Smarten
Augmented Analytics
Powered by ElegantJ BI

Application Security Implementation

Version 5.1

Document Version: 1.0
Product Version: 5.1 and above

www.Smarten.com

Document Information	
Document ID	Smarten-Application-Security-Implementation
Document Version	5.0
Product Version	5.1 and above
Date	01-Dec-2020
Recipient	NA
Author	EMTPL

Statement of Confidentiality, Disclaimer and Copyright

Disclaimer

Preface

The **Smarten Application Security Implementation** is part of the documentation set for **Smarten Augmented Analytics.**

This document contains information about application security measures integrated within Smarten framework.

> Note:
> Throughout this manual, Smarten Augmented Analytics is abbreviated as Smarten.

Contents

1 Introduction 5

2 Application Security Measures 5

2.1 Cross-Site Scripting 5
2.2 Cross-Site Request Forgery 5
2.3 Unencrypted Login Request 5
2.4 Phishing through URL Redirection 5
2.5 Insecure HTTP method Allowed 6
2.6 Link Injection (facilitates Cross-Site Request Forgery) 6
2.7 Phishing through Frames 6
2.8 Session Identifier Not Updated 6
2.9 Custom Application Error Message 6
2.10 Secure Directories 7
2.11 Admin Section with Authentication 7
2.12 Administrative Directories 7
2.13 Secure HttpOnly 7

3 Product and Support Information 8

1 Introduction

The vulnerabilities of enterprise applications and its security have grown beyond expectations, causing serious damage to the business, brand and reputation with equally undeniable damage to the enterprise. The deployment of an effective and result-oriented secure enterprise application allows enterprises to keep their applications, data and data interchange secure.

This document outlines the Application Security Measures incorporated in Smarten application.

2 Application Security Measures

Following Application Security Measures are incorporated within Smarten framework to ensure enterprise application and data security.

2.1 Cross-Site Scripting

Cross-Site Scripting occurs when dynamically generated web pages display user input, such as login information, that is not properly validated. This allows an attacker to embed malicious scripts into the generated page and then execute the script on the machine of any user who views the site. The user simply visits a page to make the malicious scripts execute. If successful, Cross-Site Scripting vulnerabilities can be exploited to manipulate or steal cookies, create requests that can be mistaken for those of a valid user, compromise confidential information, or execute malicious code on end-user systems.

2.2 Cross-Site Request Forgery

CSRF is an attack which forces an end-user to execute unwanted actions on a web application in which he/she is currently authenticated. With a little help of social engineering (like sending a link via email/chat), an attacker may force the users of a web application to execute actions of the attacker's choice. A successful CSRF attack can compromise end-user data and operation in case of normal user. If the targeted end-user is the administrator account, this can compromise the entire web application.

2.3 Unencrypted Login Request

HTTP clients are often privy to large amounts of personal information (e.g. the user's name, location, mail address, passwords, encryption keys, etc.), and it is important to prevent unintentional leakage of this information via the HTTP protocol to other sources. It is recommended to provide a convenient interface for the user to control dissemination of such information. Errors in this area often create serious security and/or privacy problems.

- Refer to Application Server Security Guide for information related to this configuration.

2.4 Phishing through URL Redirection

URL Redirection Attack is a kind of vulnerability that redirects you to another page freely out of the original website when accessed, usually integrated with a phishing attack. Redirected page could lead to a malicious page that resembles the original, and tries to trick the user into giving their credentials. This can catch some unwary user’s off-guard.

2.5 Insecure HTTP method Allowed

HTTP offers a number of methods viable to perform actions on the web server. Many of these methods are designed to aid developers in deploying and testing HTTP applications. These HTTP methods might be used for nefarious purposes if the web server is misconfigured. It is advisable to disable all methods from server configuration except GET and POST.

- Refer to Application Server Security Guide for information related to this configuration.

2.6 Link Injection (facilitates Cross-Site Request Forgery)

Link Injection is the act of modifying the content of a site by embedding a URL to an external site, or to a script in the vulnerable site. By embedding a URL in the vulnerable site, an attacker is able to use it as a platform to launch attacks against other websites, as well as against the vulnerable site itself.

Some of these possible attacks require the user to be logged in to the site during the attack. By launching these attacks from the vulnerable site, the attacker increases the chances of succeeding in his efforts when the user is more likely to be logged in.

The Link Injection vulnerability is a result of insufficient user input sanitation, which is later returned to the user in the site response. The ability to inject hazardous characters into the response makes it possible for attackers to embed URLs, among other possible content modifications.

2.7 Phishing through Frames

Phishing is a general term for attempts to scam users into surrendering private information which could be used for identity theft. It is possible for an attacker to inject a frame or an iframe tag with malicious content which resembles the attacked site. An incautious user may browse and not realize that he is leaving the original site and surfing a malicious site. The attacker may then lure the user to login again, thus acquiring his login credentials. The fact that the fake site is embedded in the original site helps the attacker by giving his phishing attempts a more reliable appearance.

2.8 Session Identifier Not Updated

Insufficient Session Expiration is when a website permits an attacker to reuse old session credentials or session IDs for authorization. Insufficient Session Expiration increases a website's exposure to attacks that steal or impersonate other users. After a user signs out of the application, the identifiers that were used during the session are supposed to be invalidated. If the server fails to invalidate the session identifiers, it is possible for the attackers to use those identifiers to impersonate the user and perform actions on his behalf.

2.9 Custom Application Error Message

Error messages tend to leak useful information about the application and can possibly make way for attacks. Detailed error reporting should never be provided on a production web application. Error messages give very useful information to an attacker about the application and is usually the first stepping stone to help carry out an attack.

2.10 Secure Directories

Risks associated with an attacker discovering a directory on your application server depend upon what type of directory is discovered, and what types of files are contained within it. The primary threat, other than accessing files containing sensitive information, is that an attacker can utilize the information discovered in that directory to perform other types of attacks. Recommendations include restricting access to important directories or files by adopting a "need to know" requirement for both the document and server root, and turning off features such as Automatic Directory Listings that provide information that could be utilized by an attacker when formulating or conducting an attack.

Some of the directories associated with it are as follows:

Common Web Site Structure Directories
Download/Upload Directories
General Business Directories
Web Application Common Directories

- Refer to Application Server Security Guide for information related to this configuration.

2.11 Admin Section with Authentication

This policy states that any area of the website or web application that contains sensitive information or access to privileged functionality such as site administration requires authentication before allowing access.

2.12 Administrative Directories

The primary danger from an attacker finding a publicly available directory on your web application server depends on what type of directory it is, and what files it contains. Administrative directories typically contain applications capable of changing the configuration of the running software; an attacker who gains access to an administrative application can drastically affect the operation of the website.

- Refer to Application Server Security Guide for information related to this configuration.

2.13 Secure HttpOnly

'HttpOnly' cookie, also known as a **secured cookie**, is a file stored on user's hard drive. It is used for transmitting http or https over the internet where https is a secure protocol and provides a secure transmission of data over your internet connection. This contains a special 'HttpOnly' flag included in the http cookie header that ensures cookies will only be used when transmitting HTTP (or HTTPS) requests.

2.14 Accoutn lockout policy

Admin can set an account lockout policy. The account lockout policy "locks" the user's account after a defined number of failed password attempts. The account lockout prevents the user from logging onto the application for a period of time even if the correct password is entered. Account lockout policy should be enabled to help thwart off those who may attempt to compromise user accounts by brute force methods of guessing username and password combinations.

2.15 Click-jacking

Clickjacking, also known as a “UI redress attack”, is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top level page. Thus, the attacker is “hijacking” clicks meant for their page and routing them to another page, most likely owned by another application, domain, or both.

Using a similar technique, keystrokes can also be hijacked. With a carefully crafted combination of stylesheets, iframes, and text boxes, a user can be led to believe they are typing in the password to their email or bank account, but are instead typing into an invisible frame controlled by the attacker.

3 Product and Support Information

Find more information about Smarten and its features at www.smarten.com
Support: support@smarten.com
Sales: sales@smarten.com
Feedback & Suggestions: support@smarten.com
Support & Knowledgebase Portal: support.smarten.com