistently reviewing for security misconfiguration, can be applied to other parts of your site and infrastructure.

The Drupal Security Team

Comprised of a set of respected community volunteers, and one of the first dedicated Security Teams in an open source CMS project[1], the Drupal Security Team works to resolve reported security issues for code hosted on drupal.org, to review code for vulnerabilities, and to provide security expertise and assistance to contributors. The Drupal Security Team regularly publishes Security Advisories (SA's) which disclose vulnerabilities in Drupal and contributed modules and themes and indicate solutions in the form of patches, updated versions, or mitigation instructions[2]. The distinction between Drupal core, drupal.org contributions, and distributions is an important one to make as it relates to security and the process of the Drupal Security Team. The Drupal Security Team currently publishes three streams of SA's: one for Drupal core, one for contributed projects on drupal.org, and one for “public service announcements” which contains information about threats relevant to Drupal sites.

Drupal’s Security Risks, Process, and History

While Drupal code is quite mature[3] and the community and Security Team make a concerted effort to implement secure code and address risks, vulnerabilities are discovered in stable releases of core and contributed projects. The Security Advisories for Drupal are artifacts that can be studied to better understand Drupal security.

1 http://buytaert.net/drupal-security-team-past-current-and-future
2 See http://drupalsecurityreport.org/infographic-drupal-security-release-process for illustration of SA release process
3 The Drupal project began in 2001 and there has been a dedicated Security Team since 2005

The Drupal Security Team operates on the principle of limited advance disclosure, holding for full, public disclosure until the threat can be eliminated with a fixed version. Vulnerabilities in Drupal core are coordinated with core branch maintainers and vulnerabilities in contributed projects are coordinated with the individual project maintainers. In the case of an unresponsive community maintainer the Team will unpublish vulnerable releases and issue a Security Advisory urging users to disable the module on their sites. Drupal installations not running the vulnerable code are not affected and need not take any action.

Table 1: Number of Security Advisories for Drupal core and contributed projects per year (2013 data to 10/1)

Year	Core	Contributed
2013	*2*	*77*
2012	3	174
2011	3	59
2010	2	31
2009	8	115
2008	11	64

Table 1 depicts the number of Security Advisories for Drupal core and contributed projects per year[4] that contained vulnerabilities of any type. The table illustrates that there are more Security Advisories for community-contributed projects than there are for Drupal core.

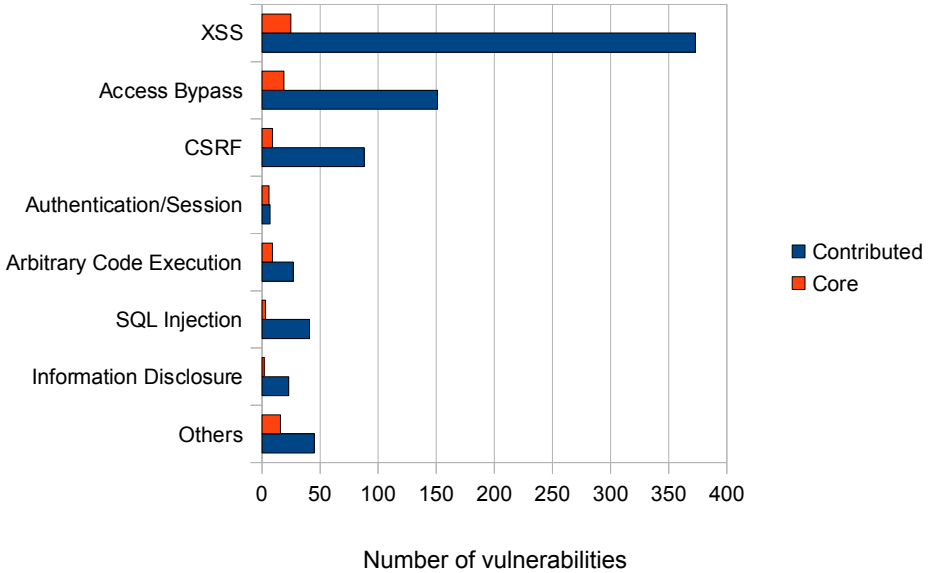


Illustration 2: Number of vulnerabilities by type reported in SAs for Drupal core and contributed projects[5]

4 SAs and vulnerabilities collected and analyzed from June 1, 2005 through October 1, 2013
5 "Others" include HTTP Header injection, Phishing, Validation Bypass, and more. Vulnerability descriptions are provided in the Appendix.

Of the 647 contributed project Advisories, 373 have reported at least one XSS issue in the vulnerable project. Illustration 2 shows that Cross Site Scripting (XSS) is the most common vulnerability in Drupal code.

The types of vulnerabilities that exist in core are often more obscure than those in contributed code. For example, the XSS vulnerability fixed with the release in SA-CORE-2009-005 was only exploitable in Internet Explorer browsers due to the browser interpreting certain byte sequences as UTF-7. Less obscure vulnerabilities are often mitigated by only being exploitable when the attacker has non-default permissions on a victim's site. For example, the XSS attacks referenced in SA-CORE-2009-009 require the attacker to have administrative permissions which are not granted to all users by default.

Drupal core is heavily reviewed; each piece of code committed must meet strict coding and security standards, among other factors. A final count of contributors mentioned in code commits for Drupal 7 was just under 1,000. Only a few people have commit access to the Drupal 7 codebase branch which helps to enforce a high quality of committed code. Contributed projects often have only one committer and are peer-reviewed far less than core.

Custom code for an individual site is a further source of many exploits. A review by WhiteHat Security of the Drupal site Greenopolis.com found that 90% of the 120 vulnerabilities discovered existed in their custom Drupal theme[6]. Security audits from Acquia, a Drupal services company, often find the majority of security issues are from erroneous configurations along with XSS, CSRF and SQL injection issues in custom code[7].

There have been no widely exploited vulnerabilities in Drupal core for which there was no patch or upgrade available at the time of public disclosure. The closest known instance was an exploit in a common XML-RPC code library that was in use in early versions of Drupal. All vendors using the library were at risk, but no wide attack was known to be in play.

In May of 2013, drupal.org, the main site of the Drupal project, announced[8] that user data on drupal.org and a sub-site had been compromised by way of an insecure non-Drupal third-party extension. Compromised data included usernames, email addresses, country information and hashed passwords of many account holders. Use of a third-party extension as a vector for attack is not uncommon since many websites use various software applications and components for different functions.

The Open Web Application Security Project (OWASP) publishes a list of the most critical security risks facing

6 http://szeged2008.drupalcon.org/files/Hack-proof%20Your%20Drupal%20App.pdf
7 http://www.acquia.com/blog/things-we-found-your-site-part-2
8 https://drupal.org/news/130529SecurityUpdate

organizations on the web. The risks can be used to discuss Drupal security in a standardized manner for comparison to other applications. The OWASP Top 10 Most Critical Web Application Security Risks, from most to least critical, are defined online[9] and addressed in the following section.

How Drupal Addresses Security Risks

Drupal is built upon a rich set of APIs which, when used correctly, mitigate common security risks. This section is about how Drupal addresses security risks including the 2013 OWASP Top 10.

2013 OWASP Top 10

Injection

Drupal contains a robust object-oriented database API that makes it difficult for developers to unknowingly create injection holes by automatically sanitizing query parameters and enforcing an interface. Drupal's file system interaction layer limits where files can be written and alters dangerous file extensions that the server could potentially execute.

Broken Authentication and Session Management

User accounts and authentication are managed by Drupal core. Authentication cookies and a user's name, ID, and password are managed on the server to prevent a user from easily escalating authorization. User passwords are salted and hashed using an algorithm based on the Portable PHP Password Hashing Framework and existing sessions are destroyed upon login and logout.

Cross Site Scripting - XSS

Drupal has a strong system for filtering user-generated content on display. Untrusted user's content is filtered to remove dangerous elements by default. For developers, Drupal has at least eight API functions for filtering output to prevent XSS attacks. When identified, common developer errors that lead to XSS vulnerabilities are mitigated by building safer defaults. For example, a page title function in Drupal 6 is the source of many XSS holes due to a lack of proper escaping. In Drupal 7 this function escapes output by default.

Insecure Direct Object Reference

Drupal often provides direct object reference, such as unique numeric identifiers of user accounts or content available in the URL or form fields. While these identifiers disclose direct system information, Drupal's rich permissions and access control system prevent unauthorized requests. Methods for obfuscation are available through configuration and community-contributed code. Further, validation and protection against semantic forgery attacks is implemented in Drupal core via the Form API.

9 http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Security Misconfiguration

Many critical risks, such as access to administrative site controls, text formats, and private information are restricted to a single admin account by default. Identified design inefficiencies that lead to misconfiguration are corrected often through usability testing and fixes are recommended for inclusion to core. Documentation of best practices for secure configuration and site building are provided for free on drupal.org and there are several contributed projects that conduct automated security review or implement further secure configurations.

Sensitive Data Exposure

Account passwords are salted and repeatedly hashed based on the Portable PHP Password Hashing Framework. Available Drupal community-contributed code offer solutions to encrypt sensitive data at rest or in transit.

Missing Function Level Access Control

Function level access in Drupal is protected by a powerful permission-based system which checks for proper authorization before the action is taken. For the case of URL access, access checking is tightly-integrated into the entire menu-rendering and routing system which means that visibility of navigation links and pages are protected by the same system that handles incoming requests.

Cross Site Request Forgery - CSRF

Drupal validates user intention in actions using industry standard techniques. Typical actions with side-effects (such as actions that delete database objects) are generally conducted with the HTTP POST method. Drupal's Form API implements unique form tokens to protect against CSRF in POST requests. Less important actions can leverage the one-time token generation and validation functions of the Form API while still using an HTTP GET request.

Using Components with Known Vulnerabilities

Included libraries and frameworks (of which there are few) in Drupal core are system-level, unsophisticated, and of low risk to full server or application compromise.

Unvalidated Redirects and Forwards

Internal page redirects cannot be used to circumvent Drupal's integrated menu and access control system. Drupal protects against automatic redirection to off-site URLs which could be used in a phishing attack.

Further Risks and Concerns

Zero-day vulnerabilities

Drupal core has had no major zero-day vulnerability threat. The established process of the Security Team and resolution record mitigates the threat of a vulnerability being disclosed publicly before a fix is available. Responsible vulnerability reporters are credited in all Security Advisories to encourage continued advance disclosure.

Open-source software

Secrecy of source code is not a sustainable security practice. Developers make mistakes and cut corners but the increased visibility of code and emphasis on the individual in open-source communities encourages improved programming skill and practices. With Drupal, the strict requirements before code can be committed to core increases collaboration and peer-review as well as protecting against security holes. Drupal APIs aim to make it easier for plugin code to be more secure than if it were written without Drupal.

Maintaining strong security

A strong security process on a Drupal site involves running the latest secure releases of core and contributed code, maintaining secure configuration, and implementing custom code that uses the established APIs and follows best practices. It is important to remember that security must be maintained on all software and hardware layers as well.

Staying informed of the latest security releases is possible by many means. Advisories are published on drupal.org and accessible as RSS, sent via an email list, and posted on several Twitter accounts. In the administration pages, each Drupal site itself also informs administrators of relevant security releases. The front page of drupal.org always lists the most up-to-date and secure stable release of core.

Educational resources regarding secure configuration and writing secure code are provided on drupal.org and many community sites. Several contributed modules provide security-related services or implement specific security additions[10].

Applying security upgrades

Drupal core updates within a major branch almost exclusively contain security and bug fixes only, so upgrades are usually quick and without fault[11]. In Drupal 7, applying module upgrades can be done within the administrative interface of the website. Contributed projects are not subject to Drupal core's strict policy so upgrade procedures and results are not of as consistently high quality as core. Popular community modules are often better documented and supported.

10 https://drupal.org/node/382752
11 Upgrades are usually quick and without fault if the core codebase is not modified

State of Drupal Security

Many other open source CMS applications do not publicly produce vulnerability disclosures or resolutions to the same degree as the Drupal project. The number of known vulnerabilities in Drupal is sometimes seen as an indication that the code is less secure than software which has fewer published security announcements. It is important to understand that obscurity is not a safe nor sustainable security practice. Lack of public vulnerability data does not indicate that an application lacks vulnerabilities and previous vulnerabilities are not a good indicator of future performance.

The Security Team publishes vulnerability disclosures in the form of Security Advisories. Security Advisories overwhelmingly provide site administrators an immediate path to safety with patches, an upgraded release, or mitigation instructions. In this way, the limited disclosure policy of the Security Team decreases security risks to Drupal sites.

Drupal core is a mature and strong codebase that's been heavily peer-reviewed and professionally audited. The core API tools and techniques, when used correctly, address critical and common security risks. Compared to Drupal core, contributed projects have less contributors, employ less peer-review, and have a greater occurrence of Security Advisories. Audits of Drupal installations have found that the majority of vulnerabilities exist within site-specific configuration and custom code that do not correctly use Drupal's APIs. Lack of peer-review and inadequate understanding and use of the APIs and security best practices is a common cause for vulnerabilities.

It is important to understand that security is a process and not a static product, and that there is always room for improvement. In Drupal, XSS vulnerabilities are too common and security best practices are not well enough understood or communicated. While Drupal 7 has made advances in supplying safer defaults over previous versions, the APIs are not uniform and often non-standardized. Security does not stop at site development and deployment. To assure strong security, diligent methodologies must be applied throughout the entire software lifecycle.

Sponsors

Examiner.com feeds the passion the national and local community has for its favorite interests, activities, and establishments by connecting them with credible and informed contributors who write and share information with passion and insight based on their personal experience and interests.

examiner.com

Acquia is a commercial open source company providing products, training, hosting, services, and technical support for the Drupal social publishing system.

acquia.com

Chapter Three is a professional Drupal services company based in San Francisco, California. For more information visit our website.

chapterthree.com

wunderkraut.com

ustima.com

zivtech.com

Appendix

Additional Reading

Core Security Advisories http://drupal.org/security
Contributed Project Security Advisories http://drupal.org/security/contrib
The Drupal Security Team http://drupal.org/security-team
Secure configuration of your Drupal site http://drupal.org/security/secure-configuration
Writing secure code http://drupal.org/writing-secure-code
Cracking Drupal – The Drupal security book http://crackingdrupal.com/
Drupal Scout – Articles on Drupal security http://drupalscout.com/knowledge-base
This paper's website http://drupalsecurityreport.org
OWASP Top Ten Project http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Web Application Risks and Vulnerability Descriptions

Online sources provide plentiful descriptions of vulnerabilities and risks, of particular highlight is the OWASP Top 10 list at http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project. See also OWASP's Vulnerability list http://www.owasp.org/index.php/Category:Vulnerability and Wikipedia's list http://en.wikipedia.org/wiki/Category:Web_security_exploits.

License