

Sitefinity CMS Security and Best Practices

WHITEPAPER

Table of Contents

Abstract / 3

Purpose / 3 Scope / 3 Disclaimer / 3

Executive Summary / 4

Software Assurance Maturity Model (SAMM) / 5

Team / 5

Secure Product Development / 5

Trainings / 7

Release Cycle / 7

Product Architecture / 8

Data Encryption / 8 Authentication Mechanisms / 9 Permissions, Users, Roles / 9 Site Shield / 10 Sanitizing / 10 Audit Trail / 10 Securing Configuration Files / 10 Web Security Module / 11

Top 10 Most Critical Security Risks / 11

Broken Access Control / 12 Cryptographic Failures / 12 Injection / 13 Insecure Design / 14 Security Misconfiguration / 15 Vulnerable and Outdated Components / 15 Identification and Authentication Failures / 16 Software and Data Integrity Failures / 16 Security Logging and Monitoring Failures / 17 Server-Side Request Forgery / 17

Best Practices for Configuring your Sitefinity Deployment / 18

Securing Your Network / 18 Web Server Security / 19 Other Countermeasures / 20

Conclusion / 21

Abstract

This document provides information about the security aspects of Progress® Sitefinity®.

Purpose

To provide software engineers and site administrators with an overview of the available security configurations in Sitefinity.

Scope

The scope of the security aspects this document covers include:

- Progress Sitefinity organization
- Processes
- Product features

Disclaimer

Progress Sitefinity does not protect against poor implementation or misconfiguration of the product. It cannot protect against infrastructure issues. Although Sitefinity may protect your site against some attacks, it cannot protect against each vulnerability type (e.g., social engineering attacks). We strongly recommend providing proper and regular security training for each person who builds and maintains websites using Sitefinity. Refer to the end of the document for links to additional resources, training and certification courses.

Executive Summary

Organizations of all sizes and industries—from government agencies and financial institutions to Fortune 500 companies—rely on Progress Sitefinity to deliver their web presence.

Security is not open to compromise, and this is reflected in the broad spectrum of questions about security the Sitefinity team at Progress receives.

This whitepaper addresses those questions by listing the most common threats organizations face, explaining what they are, what Sitefinity is doing to prevent them and what extra steps are available to make your environment more secure.

The document explains the most important security aspects of Sitefinity:

- The organization's commitment to security and how this affects the secure software development lifecycle, team structure and training.
- Security features of the product.
- OWASP Top 10 The latest list of common threats for any web application and how Sitefinity handles these kinds of threats to ensure the security of your system.
- Best practices for securing your website in a production environment.

Software Assurance Maturity Model (SAMM)

Sitefinity uses OWASP SAMM framework to implement a strategy for software security. This model helps organizations identify the areas in a web content management system that have higher risk, and to focus resources to deliver a secure product that each customer can rely on.

Team

There is a cross-function team at Progress responsible for security: Sitefinity Security Group. The team has engineers from different teams and positions, including:

- Software architects
- Team managers
- Software engineers
- QA engineers
- Support engineers

The team conducts regular meetings where all security-related issues are discussed and acted upon to maintain the high security standards of the product.

Secure Product Development

Process

Each release of Sitefinity goes through several phases—from planning, design and implementation to testing and maintaining the released version. Security is an important part of the entire system development lifecycle. There are security review procedures in the design and implementation phases that include highly qualified security experts who check for security vulnerabilities. All external libraries integrated in the product are regularly checked. Testing of a new feature includes specific security scans (static and dynamic) to prevent vulnerabilities in the product. During the maintenance phase of the product, security-related reports are considered and handled with the highest priority.

Standards, Certificates and Compliance



SOC 2

Sitefinity is certified by an independent service auditor to comply with the Service Organization Control Standards (SOC 2) developed by the Association of International Certified Professional Accountants (AICPA).

The Progress SOC 2 certification report covers the following areas of internal controls:



Security – helps protect against unauthorized access, use or modification

Availability – ensures service is available for operation and use as committed or agreed upon



Confidentiality - ensures confidential information is well protected

HIPAA

<u>Progress Sitefinity Cloud</u> is also certified by an independent service auditor to comply with the Health Insurance Portability and Accountability Act (HIPAA). The <u>HIPAA Security</u> <u>Complianc</u>e Assessment addressed Administrative, Physical and Technical Safeguards, with a primary focus on the Security Final Rules as they relate to Progress' Web Content Management services. More specifically:

- Administrative Safeguards security includes the overall security management process; assigned security responsibility, workforce security, information access management, security awareness and training, security incident procedures, contingency planning, evaluation and business associate contracts or other arrangements.
- **Physical Safeguards** security includes facility access controls, workstation use, workstation security, as well as portable devices and media controls.
- **Technical Safeguards** security includes audit controls, integrity, personnel or entity authentication and transmission security.
- Organizational Requirements implement BAA contracts that comply with the security measures.

Policies, Procedures and Documentation Requirements implement reasonable and appropriate policies and procedures to comply with standards, implementation specifications or other requirements. Compliance with SOC 2 and HIPAA is a testament that Progress has established a comprehensive set of internal procedures and controls to ensure the security, processing integrity, confidentiality and availability of software development infrastructure. This increases the confidence that organizations have when choosing to rely on Progress services and products for their business.



GDPR

Progress also operates a GDPR Office that conducts a range of activities that address GDPR regulatory requirements. Administrative, technical and operational capabilities are in place that can assist customers with GDPR questions, needs and requirements.

Trainings

Each year, engineers on the Progress Sitefinity team participate in at least two different platforms for security training—Veracode and KnowBe4.

Some of the security experts on the team take additional SANS training. There are also internal training, knowledge sharing sessions and events to keep knowledge in software security up to date.

Release Cycle

There are several types of releases of Sitefinity:

- Official releases Two to four times a year (e.g., versions 14.0, 14.1, 14.2). They contain new features and improvements to old features and fixes from previous releases.
 Official releases are extensively tested with diverse types of tests.
- **Patch** regularly released minor improvements and bug fixes. They contain the latest fixes, including, but not limited to, security fixes that cover even low CVSS score issues.

Critical and High security issues are fixed and backported to supported older versions of the product.

For more information, please refer to Progress Sitefinity Lifecycle Policy.



Product Architecture

Sitefinity has many protection layers to ensure data integrity, confidentiality and availability. Appropriate checks are executed on different layers to prevent security attacks on the system.

In the following diagram, you will find the different tiers and their respective modules.

Clients	Admin App (Angular)Frontend App (.NET Core)Web Client (Browser)Desktop AppMobile AppDevice (loT)	
Web Services	OData REST Admin Frontend Services (Designer, (Pages, Images, (Auto-generated and fully configurable) Metadata) Documents, etc.)	
	Web API MVC	
System Services	Web API MVC Web API MVC Workflow Recycle Bin Publishing Search Comments Statistics Multilingual Module Builder Security Bildignostics Diagnostics Bildignostics Security Multilingual	
⊘ A P O O O O O O O O O O O O O O O O O O	Fluent API Lifecycle Native API (Managers->Providers)	
+) Data	SQL Server, Oracle, etc File System Azure, Amazon S3 LDAP Sitefinity Insight SharePoint Marketo Eloqua Hubspot Elastic Search	

For more information, please refer to Progress Sitefinity Architecture.

Data Encryption

Sensitive data is encrypted with the appropriate algorithms depending on the risk profile, like for example, data at rest or data in transit. Sitefinity offers encryption at application level and database level. Sensitive information is encrypted or hashed. Passwords, for example, are hashed.

Sitefinity is FIPS compliant. Therefore, running Sitefinity on servers that require FIPS compliance is safe, except for the following optional, non-default or external areas:

- POP3 client is not compliant unless the authentication mode is AuthenticationMethod.APOP or AuthenticationMethod.TRYBOTH. AuthenticationMethod.USERPASS
- In the LibrariesConfig, only the default ImageUrlSignatureHashAlgorithm.SHA1 algorithm is compliant for ImageUrlSignatureHashAlgorithm

• Export and import of content and settings in order to transfer them between different Sitefinity instances

Authentication Mechanisms

Sitefinity offers ADFS and several other password-less external authentication mechanisms for SSO (Single Sign-On), as well as password-based authentication for local users. OAuth 2's authorization code, implicit and resource owner password credentials flows are supported.

ADFS and the multiple other external authentication providers based on Microsoft's Katana OWIN are ready to use out-of-the-box and require only basic configuration.

For more information, see Sitefinity's official documentation: Progress Sitefinity Authentication.

Permissions, Users, Roles

Sitefinity comes with Role providers and Membership providers that help manage users in the system and assign them different roles. This helps to configure proper permissions for managing different types of content. The product also has a flexible system for defining granular permissions per item. The constrained or allowed principals are Roles and individual Users. Permissions are applied on different types or items for a variety of operations, including View, Create, Delete, Modify, etc. Depending on the type of object, permissions are verified on the level of different system layers and on different module levels.

API level checks: Permissions for content items, such as news, events, dynamic content items, etc., are verified on the lowest API level (providers).

You can apply filtering by view permissions when reading from the database to prevent unauthorized read access.

Sitemap filters: Pages with restricted access have specific checks to prevent unauthorized access. Sitemap filters are responsible for this protection. For more information about configuring page permissions, see Sitefinity's official documentation: <u>Grant permissions for pages</u>.

Site Shield

You can use the Site shield feature to protect a website that is under development from unauthorized access. You can use it to allow users without backend permissions to view the site while it is under development. For example, when stakeholders want to evaluate e.g. the progress of website project without backend access and permissions. For more information, see Sitefinity's official documentation: <u>Site shield: View unpublished websites</u>.

Sanitizing

HTML sanitization: Sitefinity has an out-of-the-box HTML Sanitizer that prevents dangerous HTML and possible XSS attacks. For more information, see Sitefinity's official documentation: <u>HTML sanitization</u>.

SVG sanitization: The product has a built-in file processor that sanitizes the SVG images on upload. It uses an allowlist to prevent dangerous user input. For more information, see Sitefinity's official documentation: <u>Upload SVG images</u>.

Audit Trail

Enterprise systems that must conform to security standards must have an audit trail that preserves user action logs. Sitefinity provides a module that persists this type of information. For more information, see Sitefinity's official documentation: <u>Audit Trail module</u>.

Securing Configuration Files

In Sitefinity, config files may contain sensitive data, such as credentials to external systems, connection strings, etc., that should not be visible by default. For this purpose, there is an option to encrypt values in the configuration files. Moreover, they could be stored in external key management services. For more information, see Sitefinity's official documentation: Encrypt configurations.

Web Security Module

Sitefinity has an additional layer of protection to your site—the <u>Web security module</u>. It prevents different types of web attacks and can be configured only by the website administrator.

- Security HTTP headers As of Sitefinity 11.0, the system can send HTTP headers to configure web clients (browsers) and turn on their built–in security features. There are various types of attacks that can be prevented, including XSS, clickjacking, code injection, MTM, as well as stealing or modifying data in transit.
- Open redirect protection As of Sitefinity 11.1, the system comes with built-in <u>Open</u> <u>Redirect protection</u> that notifies the user when they are leaving the site and being redirected to an external domain. This can prevent phishing attacks and stealing of end-user data.
- Cross-Site Request Forgery (CSRF) prevention As of Sitefinity 12.1, the Web security module enables IT administrators to configure a centralized mechanism that helps secure the website cookies, preventing CSRF vulnerability. Website administrators can set a minimum-security policy for all website cookies by configuring the SameSite, HttpOnly and Secure attributes.

For more information about the different types of protection and how to configure your site, please see the <u>Web security module official documentation</u>.

Top 10 Most Critical Security Risks

The following list discusses the <u>OWASP Top 10 Application Security Risks for 2021</u> and the actions that Sitefinity took in response to these risks.

In general, any web application can expose ways for attackers to gain unauthorized access or compromise its integrity. Some of those threats have become widely popular and the top ten security risks have been compiled in an extensive list provided by the Open Web Application Security Project (OWASP). The following is a summary of those threats and vulnerabilities in the context of the security Sitefinity offers.

Broken Access Control

Security risk: Restrictions on the actions allowed to authenticated users can often be poorly enforced. Attackers can abuse these flaws to access unauthorized functionality or data, such as access to other users' accounts, sensitive files, modify other users' data, change access rights, etc.

Sitefinity response: Sitefinity checks for authentication permissions for each create, retrieve, update and delete operation. Bypassing security checks is impossible externally through any mechanism—URL, service call or API.

Cryptographic Failures

- Security risk: Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare and PII (Personal Identifiable Information).
 Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data requires special precautions and extra protection, such as encryption at rest or in transit, when exchanged with a browser.
- **Sitefinity response:** Sitefinity stores as little sensitive data as possible and only when it is required for the functionality of the product.

Sitefinity ensures the security of data at rest through a robust cryptographic API that employs high-standard algorithms. This API is utilized for safeguarding all internal sensitive data. For instance, Sitefinity securely stores password hashes by default. Based on the site's configuration, any additional sensitive data can also leverage this API for secure storage.

To protect data in transit, we strongly recommend using an encrypted transport layer security (TLS protocol). You can enforce it by configuring strict transport security header (HSTS) in the Web security module of Sitefinity.

Injection

Security risk: There are several types of injection flaws—SQL, OS Command and LDAP. The attacker's hostile data can trick the interpreter to execute unintended commands or access data without proper authorization. For more information, see <u>OWASP Top 10-2021 A03-Injection</u>.

The most common attack is a SQL injection since the product uses a database to store most of its data. A SQL injection is often used to attack the security of a website by inserting SQL statements in a web form. The main purpose of a SQL injection is to get a badly designed website to perform operations on the database that were not intended by the designer of the system, such as dumping information stored in the database and exposing it to an attacker. An application is vulnerable when data provided by user input can be executed.

Another form of injection attack is cross-site scripting (XSS). XSS flaws occur whenever a) an application includes untrusted data in a new web page without proper validation or b) an application updates an existing web page with usersupplied data, using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the target's browser that can hijack user sessions, deface websites or redirect the user to malicious sites.

Sitefinity response:

 To prevent a SQL injection, applications should provide an API that either avoids the use of the interpreter or exposes an entirely parameterized interface. Sitefinity offers a combination of both methods. It does not execute a single native SQL statement. It calls the underlying provider that manages data access through Data Access ORM (an enterprise level object relational mapping tool). In addition, Data Access internally provides an entirely parameterized interface.

Furthermore, the security API is on the provider level, ensuring that not a single method can be executed without privileges.

- To prevent XSS attacks, several mechanisms are implemented in Sitefinity:
 - Sanitizers. Sitefinity has HTML sanitizers that prevent dangerous content from being rendered in the browser. It uses an allowlist with the permitted HTML tags and attributes. This is the most strict and recommended approach for protection against XSS. For more information, see the official Sitefinity documentation: <u>HTML sanitization</u>.
 - Encoding. All out-of-the-box widgets use the appropriate encoding and sanitization. Depending on the context—HTML, JavaScript or URL—

appropriate encoding is applied to prevent the rendering of potentially dangerous content.

- HTTP security headers. The system sends HTTP headers to configure web clients (browsers) and turn on their built-in security features.
- Content-Security-Policy header. This is one of the most powerful weapons for protection against XSS. To mitigate the risk, a web application can declare that it only expects to send data to and load resources from a specified allowlist of sources and restrict the framing of the website.

Insecure Design

Security risk: Risk factors that contribute to some of the Top 10 risk categories are insecure design and insecure implementation.

Insecure design is a broad category representing different weaknesses, expressed as "missing or ineffective control." It is often the result of a lack of business risk profiling inherent in the software or system being developed and a failure to determine what level of security design is required.

When it comes to implementation, an insecure design cannot be fixed by a perfect implementation as by definition, needed security controls were never created to defend against specific attacks and vice-versa. Even a secure design can still have implementation defects leading to vulnerabilities that may be exploited.

Sitefinity response:

- The Sitefinity software development cycle starts with the architectural vision, negotiating business requirements and threat modeling, and includes regular architecture and security reviews, ensuring the highest quality of the design and the implementation.
- Sitefinity is built according to the defense-in-depth principles, which means having multiple security layers aimed at increasing the security posture of the system.
- Sitefinity is subject to regular static and dynamic security scans as well as composition analysis.

Security Misconfiguration

Security risk: Security misconfiguration is the most common issue. This is usually a result of insecure default configurations, incomplete or ad-hoc configurations, open cloud storage, misconfigured HTTP headers and verbose error messages that contain sensitive information. In addition to securely configuring all operating systems, frameworks, libraries and applications, you must also patch and upgrade them in a timely fashion.

Sitefinity response: While some aspects of security configuration are in the scope of system administrators and not the application itself, Sitefinity provides an easy infrastructure for deploying and applying updates to a secured environment. The system also runs on the latest security features provided by the .NET Framework. Furthermore, to ensure top-line security standards and best practices, the application is run through independent audits: Veracode static and dynamic security scans.

Vulnerable and Outdated Components

Security risk: Components, such as libraries, frameworks and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover.
Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

Sitefinity response: The product is built on top of the .NET Framework and uses many external libraries and services. They are strictly checked for updates and especially for security patches. If such are available, they are applied in the product and a new version of Sitefinity is released.

Identification and Authentication Failures

Security risk: Application functions related to authentication and session management can be implemented incorrectly, allowing attackers to compromise passwords, keys or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.

Sitefinity response:

- Sitefinity provides an extensive set of measures in order to prevent such attacks. The application provides an authentication model that complies with high security standards, for example FIPS compliant algorithms. The default authentication mode is based on OAuth 2.0 and supports ADFS and multiple other SSO providers as well as password-based local authentication.
- The passwords are salted and hashed and then persisted in the database. The default settings in Sitefinity require a minimum of seven characters per password and secure timeout settings. These settings can be overridden to enforce a stricter security and password policy.

Software and Data Integrity Failures

Security risk: Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations. An example of this is an application that relies upon plugins, libraries or modules from untrusted sources, repositories and content delivery networks (CDNs). An insecure CI/CD pipeline can introduce the potential for unauthorized access, malicious code or system compromise. Lastly, many applications now include auto-update functionality, where updates are downloaded without sufficient integrity verification and applied to the previously trusted application. Attackers could potentially upload their own updates to be distributed and run on all installations. Another example of data integrity failure that an attacker can potentially exploit is when objects or data are encoded or serialized into a structure that is vulnerable to insecure deserialization.

Sitefinity response:

- All Sitefinity assemblies are signed to ensure they cannot be altered.
- All Sitefinity additional packages are securely stored in trusted repositories.
- All external dependencies are regularly analyzed with an automated tool.
- All CI/CD pipelines are secured, and the code is reviewed to ensure the integrity of the code and configurations.

 Sitefinity uses Json.NET, JavascriptSerializer and DataContractJsonSerializer. By default, .NET serializers are protected, unless they are configured with non-default settings, or the user controls the deserialized type. There are no such use cases in Sitefinity. The serializers are used securely—with default configuration or by specifying the type that is deserialized.

Security Logging and Monitoring Failures

Security risk: Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems and tamper, extract or destroy data. Most breach studies reveal that the time to detect a breach is over 200 days. Usually, breaches are detected by external parties, rather than through internal processes or monitoring.

Sitefinity response: Sitefinity provides a logging mechanism that is extensible and can be used to persist information in different auditing systems. By default, the product provides error logging and audit trail functionalities that persist their data to the file system or can be sent to Elastic for further analysis and monitoring.

Server-Side Request Forgery

Security risk: SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN or another type of network access control list (ACL). The response from the crafted request can be used to read or update internal server resources. As modern web applications provide end-users with convenient features, fetching a URL becomes a common scenario. As a result, the incidence of SSRF is increasing. Also, the severity of SSRF is becoming higher due to cloud services and the complexity of architectures.

Sitefinity response:

- All user inputs used in server calls are validated and sanitized.
- Any new code is scanned for vulnerabilities with Veracode. In addition, every new feature or bugfix is also checked manually, by regular security reviews. If

Progress

there is newly introduced logic making third-party system requests—all trusted destinations are allowlisted to prevent requests to any arbitrary locations.

Best Practices for Configuring your Sitefinity Deployment

This whitepaper discusses the top ten most common attacks that target web applications and the countermeasures that Sitefinity takes to prevent them. There are a lot of other layers, devices and systems where you would need to enforce security. We recommend being aligned with the latest security best practices. Most of those countermeasures deal with lower-level software and protocols. The list of possible software and network vulnerabilities is long, and any attacker's task is to learn as much as possible about your system, specifics and topology.

The following is a summary of the most common attacks and respective best practices.

Securing Your Network

Network security has various aspects—computer systems, access control, preventing unauthorized information gathering, firewalls, physical security, as well as detection and response to unwanted incursions. The most common attacks that physical networks face are information gathering, sniffing, spoofing and session hijacking. Multiple vulnerabilities enable these kinds of attacks, including exposed ports, services, protocols, poorly encrypted data, weak physical security and the inherently insecure nature of the TCP/IP protocol.

Below is a checklist of best practices that you can follow to build a more solid defense:

- 1. Enforce strong physical security of your network. This is a broad topic and measures could vary from locking machines that are not in use to access cards or biometric access.
- 2. Do not give out custom errors, configuration information and software versions.
- **3.** Apply the latest patches and updates to your OS, routers, switches and firewalls.

- **4**. Disable ports and services that are not used.
- 5. Use firewalls between your DMZ and the public network and between your internal LAN network and your DMZ that mask all internal services.
- 6. Encrypt credentials and application traffic over the network.
- 7. Apply ingress and egress filtering on perimeter routers to prevent spoofing.
- 8. Apply inspection at the firewall. Distributed Denial of Service (DDoS) attacks have become a powerful weapon in any attacker's toolset. While many prevention methods exist, the best way to handle those attacks is at a firewall level.
- 9. Filter broadcast and ICMP requests.
- 10. Apply strong password policies.
- **11.** Centralize logging on all allowed and denied activities and have auditing against unusual patterns in place.

Web Server Security

A secure IIS instance can provide a solid foundation to hosting your Sitefinity application. While there are many considerations, measures and resources on the topic of IIS security, this checklist once again aims to give you a summary of the best practices that help you prevent some of the most common attacks and vulnerabilities. The main threats that your server can face include profiling, unauthorized access, elevation of privileges, viruses and worms, etc.

- Block all unnecessary ports, ICMP traffic and unnecessary protocols and services. This will prevent port scans and ping sweeps that may give out information or locate doors open for attacks.
- **2.** Apply the latest system patches and updates frequently.
- **3.** Use separate application pool identities for each instance of Sitefinity you are hosting.
- **4**. Do not give administrative rights to the application pool identity. It must have access only to the web application files, rather than the entire server.

- 5. Reject URLs with ../ to prevent path traversal.
- 6. Run processes using least privileged accounts.
- 7. Remove unnecessary file shares.
- 8. Disable unused ISAPI filters.
- 9. Properly configure the UrlScan tool, if you are utilizing it.
- 10. Carefully analyze the default and the installed IIS services and disable those that are not needed by the system, as defined by our installation guide. Disable FTP, SMTP and NNTP, unless you require them.
- 11. Install the SQL Server (or any of the other supported databases) on a separate, dedicated, physically secured, patched and updated server. Do not install unnecessary tools and debug symbols on the production server.

Other Countermeasures

Other good practices that should be considered when you deploy Progress Sitefinity:

- 1. Give out as little information about your system as possible. The first step prior to deploying your system is enabling friendly error pages. Abundance of information is one of the greatest weapons in the hands of an attacker.
- 2. Monitor the Sitefinity logs for any unusual patterns and errors.
- **3.** We always recommend upgrading to the newest version of the software, including Sitefinity itself.
- 4. Think about the password policy that you want to apply to both frontend and backend users. Sitefinity enables a wide set of measures including minimal password length, validating against a regular expression and maximal password attempts to make a brute force attack technically impossible. The default restrictions require complex passwords that ensure the security of the solution, and they can also be tailored to fit a very strict security policy. Sitefinity natively supports LDAP and ADFS integration. Therefore, you can avoid maintaining a separate user base and password policy for your CMS and enforce this at a system level in your organization.

5. Sitefinity is designed according to current best practices in security. However, as with any other system, you must be careful with how you extend it. As a powerful framework, Sitefinity allows you to plug any custom functionality into the system. Make sure that you follow the patterns and practices that would make your application secure when developing to avoid Injection, XSS attacks, etc. Utilize the tools and libraries provided by Sitefinity, because they are designed in a secure way and ensure stable implementation throughout your custom functionality.

Conclusion

Security is a sensitive and important subject in today's digital world and an application like Progress Sitefinity can provide a stable platform for your entire online presence. Sitefinity implements several layers that can prevent different types of attacks, which makes it a reliable partner in this never-ending war.

By following the best practices and guidelines for network security and applying measured project-specific and system-specific prevention mechanisms, Sitefinity enables you to have a powerful, scalable and secure solution to power your organization's online presence.

For tips and recommendations on how to configure Sitefinity securely, download our <u>Security hardening guide</u>.

For more information about security, <u>contact us</u> to discuss your specific security needs by filling out our contact form.

If you encounter a security issue, submit a vulnerability report as described in this article.

Learn more about Sitefinity Training and Certification.

Learn More

About Progress

Dedicated to propelling business forward in a technology-driven world, <u>Progress</u> (NASDAQ: PRGS) helps businesses drive faster cycles of innovation, fuel momentum and accelerate their path to success. As the trusted provider of the best products to develop, deploy and manage high-impact applications, Progress enables customers to build the applications and experiences they need, deploy where and how they want and manage it all safely and securely. Hundreds of thousands of enterprises, including 1,700 software companies and 3.5 million developers, depend on Progress to achieve their goals—with confidence. Learn more at <u>www.progress.com</u>

2023 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved. Rev 2023/08 RITM0206309

f /progresssw
 y /progresssw
 progresssw
 in /progress-software
 ⊘ /progress_sw_

Progress*