SDG
[technology + passion] – risk

- 55 North Water Street
  Norwalk, CT 06854
- 203.866.8886
- sdgc.com

Leveraging Drupal Website Security to Mitigate Internet Risks

Open source content management systems (CMS) *have grown in popularity and usage, but the security of these systems has been a matter of debate. Drupal has earned the reputation as a secure CMS that can help organizations limit their Internet risk.*

Since its initial release in 2001, Drupal has established itself as a secure, open source content management system (CMS) and application framework. Currently, over one million websites run on Drupal.[1] The platform is designed with robust security that enables users to be proactive in protecting their web information assets, and it stands up well to other open source and proprietary CMSs. However, unlike private organizations that use proprietary technologies to support their web presence, community-driven open source projects such as Drupal have no incentive to hide known or suspected security vulnerabilities. The Drupal security team, which is comprised of respected community volunteers, collaborates closely to resolve security issues, conducts code reviews to identify vulnerabilities, and publishes security advisories to the community regularly.

In contrast to Drupal's transparent process, organizations utilizing proprietary technologies are often forced to first consider the potential negative impact to their brands and financials before deciding on any course of action in the event of a breach.

OWASP's Top 10 Web Application Threats

The Open Web Application Security Project (OWASP) is a worldwide non-profit organization focused on improving software security and making it more transparent. Beginning in 2007, OWASP started publishing approximately every three years the "OWASP Top 10," an awareness document that represents a broad consensus among security experts about the most critical security threats to web applications.

Drupal's API and default configurations are designed to be secure and mitigate the most common security risks. The organization's revised list for 2017[2] identified the following top 10 critical threats to website security:

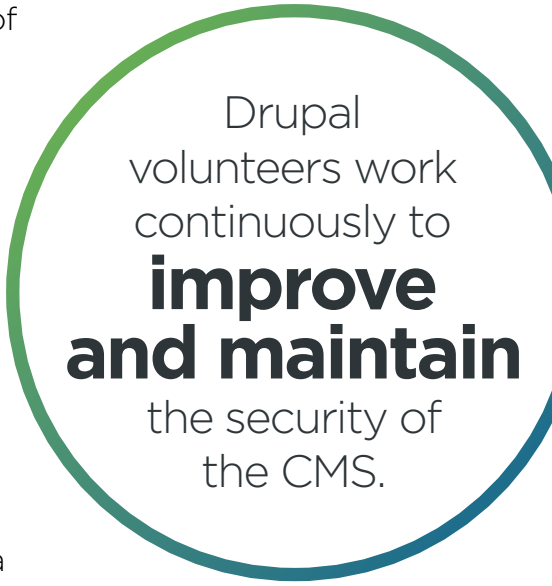
1. **Injection –** Method that attackers exploit to send malicious code through an application to another system
2. **Broken Authentication –** Poor authentication and session management enables attackers to compromise passwords, keys, or session tokens to assume other user identities temporarily or permanently
3. **Sensitive Data Exposure –** Weakly protected sensitive data without the benefit of extra protection, such as encryption at rest or in transit, may be stolen and/or modified by attackers
4. **XML External Entities (XXE) –** Older or poorly configured XML processors evaluate external entity references within XML documents that can be used to disclose internal files using file URI handler, internal file shares, internal port scanning, remote code execution, and DoS attacks
5. **Broken Access Control –** Improperly enforced restrictions for authenticated users enables attackers to gain unauthorized access to functionality and/or data
6. **Security Misconfiguration –** Insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information enables attackers to gain unauthorized access to systems
7. **Cross-Site Scripting –** Occurs whenever an application includes untrusted data in a new web page and enables attackers to execute scripts in the victim's browsers to hijack user sessions, deface websites, or redirect to malicious sites
8. **Insecure Serialization –** Leads to remote code execution and can be used to perform attacks, such as replay attacks, injections attacks, and privilege escalation attacks
9. **Using Components with Known Vulnerabilities –** Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks
10. **Insufficient Logging and Monitoring –** Insufficient logging and monitoring systems enable attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data

Drupal is designed to be **secure and mitigate** the most common security risks.

Drupal Security Countermeasures to OWASP's Top Ten

Drupal is supported by a team of dedicated volunteers that works continuously to improve and maintain the security of the CMS. Drupal's security stands up well against OWASP's Top 10 threats.

- **Injection:** Drupal's API makes use of parameterized queries for built-in SQL injection attack prevention. 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.
- **Broken Authentication:** The Drupal core manages user account authentication, and user sessions (as well as related cookies) are completely destroyed and recreated at login and logout. Sessions cookies receive unique names for each Drupal installation and are strongly restricted by domain to limit cross-site snooping.
- **Sensitive Data Exposure:** Drupal has a module that provides an application programming interface (API for performing symmetric or asymmetric encryption. It enables integrating modules to encrypt and decrypt data in a standardized manner.



- **XML External Entities (XXE):** The Drupal services module supports integrating external applications with Drupal. Service callbacks may be used with multiple interfaces like REST, XMLRPC, JSON, JSON-RPC, SOAP, and AMF.
- **Broken Access Control:** Drupal enables Super users to manage permissions for content types by role and author. Permissions can be viewed, edited, and deleted for each content type and access can be customized for each content node.
- **Security Misconfiguration:** Drupal provides a number of modules that can automate the review of a site's configuration from a security perspective.
- **Cross-Site Scripting:** Drupal has a built-in system for filtering user-generated content for display. By default, untrusted user content is filtered to remove dangerous elements.
- **Insecure Serialization:** Drupal provides a module for serializing and deserializing data to and from formats, including JSON and XML.
- **Using Components with Known Vulnerabilities:** There are several libraries and frameworks in the Drupal core, but they are system level and do increase risk to servers or applications.
- **Insufficient Logging and Monitoring:** Drupal's administrative logging tool provides valuable information for use in system management and security auditing.

Best Practices in Secure Website Development Using Drupal

Before building a Drupal website, content should be carefully reviewed, user types defined, and user and data interactions needing support identified. Security will be a paramount issue for the web development team, who should anticipate the site's security needs and how to support them. The outcomes of this process will enable IT leadership to adequately budget and schedule sufficient time for security reviews. Implementing the following best practices will help ensure the launch of a secure Drupal website. Drupal website developers should always:

- ✔ Enable the Update Manager Core Module to access the list of new releases ready for download
- ✔ Apply every security patch in a timely manner after backing up code and databases
- ✔ Make regular backups of code, files, and databases
- ✔ Subscribe to Drupal security notification lists
- ✔ Have a testing environment ready to evaluate updates for deployment
- ✔ Avoid storing anything other than the base Drupal install files in the web root
- ✔ Exercise caution when writing modules or making code changes to themes
    - Separate/comment on any code changes
    - Do not hack Drupal core files, as this will make it difficult to install Drupal updates and security patches
    - Conduct rigorous code reviews of work, including modified contributed modules

Of course, Drupal is just one important part of an application's overall platform. Apache, PHP, and the remaining components of a server network and hosting environment must also be kept current and secure.

Defining Roles and Permission in Drupal

In the interest of website security, it's important to carefully define the permissions that each type of administrative role will require. Administrative roles and permissions should be defined differently from the roles of non-administrative users. Typically, it is considered more secure to have a greater number of roles defined with each having a narrower set of permissions than to have fewer roles with broader permissions. Users can be given multiple roles, which will help ensure that each user is only granted the specific permission they require. Only developers or Super Admins should be assigned the Drupal Administrator role because these individuals already fully understand the implications of the granted permissions.

Managing Users in Drupal

In order to control how new users can contribute to a Drupal website, the options for user validation should be reviewed to achieve the right balance of user convenience and security for the application. Establishing an authentication tool, such as CAS, Shibboleth, OAuth, or LDAP, will greatly simplify this process for users and admins. In addition, the implementation of a validation module will enable anonymous users to enter content and/or comments.

CAPTCHA/reCAPTCHA is a very effective and popular tool for preventing bots or other automated tools from creating spam by requiring human-like reasoning. The downside is that some users can find it frustrating to interact with images or text they find hard to read. 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, supports ease of use while still being a very effective tool for preventing comment spam and allowing anonymous users to post comments.

Controlling Content Entry in Drupal

Users want and need powerful, sophisticated tools for adding content to websites, and it's important to preserve all valid user input. At the same time, that user input should not be capable of breaching the security of the website, either maliciously or accidentally. This challenge can be met by carefully managing the website's user input options by taking these actions:

- Enabling the Drupal Core Filter module and configuring the text input formats and which specific user roles can use them
- Implementing WYSIWYG text editor settings and limiting available buttons, features, and user and file types
- Avoiding use of SCRIPT, IMG, IFRAME, EMBED, OBJECT, INPUT, LINK, STYLE, META, FRAMESET, DIV, SPAN, BASE, TABLE, TR, and TD tags and whitelisting the specific HTML tags required by users
- Disabling the PHP Filter module, which can cause security and performance issues since it enables users to execute PHP code on websites
- Disallowing anonymous users to have access to full or filtered HTML text input formats
- Scrubbing user tables when cloning or backporting the database to other environments
- Changing passwords and removing users as necessary
- Using placeholders in functions and creating unique names for fields that contain sensitive data
- Determining the types of permissions for fields and content types, such as "create," "edit own," "edit any," "delete own," and "delete any"

Utilizing Modules Contributed to Drupal Wisely

While the most popular contributed modules in Drupal are considered secure, new modules are made available every day and security issues can arise as a result. Historically, there have been more security advisories for contributed modules than for Drupal core modules. Before deployment, a contributed module should be evaluated using the criteria below:

- ✔ Is there a well-supported version to match the Drupal installation?
- ✔ Does the maintainer have a solid reputation?
- ✔ What is the total usage/number of downloads?
- ✔ Are there any open issues, and will they affect functionality?
- ✔ Is usage of the module increasing or decreasing?

It is critical that the website is regularly audited for unused modules, and these should be disabled or uninstalled when identified. Modules, libraries, or themes not recommended by Drupal should never be installed.

Other actions that can be taken to help make a Drupal website more secure include:

- Disabling the Tracker module to prevent unauthorized users from seeing the activity of others
- Turning off the display of error and warning messages on the production site, since these messages can give attackers clues about site vulnerabilities
- Keeping the file system private and managing exceptions tightly
- Enabling Syslog instead of the default database to prevent attackers from accessing/deleting logs
- Utilizing HTTPS instead of HTTP

Conclusion

Drupal is a powerful, proven CMS and application framework capable of withstanding the most critical website vulnerabilities and preventing worst-case scenario breaches. Organizations seeking to deploy a mature and stable CMS can leverage Drupal to mitigate risks. For an in-depth discussion of Drupal's key security modules, visit **www.drupal.org**.

References

1. http://www.drupal.com/showcases

2.https://www.owasp.org/index.php/Top_10-2017_Top_10

[technology + passion] – risk

- 55 North Water Street
  Norwalk, CT 06854
- 203.866.8886
- sdgc.com

ABOUT SDG

With more than 30 years of experience partnering with global enterprises on complex business and IT initiatives, SDG is a trusted provider of advisory, transformation, and managed services. The firm empowers organizations to strengthen cyber resilience by integrating AI into identity, threat, and risk management solutions that protect digital assets and deliver measurable business value. Learn more at www.sdgc.com.

Contact Us: solutions@sdgc.com