Fortify on Demand PCI 2.0 Data Security Standard Compliance

Company:	**BuiltClean_FMA_277038053**
Project:	**ReviewMyElearning**
Version:	**ReviewMyElearning**
Latest Analysis:	**12/8/2014 9:53:51 PM**

Executive Summary

Company:	BuiltClean_FMA_277038053
Project:	ReviewMyElearning
Version:	ReviewMyElearning
Static Analysis Date:	
Dynamic Analysis Date:	12/8/2014 9:53:51 PM

Fortify Security Rating

★★★★☆

14 issues

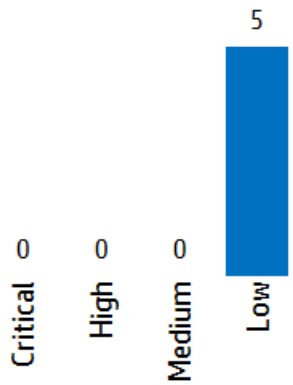
Static:	Dynamic:
✖	✔

Application Details

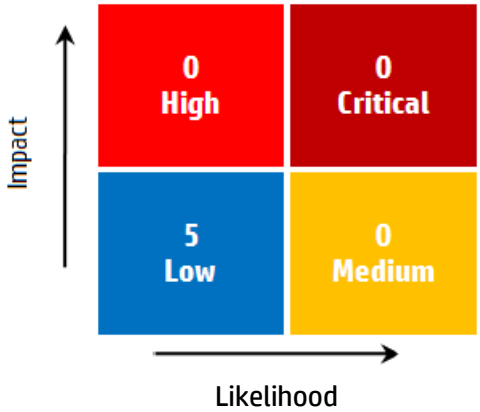
Application Type:	Other Development	Project Type:	Application
Technology Stack:	N/A	Data Classification:	
Interfaces:	Web Access		

PCI DSS 2.0 Compliance Overview

Risk Totals by Severity



Risk Totals by Instance Count



PCI 2.0 Issue Breakdown

Reported issues in the table may violate more than one PCI DSS requirement. As such, the same issue may appear in more than one row. The total number of unique vulnerabilities are reported in the executive summary table.

PCI Requirement Section	Severity				
	Critical	High	Medium	Low	Total

Issue Details

Below is an enumeration of all issues found in the project. The issues are organized by priority and category and then broken down by the package, namespace, or location in which they occur.

The priority of an issue can be Critical, High, Medium, or Low.

Issues from static analysis reported on at same line number with the same category originate from different taint sources.

4.1.1 Form Validation Disabled Low

None

OWASP Top 10: None

PCI 3.0: None

Summary
HTML5 provides a convenient way to add client-side input form field validation by simply including a "required" attribute for required fields or "fieldtype" attribute for common input types or, more granularly, by enabling users to specify a data type-sensitive context that allows pattern-based validation. Developers can supply a customizable "pattern" attribute that checks the input against a regular expression. However, this validation gets disabled when adding a "novalidate" attribute on a form field or a "formnovalidate" attribute on a submit input form field. This check detects whether the "novalidate" or "formnovalidate" attributes are used which may lead to a security vulnerability depending on how securely the application is developed.

Explanation
HTML forms with disabled validation can potentially expose the server to numerous types of attacks. Unchecked input is the root cause of vulnerabilities like cross-site scripting and SQL injection.

Recommendation
While it is much more important to validate input on the server side, validation on the client side adds another layer of protection and makes the process of completing HTML forms more user-friendly and responsive. Avoid using the "novalidate" or "formnovalidate" attributes alone unless compensating methods of validation are being used in their place, and always implement complementary form field validation from the application server; never relying solely on client-side validation routines in the browser.

References

HTML5 <form> novalidate Attribute
http://www.w3schools.com/html5/att_form_novalidate.asp

HTML5 <input> formnovalidate Attribute
http://www.w3schools.com/html5/att_input_formnovalidate.asp

Instances

Form Validation Disabled **Low**

Location
ID 4127384 - https://reviewmyelearning.com:443/

4.1.2 Password Field Auto Complete Active Low

None

OWASP Top 10: None

PCI 3.0: None

Summary
Most recent browsers have features that will save password field content entered by users and then automatically complete password entry the next time the field are encountered. This feature is enabled by default and could leak password since it is stored on the hard drive of the user. The risk of this issue is greatly increased if users are accessing the application from a shared environment. Recommendations include setting autocomplete to "off" on all your password fields.

Explanation

When autocomplete is enabled, hackers can directly steal your password from local storage.

Execution

To verify if a password filed is vulnerable, first make sure to enable the autocomplete in your browser's settings, and then input the other fileds of the form to see whether the password is automatically filled. If yes, then it's vulnerable, otherwise, not. You may need to do it twice in case it is the first time you type in the credential in your browser.

Recommendation

From the web application perspective, the autocomplete can be turned at the form level or individual entry level by defining the attribute AUTOCOMPLETE="off".

References

Microsoft:
Autocomplete Security

Instances

Password Field Auto Complete Active **Low**

Location
ID 4127381 - https://reviewmyelearning.com:443/

4.1.3 Robots.txt Access Control Information Disclosure

Low

None

OWASP Top 10: None

PCI 3.0: None

Summary

Robots.txt is a file that system administrators place on webservers that instructs automated crawling engines such as Google and Altavista to not index or crawl certain portions of the site, usually because they would cause the site's web applications to malfunction, or because they contain sensitive information or applications that should not be displayed in search engine results.

Explanation

An attacker can use the information in the robots.txt file to determine where sensitive or "hidden" application or information is on the site.

Execution

~FullURL~

Recommendation

If robots.txt is not needed, then remove it. If the file is needed, then ensure that it does not contain the locations of hidden or sensitive applications.

References

Hacker Text File
http://packetstormsecurity.nl/docs/hack/robots.txt.advisory

Instances

Robots.txt Access Control Information Disclosure	Low
Location	
ID 4127385 - https://reviewmyelearning.com:443/robots.txt	
ID 4127380 - https://www.reviewmyelearning.com:443/robots.txt	

4.1.4 Server Error Response Low

None

OWASP Top 10: None

PCI 3.0: None

Summary

A server error response was detected. The server could be experiencing errors due to a misbehaving application, a misconfiguration, or a malicious value sent during the auditing process. While error responses in and of themselves are not dangerous, per se, the error responses give attackers insight into how the application handles error conditions. Errors that can be remotely triggered by an attacker can also potentially lead to a denial of service attack or other more severe vulnerability. Recommendations include designing and adding consistent error handling mechanisms which are capable of handling any user input to your web application, providing meaningful detail to end-users, and preventing error messages that might provide information useful to an attacker from being displayed.

Explanation

The server has issued a 500 error response. While the body content of the error page may not expose any information about the technical error, the fact that an error occurred is confirmed by the 500 status code. Knowing whether certain inputs trigger a server error can aid or inform an attacker of potential vulnerabilities.

Recommendation

For Security Operations:

Server error messages, such as "File Protected Against Access", often reveal more information than intended. For instance, an attacker who receives this message can be relatively certain that file exists, which might give him the information he needs to pursue other leads, or to perform an actual exploit. The following recommendations will help to ensure that a potential attacker is not deriving valuable information from any server error message that is presented.

- Uniform Error Codes: Ensure that you are not inadvertently supplying information to an attacker via the use of inconsistent or "conflicting" error messages. For instance, don't reveal unintended information by utilizing error messages such as Access Denied, which will also let an attacker know that the file he seeks actually exists. Have consistent terminology for files and folders that do exist, do not exist, and which have read access denied.
- Informational Error Messages: Ensure that error messages do not reveal too much information. Complete or partial paths, variable and file names, row and column names in tables, and specific database errors should never be revealed to the end user. Remember, an attacker will gather as much information as possible, and then add pieces of seemingly innocuous information together to craft a method of attack.
- Proper Error Handling: Utilize generic error pages and error handling logic to inform end users of potential problems. Do not provide system information or other data that could be utilized by an attacker when orchestrating an attack.

Removing Detailed Error Messages

Find instructions for turning off detailed error messaging in IIS at this link:

http://support.microsoft.com/kb/294807

For Development:

From a development perspective, the best method of preventing problems from arising from server error messages is to adopt secure programming techniques that prevent problems that might arise from an attacker discovering too much information about the architecture and design of your web application. The following recommendations can be used as a basis for that.

- Stringently define the data type (for instance, a string, an alphanumeric character, etc) that the application will accept.
- Use what is good instead of what is bad. Validate input for improper characters.
- Do not display error messages to the end user that provide information (such as table names) that could be utilized in orchestrating an attack.
- Define the allowed set of characters. For instance, if a field is to receive a number, only let that field accept numbers.
- Define the maximum and minimum data lengths for what the application will accept.
- Specify acceptable numeric ranges for input.

For QA:

The best course of action for QA associates to take is to ensure that the error handling scheme is consistent. Do you receive a different type of error for a file that does not exist as opposed to a file that does? Are phrases like "Permission Denied" utilized which could reveal the existence of a

file to an attacker? Inconsistent methods of dealing with errors gives an attacker a very powerful way of gathering information about your web application.

References

Apache:
Security Tips for Server Configuration
Protecting Confidential Documents at Your Site
Securing Apache - Access Control

Microsoft:
How to set required NTFS permissions and user rights for an IIS 5.0 Web server
Default permissions and user rights for IIS 6.0
Description of Microsoft Internet Information Services (IIS) 5.0 and 6.0 status codes

Instances

Server Error Response **Low**

Location
ID 4127379 - https://reviewmyelearning.com:443/?class['classLoader'].resources.context.useHttpOnly=true&class['classLoader'].context.sessionHandler .sessionManager.httpOnly=true&class['classLoader'] .resources.dirContext.aliases=/PRcx^bx^bxgabgdjcfcibfhdcfgggRP=/

4.2.1 Script File Extension Disclosure — Best Practice

None

OWASP Top 10: None

PCI 3.0: None

Summary

HP suggests websites or web applications requiring the use of known script file extensions be checked as it may lead to information disclosure related to the technology used by the application.

Execution

The URL ~FullURL~ uses a known dynamic script extension.

Instances

Script File Extension Disclosure — **Best Practice**

Location
ID 4127383 - https://reviewmyelearning.com:443/admin.pl
ID 4127377 - https://www.reviewmyelearning.com:443/admin.pl

4.3.1 Hidden Form Value Info

None

OWASP Top 10: None

PCI 3.0: None

Summary

While preventing display of information on the web page itself, the information submitted via hidden form fields is easily accessible, and could give an attacker valuable information that would prove helpful in escalating his attack methodology. Recommendations include not relying on hidden form fields as a security solution for any area of the web application that contains sensitive information or access to privileged functionality such as remote site administration functionality.

Explanation

The greatest danger from exploitation of a hidden form field design vulnerability is that the attacker will gain information that will help in orchestrating a far more dangerous attack.

Execution

Any attacker could bypass a hidden form field security solution by viewing the source code of that particular page.

Recommendation

Do not rely on hidden form fields as a method of passing sensitive information or maintaining session state. One workable bypass is to encrypt the hidden values in a form, and then decrypt them when that information is to be utilized by a database operation or a script. From a security standpoint, the best method of temporarily storing information required by different forms is to utilize a session cookie.

Whether hidden or not, if your site utilizes values submitted via a form to construct database queries, do not make the assumption that the data is non-malicious. Instead, utilize the following recommendations to sanitize user supplied input.

- Stringently define the data type (for instance, a string, an alphanumeric character, etc) that the application will accept.
- Use what is good instead of what is bad.
- Validate input for improper characters.
- Do not display error messages to the end user that provide information (such as table names) that could be utilized in orchestrating an attack.
- Define the allowed set of characters. For instance, if a field is to receive a number, only let that field accept numbers.
- Define the maximum and minimum data lengths for what the application will accept.
- Specify acceptable numeric ranges for input.

Instances

Hidden Form Value **Info**

Location
ID 4127386 - https://reviewmyelearning.com:443/

4.3.2 Robots.txt Parser — Info

None

OWASP Top 10: None

PCI 3.0: None

Summary

The Robots.txt parser engine will parse any robots.txt files found within the scan for links to add to the crawler engine.

Instances

Robots.txt Parser — **Info**

Location
ID 4127382 - https://reviewmyelearning.com:443/admin
ID 4127378 - https://www.reviewmyelearning.com:443/admin