



PCI SecurityReview®

Application Security Report for PCI Compliance for CompanyName

April 4, 2008

Application:	Payment Data Manager
Application Type:	Operations Automation / Web Application
Industry Index:	Financial Services
Assurance Level:	AL5 (Very High)
Recommended Analysis:	Static and Dynamic
Type(s) of Analysis Conducted:	Static and Dynamic
Scope of Analysis:	2 of 2 Modules Analyzed

Inside This Report

About this Report	1
Flaws Related to PCI Compliance	1
Immediate Action items for Compliance	1
Summary of Flaws by PCI Requirement	2
Detailed PCI Compliance Flaws by Severity	3
Compliance Information	7
Methodology	9

While every precaution has been taken in the preparation of this document, Veracode, Inc. assumes no responsibility for errors, omissions, or for damages resulting from the use of the information herein. The Veracode SecurityReview uses static and/or dynamic analysis techniques to discover potentially exploitable flaws. Due to the nature of software security testing, the lack of discoverable flaws does not mean the software is 100% secure.

PCI SecurityReview®
Application Security Report for PCI Compliance
 for Company Name

Application Assessed:	Payment Data Manager
PCI Compliance on Tests Performed:	Did Not Pass
Current Rating:	DD
Application Assurance Level:	5 (Very High)
Current Score:	68 (Static Scan) / 75 (Dynamic Scan)
Target Rating:	AA
Date Scanned:	04.04.2008

About this Report

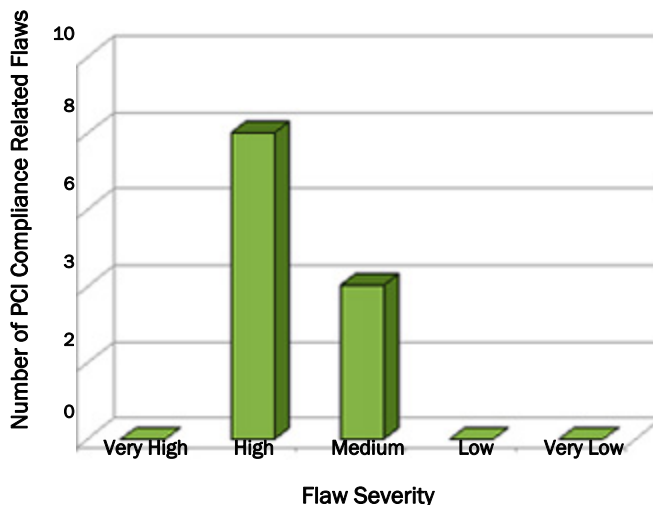
Veracode’s automated static binary and dynamic analysis techniques detect security flaws in applications. For organizations who need to comply with Payment Card Industry (PCI) regulations, this report provides guidance on fixing security flaws toward achieving compliance for PCI Section 6.3.7, 6.5 and 6.6: Develop and maintain secure systems and applications under the “Maintain a Vulnerability Management” Program.

In support of PCI Sections 6.3.7 (Review of custom code prior to release to production or customers in order to identify any potential coding vulnerability) and 6.6 (External review for web applications), Veracode has reviewed the application described in this report and has identified the flaws relevant to PCI Section 6.5 summarized to the right and below and detailed in the body of the report.

Veracode’s review of this application does not consider an exhaustive list of PCI relevant issues. In addition to this analysis, other PCI relevant issues (e.g., security training and incident response process, network security, regular testing, restriction of physical access to cardholder data, etc.) should be reviewed and assessed by a PCI Qualified Security Assessor.

For more information about PCI Regulations and Compliance, please view the Compliance Information section of this report.

Flaws Related to PCI Compliance



Immediate Action Items for Compliance

Veracode recommends the following approaches to achieve PCI Compliance and increase the overall security level of the application.

Fixes Necessary for PCI Compliance

- Fix 8 high severity and 4 medium severity flaws to achieve PCI Compliance for Section 6.5. Fixing these flaws will also increase this application’s Security Quality Score to 90 and achieve an A rating. This will significantly reduce security risks in the application.
- Submit application for follow-up analysis once flaws have been remediated.

Fixes Necessary to Reduce Additional Risk

- Fix 3 high severity and 14 medium flaws. Fixing these flaws will also increase this application’s Security Quality Score to 95 and achieve an A rating. This will significantly reduce security risks in the application.
- Submit application for follow-up analysis once flaws have been remediated.

Additional Recommendations

- Submit application for Dynamic Analysis and/or Manual Penetration Analysis and remediate the required detected flaws to achieve an overall AAA rating.
- Submit application to verify compliance every quarter.
- Certify that software engineers have been trained on application security principals and practices.

Summary of Flaws by PCI Requirement

PCI compliance requires that the application may have no flaws that are categorized in the OWASP Top 10 flaw categories (PCI Requirements 6.5.1 – 6.5.10). Accordingly, the flaws summarized below and listed in detail must be fixed.

PCI Requirement Section and Description		Flaws Detected (By Flaw Category)	
6.5.1	Unvalidated Input Information from web requests is not validated before being used by a web application. Attackers can use these flaws to attack backend components through a web application.	Insufficient Input Valiation	0
6.5.2	Broken Access Control Restrictions on what authenticated users are allowed to do are not properly enforced. Attackers can exploit these flaws to access other users' accounts, view sensitive files, or use unauthorized functions.	Directory Traversal	0
		Encapsulation	0
6.5.3	Broken Authorization and Session Management Account credentials and session tokens are not properly protected. Attackers that can compromise passwords, keys, session cookies, or other tokens can defeat authentication restrictions and assume other users' identities.	Authentication Issues	7
		Credentials Management	2
		Session Fixation	0
6.5.4	Cross Site Scripting The web application can be used as a mechanism to transport an attack to an end user's browser. A successful attack can disclose the end user's session token, attack the local machine, or spoof content to fool the user.	Cross-Site Scripting	0
6.5.5	Buffer Overflows Web application components in some languages that do not properly validate input can be crashed and, in some cases, used to take control of a process. These components can include CGI, libraries, drivers, and web application server components.	Buffer Management Errors	0
		Buffer Overflow	0
6.5.6	Injection Flaws Web applications pass parameters when they access external systems or the local operating system. If an attacker can embed malicious commands in these parameters, the external system may execute those commands on behalf of the web application.	CRLF Injection	0
		OS Command Injection	0
		SQL Injection	0
6.5.7	Improper Error Handling Error conditions that occur during normal operation are not handled properly. If an attacker can cause errors to occur that the web application does not handle, they can gain detailed system information, deny service, cause security mechanisms to fail, or crash the server.	Error Handling	1
		Information Leakage	0
6.5.8	Insecure Storage Web applications frequently use cryptographic functions to protect information and credentials. These functions and the code to integrate them have proven difficult to code properly, frequently resulting in weak protection.	Cryptographic Issues	2
		Credentials Management	0
6.5.9	Application Denial of Service Attackers can consume web application resources to a point where other legitimate users can no longer access or use the application. Attackers can also lock users out of their accounts or even cause the entire application to fail.	Code Quality	0
		Error Handling	0
6.5.10	Insecure Configuration Management Having a strong server configuration standard is critical to a secure web application. These servers have many configuration options that affect security and are not secure out of the box.	Deployment Configuration	0
		Server Configuration	0
6.3.7 / 6.6	Review of Custom Code/External Review for Web Applications Review of all custom application code for common vulnerabilities by an organization that specializes in application security. This requirement summarizes all flaws found in the custom code.	TOTAL FLAWS RELEVANT TO PCI COMPLIANCE	12

Detailed PCI Compliance Related Flaws by Severity

Very High (0 flaws)

No flaws of this type were found.

High (1 flaw)

→ Error Handling *PCI Requirement 6.5.7*

Description

Error handling problems occur when an application does not properly handle errors that occur during processing. If a function does not generate the correct return/status codes, or if the product does not handle all possible return/status codes that could be generated by a function, then security issues may result. Similarly, failing to catch an exception thrown by a function can potentially cause the program to crash or to behave in an unexpected manner.

This type of problem is most often found in edge conditions that are rarely encountered during normal application use. Presumably, most bugs related to common conditions are found and eliminated during development and testing. In some cases, the attacker can directly control or influence the environment to trigger these edge conditions.

Recommendations

Never ignore return codes, assuming that a function will always succeed. Check for and handle all possible return codes to ensure that all scenarios are covered, including boundary or edge conditions. Subject the application to extensive testing to discover some of the possible instances of where and how errors or return values are not handled.

Use a standard exception handling mechanism to be sure that the application properly handles all types of processing errors. Do not allow the application to throw errors up to the application container, generally the web application server.

Associated Software Flaw Types:

→ Unchecked Error Condition (CWE ID 391)

Description

The result of this call are not captured. Failing to check the return code can result in unexpected behavior.

Effort/Complexity of Fix: 2

Recommendations

Check the function return code for success.

Instances

Module	Filename	Line	Reviewer Notes
foo.jar	ScaledFile.java	71	

Attack Vectors

All instances of this flaw were found through static analysis; therefore specific attack vectors are unknown.

Medium (11 flaws)

→ Authentication Issues *PCI Requirement 6.5.3*

Description

Authentication is the process of attempting to verify the digital identity of the sender of a communication such as a request to login to an application. Authentication is a way to ensure users are who they claim to be and that the user who attempts to perform functions in a system is in fact the user who is authorized to do so. When an application does not properly ensure that the user has proven their identity or provides a way to bypass or circumvent the authentication process, the security of the application is compromised.

Detailed PCI Compliance Related Flaws by Severity, cont'd

Medium (11 flaws)

→ **Authentication Issues** *PCI Requirement 6.5.3*

Continued from previous page

Recommendations

When using password systems, implement strong password complexity requirements to evade brute force attacks, and ensure that accounts with default or predictable credentials are removed from production systems. Re-authenticate users for high-value transactions and access to protected areas. Use the most appropriate form of authentication for the assets being protected, e.g. multi-factor authentication for high assurance applications.

Associated Software Flaw Types:

→ **Often Misused (CWE ID 247)**

Description

This function relies on the results of DNS queries. DNS results can easily be forged and should not be trusted, particularly for security-critical functionality such as authentication or authorization. Application design context may be required in order to determine whether it is safe to trust DNS results in this instance.

Effort/Complexity of Fix: 4

Recommendations

Never rely on DNS for security-related decisions. Always sanity check the results of DNS queries to ensure that the data conforms to the expected format.

Instances

Module	Filename	Line	Reviewer Notes
foo.jar	ServerConf.java	162	
foo.jar	ServerConf.java	166	
foo.jar	ServerConf.java	159	
foo.jar	ServerConf.java	149	
foo.jar	ServerConf.java	172	
foo.jar	ServerConf.java	142	
foo.jar	ServerConf.java	145	

Attack Vectors

All instances of this flaw were found through static analysis; therefore specific attack vectors are unknown.

→ **Credentials Management** *PCI Requirement 6.5.3*

Description

Improper management of credentials, such as usernames and passwords, may compromise system security. In particular, storing passwords in plaintext or hard-coding passwords directly into application code are design issues that cannot be easily remedied. Not only does embedding a password allow all of the project's developers to view the password, it also makes fixing the problem extremely difficult. Once the code is in production, the password cannot be changed without patching the software. If a hard-coded password is compromised in a commercial product, all deployed instances may be vulnerable to attack, putting customers at risk.

One variation on hard-coding plaintext passwords is to hard-code a constant string which is the result of a cryptographic one-way hash. For example, instead of storing the word "secret," the application stores an MD5 hash of the word. This is a common mechanism for obscuring hard-coded passwords from casual viewing but does not significantly reduce risk. However, using cryptographic hashes for data stored outside the application code can be an effective practice.

Detailed PCI Compliance Related Flaws by Severity, cont'd

Medium (11 flaws)

→ **Credentials Management** *PCI Requirement 6.5.3*

Continued from previous page

Recommendations

Avoid storing passwords in easily accessible locations, and never store any type of sensitive data in plaintext. Avoid using hard-coded usernames, passwords, or hash constants whenever possible, particularly in relation to security-critical components. Store passwords outof-band from the application code. Follow best practices for protecting credentials stored in alternate locations such as configuration or properties files.

Associated Software Flaw Types:

→ **Hard-Coded Password (CWE ID 259)**

Description

A method uses a hard-coded password that may compromise system security in a way that cannot be easily remedied. The use of a hard-coded password significantly increases the possibility that the account being protected will be compromised. Moreover, the password cannot be changed without patching the software. If a hard-coded password is compromised in a commercial product, all deployed instances may be vulnerable to attack.

Effort/Complexity of Fix: 4

Recommendations

Store passwords out-of-band from the application code. Follow best practices for protecting credentials stored in locations such as configuration or properties files.

Instances

Module	Filename	Line	Reviewer Notes
foo.jar	BarDatabaseManager.java	233	

Attack Vectors

All instances of this flaw were found through static analysis; therefore specific attack vectors are unknown.

→ **Plaintext Storage of a Password (CWE ID 256)**

Description

A method reads and/or stores sensitive information in plaintext, making the data more susceptible to compromise.

Effort/Complexity of Fix: 4

Recommendations

Never store sensitive data in plaintext. Consider using cryptographic hashes as an alternative to plaintext.

Instances

Module	Filename	Line	Reviewer Notes
foo.jar	SqlToolkit.java	173	

Attack Vectors

All instances of this flaw were found through static analysis; therefore specific attack vectors are unknown.

Detailed PCI Compliance Related Flaws by Severity, cont'd

Medium (11 flaws)

→ **Cryptographic Issues** *PCI Requirement 6.5.8*

Description

Applications commonly use cryptography to implement authentication mechanisms and to ensure the confidentiality and integrity of sensitive data, both in transit and at rest. The proper and accurate implementation of cryptography is extremely critical to its efficacy. Configuration or coding mistakes as well as incorrect assumptions may negate a large degree of the protection it affords, leaving the crypto implementation vulnerable to attack.

Common cryptographic mistakes include, but are not limited to, selecting weak keys or weak cipher modes, unintentionally exposing sensitive cryptographic data, using predictable entropy sources, and mismanaging or hard-coding keys.

Developers often make the dangerous assumption that they can improve security by designing their own cryptographic algorithm; however, one of the basic tenets of cryptography is that any cipher whose effectiveness is reliant on the secrecy of the algorithm is fundamentally flawed.

Recommendations

Select the appropriate type of cryptography for the intended purpose. Avoid proprietary encryption algorithms as they typically rely on "security through obscurity" rather than sound mathematics. Select key sizes appropriate for the data being protected; for high assurance applications, 256-bit symmetric keys and 2048-bit asymmetric keys are sufficient. Follow best practices for key storage, and ensure that plaintext data and key material are not inadvertently exposed.

Associated Software Flaw Types:

→ **Insufficient Entropy in PRNG (CWE ID 332)**

Description

Standard random number generators do not provide a sufficient amount of entropy when used for security purposes. Attackers can brute force the output of pseudorandom number generators such as rand().

Effort/Complexity of Fix: 1

Recommendations

If this random number is used where security is a concern, such as generating a session key or session identifier, use a trusted cryptographic random number generator instead. These can be found on the Windows platform in the CryptoAPI or in an open source library such as OpenSSL.

Instances

Module	Filename	Line	Reviewer Notes
foo.jar	DbMgrCommon.java	220	
foo.jar	DbMgrCommon.java	402	

Attack Vectors

All instances of this flaw were found through static analysis; therefore specific attack vectors are unknown.

Low (0 flaws)

No flaws of this type were found.

Very Low (0 flaws)

No flaws of this type were found.

PCI Compliance Information

PCI Overview

The PCI Security Standards Council is an open global forum for the ongoing development, enhancement, storage, dissemination and implementation of security standards for account data protection. The organization was founded by American Express, Discover Financial Services, JCB International, MasterCard Worldwide, and Visa Inc. and is chartered with enhancing payment account data security by fostering broad adoption of the PCI Security Standards.

The PCI Data Security Standard (PCI DSS) represents a common set of industry tools and measurements to help ensure the safe handling of sensitive information which Merchants and Service Providers must meet in order to do business with the major credit card companies. The standard provides an actionable framework for developing a robust account data security process - including preventing, detecting and reacting to security incidents. The PCI DSS consists of 12 overall requirements, organized in 6 logically related groups.

PCI DSS Scope

PCI DSS applies to merchants and service providers who accept credit cards as a form of payment or who process, store, or transmit cardholder data.

Veracode & PCI DSS

Veracode provides independent application security testing to assist merchants and service providers in meeting PCI DSS Requirement 6: Develop and Maintain Secure Systems and Applications. Specifically, Veracode’s Application Security Report can be used to provide evidence of compliance with the following PCI Requirements:

PCI DSS Requirement Number	Description
6.3.7	Review of custom code prior to release to production or customers in order to identify any potential coding vulnerabilities.
6.5.1 to 6.5.10	Develop all web applications based on secure coding guidelines such as the Open Web Application Security Project guidelines. Review custom application code to identify coding vulnerabilities. Cover prevention of common coding vulnerabilities in software development processes, to include the following: Unvalidated input, Broken access control, Broken authentication and session management, Cross-site scripting, Buffer overflows, Injection flaws, Improper error handling, Insecure storage, Denial of service, Insecure configuration management
6.6	Ensure that all web-facing applications are protected against known attacks by applying either of the following methods: Having all custom application code reviewed for common vulnerabilities by an organization that specializes in application security or installing an application layer firewall in front of web-facing applications.

Visa PABP Overview

Visa developed the Payment Application Best Practices (PABP) to assist software vendors in creating secure payment applications that help merchants and agents mitigate compromises, prevent storage of sensitive cardholder data (i.e. full magnetic stripe data, CVV, CVV2 or PIN data) and support overall compliance with the PCI Data Security Standard (PCI DSS).

Visa PABP consists of 14 overall requirements which payment application vendors must meet in order to have their software validated for use by merchants and service providers. The requirements for the PABP are derived from the PCI DSS and are almost identical with regards to application security testing.

Visa PABP Scope

Visa PABP applies to software vendors who develop payment applications that store, process, or transmit cardholder data as part of authorization or settlement. Examples of applicable payment applications include but are not limited to POS software, e-commerce shopping carts, and web-based payment applications.

Veracode & Visa PABP

Veracode provides independent application security testing to assist payment vendors in meeting Visa PABP Requirement 5. Specifically, Veracode’s PCI Application Security Report can also be used to provide evidence of compliance with Visa PABP because the applications security requirements of Visa PABP are equivalent to PCI as described below:

Visa PABP Requirement Number	Description
5.2.7 (equivalent to PCI DSS 6.3.7)	Review of custom code prior to release to production or customers in order to identify any potential coding vulnerabilities.
5.1.1 to 5.1.10 (equivalent to PCI DSS 6.5.1 to 6.5.10)	Develop all web applications based on secure coding guidelines such as the Open Web Application Security Project guidelines. Review custom application code to identify coding vulnerabilities. Cover prevention of common coding vulnerabilities in software development processes, to include the following: Unvalidated input, Broken access control, Broken authentication and session management, Cross-site scripting, Buffer overflows, Injection flaws, Improper error handling, Insecure storage, Denial of service, Insecure configuration management
5.5 (equivalent to PCI DSS 6.6)	Ensure that all web-facing applications are protected against known attacks by applying either of the following methods: Having all custom application code reviewed for common vulnerabilities by an organization that specializes in application or installing an application layer firewall in front of web-facing applications.

Veracode PCI SecurityReview© Summary Report Methodology

Veracode Rating System Using Multiple Analysis Techniques

Higher assurance applications require more comprehensive analysis to accurately score their security quality. Each analysis technique (automated static, automated dynamic, manual penetration testing or manual review) has differing false negative (FN) rates for different types of security flaws. Any single analysis technique or even combination of techniques is bound to produce a certain level of false negatives. Some false negatives are acceptable for lower assurance levels, so a less expensive analysis using only one or two analysis techniques is acceptable. At higher assurance levels the FN rate should be close to zero, so multiple analysis techniques are recommended.

The summary report lists the recommended and performed analysis techniques on page 1. Below is a chart which maps Veracode’s recommended level of analysis to each assurance level.

Application Assurance Level	Rating based on Analysis Score	Automated Static Analysis	Automated Dynamic Analysis	Manual Testing & Design Review	Veracode Recommended Rating
VERY HIGH (AL5)	90 – 100 A 80 – 89 B 70 – 79 C 60 – 69 D	Required	Required	Required	AAA
HIGH (AL4)	90 – 100 A+ 80 – 89 A 70 – 79 B 60 – 69 C 50 – 59 D	Required	Required	Recommended	AAA
MEDIUM (AL3)	80 – 100 A+ 70 – 79 A 60 – 69 B 50 – 59 C 40 – 49 D	Required	Recommended		AA
LOW (AL2)	70 – 100 A+ 60 – 69 A 50 – 59 B 40 – 49 C 30 – 39 D	Recommended			A

Ratings Description

The foundation of the Veracode rating system is the concept that higher assurance applications require higher security quality scores to be acceptable risks. Lower assurance applications can tolerate lower security quality. The assurance level is dictated by the typical deployed environment and the value of data used by the application. Factors that determine assurance level are: reputation damage, financial loss, operational risk, sensitive information disclosure, personal safety, and legal violations.

Assurance Level Definitions

Very High (AL5) – This is typically an application where the safety of life or limb is dependent on the system; it is mission critical the application maintain 100% availability for the long term viability of the project or business. Examples are control software for industrial, transportation or medical equipment or critical business systems such as financial trading systems.

High (AL4) – This is typically an important multi-user business application reachable from the internet and is critical that the application maintain high availability to accomplish its mission. Exploitation of high assurance application cause serious brand damage and business/financial loss and could lead to long term business impact. Exploitation is a result of a breach in any two impact categories of confidentiality, integrity and availability of the application.

Medium (AL3) – This is typically a multi-user application connected to the internet or any system that processes financial or private customer information. Exploitation of medium assurance applications typically result in material business impact resulting in some financial loss, brand damage or business liability. Exploitation is a result of a breach in confidentiality, integrity or availability of the application. An example is a financial services company’s internal 401K management system.

Low (AL2) – This is typically an internal only application that requires low levels of application security such as authentication to protect access to non-critical business information and prevent IT disruptions. Exploitation of low assurance applications may lead to minor levels of inconvenience, distress or IT disruption. An example internal system is a conference room reservation or business card order system.

Very Low (AL1) - Applications that have no material business impact should its confidentiality, data integrity and availability be affected. Code security analysis is not required for this assurance level and security spend should be directed to other higher level assurance applications.

Security Quality Score Description

The Veracode scoring system, Security Quality Score, is built on the foundation of two industry standards, Common Vulnerabilities and Exposures (CVE - <http://cve.mitre.org>) and Common Vulnerability Scoring System (CVSS - <http://nvd.nist.gov/cvss.cfm>). CVE provides the dictionary of security flaws and CVSS provides the formula for computing severity. Security Quality Score is a single score from 0 to 100, where 0 is the most insecure application and 100 is an application with no detectable security flaws.

Veracode assigns a severity level to each flaw type based on three foundational application security requirements - Confidentiality, Integrity and Availability. Each of the severity levels reflects the potential business impact if a security breach occurs across one or more of these security dimensions.

Confidentiality Impact: According to CVSS, this metric measures the impact on confidentiality if a exploit should occur using the vulnerability on the target system. At the weakness level, the scope of the Confidentiality in this model is within an application and is measured at three levels of impact - None, Partial and Complete.

Integrity Impact: This metric measures the potential impact on integrity of the application being analyzed. Integrity refers to the trustworthiness and guaranteed veracity of information within the application. Integrity measures are meant to protect data from unauthorized modification. When the integrity of a system is sound, it is fully proof from unauthorized modification of its contents.

Availability Impact: This metric measures the potential impact on availability if a successful exploit of the vulnerability is carried out on a target application. Availability refers to the accessibility of information resources. Almost exclusive to this domain are denial-of-service vulnerabilities. Attacks that compromise authentication and authorization for application access, application memory, and administrative privileges are examples of impact on the availability of an application.

Veracode Flaw Severities

Severity	Name	Description
5	Very High	The offending line or lines of code have a very serious weakness and is an easy target for an attacker. The code should be modified immediately to avoid potential attacks.
4	High	The offending line or lines of code have significant weakness, and the code should be modified immediately to avoid potential attacks.
3	Medium	A weakness of average severity. These should be fixed in high assurance software. A fix for this weakness should be considered after fixing the very high and high for medium assurance software.
2	Low	This is a low priority weakness that will have a small impact on the security of the software. Fixing should be consideration for high assurance software. Medium and low assurance software can ignore these flaws.
1	Very Low	Minor problems that some high assurance software may want to be aware of. These flaws can be safely ignored in medium and low assurance software.

Security Quality Score Calculation

The overall Security Quality Score is computed by aggregating impact levels of all weaknesses within an application and representing the score on a 100 point scale. This score does not predict vulnerability potential as much as it enumerates the security weaknesses and their impact levels within the application code.

The Raw Score formula puts weights on each flaw based on its impact level. These weights are exponential and determined by empirical analysis by Veracode's application security experts with validation from industry experts. The score is normalized to a scale of 0 to 100, where a score of 100 is an application with 0 detected flaws using the analysis technique for the application's assurance level.

Ratings Calculation

The Rating is calculated by looking up the required Security Quality Score for a particular assurance level (AL) in the ratings table on page 9. Higher assurance levels require higher scores to achieve an A rating. For example an AL5 application requires a score of 90 for an A rating while an AL4 application requires a score of 80.

When multiple analysis techniques are used, an application can achieve a multi-letter rating. A score is computed for all the vulnerabilities detected for each analysis technique separately and a letter rating is assigned based on the assurance level. For example, an AL4 application with a static analysis score of 85 and a dynamic analysis score of 78 gets an A static analysis rating and a B dynamic analysis rating for an overall rating of AB.

Flaws Related to PCI Compliance

The flaws related to PCI compliance chart shows the distribution of compliance related flaws by severity. An application can get a mediocre security rating by having a few high risk flaws or many medium risk flaws.

Immediate Action Items for Compliance

The list of action items for compliance is Veracode's short term to long term recommendations for improving the compliance posture and security quality of this application.

The fixes necessary for PCI compliance section provides recommendations on all the flaws that should be fixed to achieve compliance. Once the flaws are remediated, the application should be submitted for a re-scan to verify that the flaws have been fixed and no serious new flaws have been introduced.

The fixes necessary to reduce additional risk section highlights security flaws that may not be covered under PCI but still have a security risk relevance for your application. These are flaws that were detected in the course of automated and/or manual analysis of the application that we recommend should be remediated. This section provides guidance on which flaws to remediate to obtain an A rating.

The additional recommendations section provides guidance on maintaining good security practices for the long term and on a continuous basis. Examples include additional automated analysis techniques based on the assurance level that have not yet been completed, manual analysis if recommended, fixing any flaws to obtain an A rating and training software engineers on application security principles and practices.

Summary of Flaws by PCI Requirement

The summary of flaws by PCI requirement categorizes all PCI relevant flaws according to PCI Requirement 6.5.1 through 6.5.10, including a description of the requirement, the flaw category in which the flaws were found and a count of flaws per flaw category.

We appreciate your business and your commitment to working with us to make the world a safer place.