Progress® Sitefinity™

SITEFINITY SECURITY AND BEST PRACTICES

Table of Contents

Overview

Many large organizations rely on Progress® Sitefinity™ for delivering their web presence—from government agencies, financial institutions and Fortune 500 companies to various businesses all over the world. Security isn't open to compromise and this is reflected in the CMS evaluation process. We often receive a great spectrum of questions about security in Sitefinity. This whitepaper addresses those questions by listing the most common threats that organizations face today, explaining what they are, what Sitefinity is doing to prevent them and what extra steps are available in order to make your environment more secure.

In the first part of this document we will be looking in the most common threats that any web application faces today and how Sitefinity handles these kinds of threats. The Open Web Application Security Project (OWASP) has compiled an extensive list of the top 10 vulnerabilities that Web applications expose today. In the first part of this whitepaper we explain each one of those vulnerabilities and how Sitefinity responds to ensure the security of your system.

The second part contains checklists that cover a larger set of best practices to secure your application and our production environment, and to make them comply with the highest industry standards following best practices.

The Ten Most Critical Web Application Security Risks

Any web application in general can expose many different ways for attackers to get unauthorized access or compromise its integrity. Some of those threats have become widely popular and discussed – the top ten security risks have been compiled in an extensive list provided by the Open Web Application Security Project (OWASP). You can find the full list and analysis in the OWASP Top 10 full report.

Here is a summary of those threats and vulnerabilities in the context of Sitefinity security.

Injection

An **SQL injection** is often used to attack the security of a website by inputting 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 (for example to dump information stored in the database and expose it to an attacker). An application is vulnerable when data provided by user input can be executed.

Sitefinity response

The preferred option for SQL injection prevention is for applications to provide an API that either avoids the use of the interpreter or exposes an entirely parameterized interface. Sitefinity is a combination between those two. Sitefinity does not execute a single Native SQL statement. It calls upon underlying provider that manages data access through OpenAccess ORM – an enterprise level object relational mapping tool. On top of this, OpenAccess 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.

Cross-Site-Scripting (XSS)

Cross Site Scripting or XSS is one of the most common attacks to web applications today. It enables attackers to inject client side script into web pages (usually using an encoded parameter in an URL that they send out to victims). A cross site scripting attack can be used to manipulate server output, store malicious data, get a hold of an authentication cookie or manipulate the DOM. Vulnerabilities can be exposed when there is no proper user input

validation or proper escaping of input, before it's included in the output page. Most simply explained, if your application does not escape > to > this character can be inputted on the pages and used to include a <script> tag and execute any client-side script on the user's browser. From then on it's all up to the attacker's imagination.

Sitefinity response

Both Sitefinity and ASP .NET expose mechanisms to successfully remove such areas vulnerable to XSS attacks. Sitefinity uses the ASP.NET server side and client side validation wherever applicable and escapes untrusted data and symbols. On top of this the Sitefinity authentication cookie is not accessible via client side code. Furthermore, Sitefinity also uses various JavaScript libraries for client side validation.

Broken Authentication and Session Management

This happens when a system exposes vulnerabilities that allow an attacker to guess or get a hold of credentials or a session. An application is vulnerable when credentials are not encrypted or are poorly encrypted, the applications uses weak session IDs, exposes session IDs in the URL through URL rewriting, the passwords are weak and can be guessed, or timeouts are not properly handled allowing a malicious attacker to exploit a session that hasn't timed out.

Sitefinity response

Sitefinity provides an extensive set of measures in order to prevent such attacks. The passwords are stored in an encrypted format. The application provides two authentication models that comply with the highest ASP .NET and Federal Security standards. Authentication is securely done either through an encrypted cookie in the case of Forms authentication, or through a digitally signed identity token issued by a Secured Token Service in the case of Claims Based authentication. The basic settings in Sitefinity require a minimum of 7 characters per password and secure timeout settings. Those settings can be overridden in order to enforce a much more strict security and password policy.

Insecure Direct Object References

This vulnerability allows attackers to get access data that they aren't authorized to view. An application is vulnerable when it doesn't enforce access control checks for all resources at all times.

Sitefinity response

Sitefinity checks for authentication permissions for **each** create, retrieve, update and delete operation. Surpassing security checks is impossible externally through any mechanism – URL, Service Call or API.

Cross-Site Request Forgery

Cross-site request forgery (CSRF), also known as a one-click attack or session riding is where unauthorized commands are transmitted from a user that a website trusts. Unlike cross-site scripting (XSS), which exploits a user's trust for a particular site, CSRF exploits the trust that a site has in a user's browser. A good example of this would be tricking a privileged user of a system to visit a malicious site that includes <img> tags that call handlers for the targeted website through URL. If the website trusts this user, that could lead to unwanted side effects.

Sitefinity response

Sitefinity checks authentication and the referrer header for each request and also utilizes the Claims model of authentication with verified techniques for prevention of CSRF.

Security Misconfiguration

A security misconfiguration is a type of vulnerability in the system setup – outdated OS, outdated framework, unnecessary system features, default account passwords, etc.

Sitefinity response

While some aspects of this are in the scope of system administrators and not the application itself, Sitefinity provides an easy infrastructure for deployment and applying updates to a secured environment. The system also runs on the latest and greatest security features provided by the .NET 4.0 Framework. On top of this, the application is run through independent audits through VeraCode security in order to ensure top-line security standards and best practices.

Insecure Cryptographic Storage

These vulnerabilities would allow unauthorized users to get access to decrypted security information such as data, credit card information or passwords. This happens when the access control is not enforced or the encryption algorithm is weak.

Sitefinity response

Sitefinity doesn't store credit card details by default and relies upon secure external payment providers. It also uses strong standard algorithms to store hashed values of the passwords or digitally sign security tokens. Sitefinity is PCI and FIPS compliant in all areas where user credentials are stored.

Failure to Restrict URL Access

This type of attack occurs in the simple case where no additional access checks or authentication is required to access a page. If a malicious user gets a hold of the URL he can access a restricted area without authenticating.

Sitefinity response

Each object in Sitefinity—Page, Control or Content—is a secured object with an applied permission set. The system ensures an access control check for each operation regardless of how an object is accessed—via URL, API or a RESTful service call.

Insufficient Transport Layer Protection

These types of attacks involve packet sniffing on the transport layer. An attacker can listen to all network traffic and access sensitive information if it is sent in plain text or weakly encrypted. The transport layer security is a much wider topic and there are many precautions that you can take into account. The best way to prevent this on an application level is to provide good encryption mechanisms and wrap all sensitive transactions with SSL.

Sitefinity response

Sitefinity takes multiple steps to protect against these kinds of attacks. SSL can be required not only for the entire site, but on a per-page basis, the 'secure' flag is set on all authentication cookies, the encryption algorithms are Federal Information Processing Standard (FIPS) compliant.

Invalidated Redirects and Forwards

These attacks occur where an application allows redirects and forwards that utilize a parameter for the redirect URL. This URL can point to a malicious site or a copy-cat site that performs phishing, installs malware or tricks the user into providing confidential data.

Sitefinity response

Sitefinity avoids the use of redirects and forwards. All redirects in the system are handled through permissions or validation, thus ensuring an attacker cannot inject a third party site in a parameter and trick a user to redirect to it.

General Best Practices

We've talked about the top 10 common attacks that are directed towards any web application and the top-line countermeasures that Sitefinity takes in order to prevent them. In reality there are a lot of other layers, devices and systems, where you would have to enforce security, and a lot of best practices you should be aware of. Most of those countermeasures have to do with lower level software and protocols. The list of possible software and network vulnerabilities is long and the first and most important task that any attacker would have is to learn as much about your system, specifics and topology as possible. Here we are going to look at common attacks and best practices.

Securing Your Network

Network Security has many different aspects – computer systems, access control, preventing unauthorized information gathering, firewalls, physical security, 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 kind of attacks, including exposed ports, services, protocols, poorly encrypted data, weak physical security and the inherently insecure nature of the TCP/IP protocol. Here is a checklist of best practices that you can follow in order to build a more solid defense:

- Enforce strong physical security of your network – this is in general a wide topic and measures could vary from locking machines that are not in use, to access cards or biometric access
- Do not give out custom errors, configuration information and software versions
- Apply the latest patches and updates to your OS, routers, switches and firewalls
- Disable ports and services that are not used
- Use firewalls between your DMZ and the public network and between your internal LAN network and your DMZ that mask all internal services
- Encrypt credentials and application traffic over the network

- Apply ingress and egress filtering on perimeter routers to prevent from spoofing
- Apply stateful inspection at the firewall. Distributed Denial of Service (DDoS) attacks have become a powerful weapon in any attacker's toolset. While a lot of prevention methods exist the best way to handle those attacks is at a firewall level
- Filter broadcast and ICMP requests
- Apply strong password policies
- Centralize logging on all allowed and denied activities and have auditing against unusual patterns in place

Web Server Security

A secure IIS 6.0+ instance can provide a solid foundation to hosting your Sitefinity application. While there are a lot of considerations, measures and resources on the topic of IIS security, this checklist once again aims to give you a brief 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 exploits
- Apply the latest system patches and updates frequently
- Use separate application pool identities for each instance of Sitefinity you are hosting
- 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
- Reject URLs with '../' to prevent path traversal
- Run processes using least privileged accounts
- Remove unnecessary file shares
- Disable unused ISAPI filters

- Properly configure the UrlScan tool, if you are utilizing it

- Carefully analyze the default and installed IIS Services and disable those that are not need by the system as defined by our installation guide. Disable FTP, SMTP and NNTP, unless you require them

- Define the IP range that is allowed to access Sitefinity's back-end and lock the ~/Sitefinity folder by IP. Since ~/App_Data/Sitefinity is mapped to the /Sitefinity virtual folder you can either change this virtual mapping or transfer all public resources (static CSS, JavaScript and images) to a folder different from this one

- Install SQL Server (or any of the other supported databases) to a separate, dedicated, physically secured, patched and updated server. All unnecessary tools, debug symbols are not installed on the production server

Other Countermeasures

There are other good practices that should be considered whenever you deploy Sitefinity.

- Do not use your server as a workstation

- Give out as little information about your system as possible. In the case of Sitefinity the first step you should do prior to deploying your system is enabling friendly error pages. Attackers love as much information as they can get their hands on

- Monitor the Sitefinity logs for any unusual patterns and errors

- Authentication mode is not required for backwards compatibility, utilize the Claims Based Authentication mode in Sitefinity 5.0 or higher. In a more general sense we always recommend upgrading to the newest version of the software.

- Take a look at our Hosting Recommendations and Setup to review some of the most advanced best practices for infrastructure and deployment on highly secure and highly scalable environments

- Consider the password policy you want to apply to both front end and CMS users. Sitefinity enables a wide set of measures including minimal password length, validating against a regular expression and maximal password attempts in order to make a brute force attack technically impossible. These restrictions are not too limiting by default to ensure ease of use, but they can also be tailored to fit your very strict security policy. Sitefinity natively

supports Windows Authentication and Active Directory 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

- Sitefinity is secure and based on best practices out of the box. But as with any other systems, you have to 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 in order to avoid Injection, XSS attacks etc. Utilize the tools and libraries that Sitefinity provides you with, as they are secure by design, and ensure stable best practice-driven implementation throughout your custom functionality

Conclusion

Security is probably the most sensitive and important subjects in today's digital world and an application like Sitefinity can provide a stable platform for your entire online presence. 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.

Read more about security
Get in touch with us to discuss your specific security needs at sales@sitefinity.com

References

OWASP Top Ten Project
Improving Web Application Security - Threats and Countermeasures

About Progress

Progress (NASDAQ: PRGS) is a global leader in application development, empowering the digital transformation organizations need to create and sustain engaging user experiences in today's evolving marketplace. With offerings spanning web, mobile and data for on-premise and cloud environments, Progress powers startups and industry titans worldwide, promoting success one customer at a time. Learn about Progress at www.progress.com or 1-781-280-4000.

Worldwide Headquarters

Progress, 14 Oak Park Drive, Bedford, MA 01730 USA Tel: +1 781 280-4000 Fax: +1 781 280-4095
On the Web at: www.progress.com
Find us on facebook.com/progresssw twitter.com/progresssw youtube.com/progresssw
For regional international office locations and contact information, please go to www.progress.com/worldwide

Rev 16/12

Progress®