

White Paper

Is Open Source Software Putting Your PCI Compliance at Risk?

The Underestimated Threat and Associated Challenges of Undocumented Open Source



PALAMIDA

Application Security for Open Source Software

215 Second Street, First Floor
San Francisco, California 94105
415.777.9400

www.palamida.com

Executive Overview

In December 2004, Visa, Mastercard, American Express, Discover, and JCB joined forces and aligned their individual policies to create the Payment Card Industry Data Security Standard (PCI DSS), run by the PCI DSS Council. Aimed at organizations that process credit payments, PCI DSS is a regulation for the prevention of credit card fraud, hacking and various other security vulnerabilities. The PCI DSS Standard establishes specific requirements for payment account security, guidance for implementation of the standards, and compliance schedule by which all members, merchants, and service providers who process, store or transmit payment card data must adhere.

According to the 2008 Data Breach Study published by the Verizon Business Risk Team, the most common type of attacks by cyber criminals targeted applications and software. The study goes on to cite that 73% of all attacks were from external sources exploiting existing vulnerabilities in those same applications and software. And the two biggest industry verticals impacted by these attacks? Financial Services and Retail.

And therein lays the problem. According to Privacy Rights Clearinghouse, there have been more than 227 million records containing sensitive personal information involved in security breaches between January 2005 and June 10, 2008¹.

As such, in September 2006, in response to market needs, the PCI DSS Council published Version 1.1 of the Standard, which focused increased attention on application layer security requirements. Companies that don't comply risk losing their right to process card payments, being audited, and/or fined up to \$500,000 per incident of non-compliance.

The onus lies heavily on IT and Security Departments to focus their attention on application security at the source code level and to implement application security processes in order to ensure not only a vulnerability-free environment, but to prove that the appropriate processes are in place for ongoing vulnerability management. This is not an easy feat, due to the granular level of source code management that the PCI DSS requires. In order to achieve success with PCI compliance, it is imperative that organizations realize the potential gaps in their current application security strategies and implement changes where necessary.

Application Security – The New Frontier

Application security is more susceptible than ever in today's dynamic application development landscape. In fact, the root cause of many application security vulnerabilities lies in the application source code. According to research by Gartner Group and Symantec, close to 90% of software attacks were aimed at the application layer².

Once only a perimeter issue, security now has permeated its original boundaries and spread to an organization's most important asset - its software applications. When an application vulnerability is exploited, a company is at risk of potential loss of vital customer data, including credit card numbers and personal information attached to these records.

The Underestimated Threat and Associated Challenges of Undocumented Open Source

Often times, the philosophy of "spend small, think small" prevails for most IT organizations. Unless an organization is adopting a large open source project such as Linux, special resources are not being allotted to the management of open source adoption. In today's world of 24/7 and persistent network access, developers dispersed across multi-national sites can include open source, freeware, public domain, evalware (demos of commercial software), etc., in the code they are writing without triggering the usual checkpoints in the procurement process. Without these controls, the open source is unlikely to be detected, monitored, and tracked. As a result, IT organizations are unaware of exactly what comprises their code base.

Open Source is Often Neglected After Adoption

While vulnerabilities within commercial applications tend to be managed by the vendor, custom applications can be trickier. In recent studies, Palamida found that applications written within the last five years contain 50% or more open source code, by a line of code count. Of that 50 % of open source code, up to 70% can go undocumentedⁱⁱⁱ.

Typical Audit Results

F1000 Company Audit Results	
Size	60 million lines of code
Documented Open Source Projects (pre-audit)	303
Additional Open Source and Other 3rd-party Projects (post-audit)	535
Breakdown of Code	65% open source/3rd party vs. 35% proprietary

Source: Palamida for Fortune 1000 company in December 2007

The benefits of open source are readily apparent – it reduces cost of development, shortens development cycles, and can lower overall total cost of ownership of your applications if managed well. Developers, who have long been at the forefront of open source use, have seen a growing awareness and an acceptance among their organizations that make the adoption of open source components a common practice.

While open source is important to a company's bottom line, the large percent of undocumented open source within the custom applications presents a huge security challenge to organizations industry-wide, especially those concerned with PCI compliance. The problem is that the sheer size of a typical code base coupled with the number of contributing developers makes it nearly impossible for companies to get an accurate assessment of their software assets, much less a clear understanding of the vulnerabilities associated with the adopted code.

- What do you have?
- Where is it across your global code base?
- Where did it come from?
- What are its vulnerability risks?

While commercial software suppliers have put in place mechanisms to notify customers of updates and push them out automatically to licensed users, many open source projects do not. This puts the burden of responsibility on the user to monitor whether there are security patches or newer versions of the open source project available.

When companies do not know what code is running inside their applications, they are leaving themselves open to serious vulnerabilities that can ultimately lead to data breaches.

Security Data Breaches on the Rise

According to the 2008 Data Breach Study published by the Verizon Business Risk Team, the most common type of attacks by cyber criminals targeted applications and software. The study goes on to cite that 73% of all attacks were from external sources exploiting existing vulnerabilities in those same applications and software. And the two biggest industry verticals impacted by these attacks? Financial Services and Retail.

And therein lays the problem. According to Privacy Rights Clearinghouse, there have been more than 227 million records containing sensitive personal information involved in security breaches between January 2005 and June 10, 2008^{iv}. And in subsequent years, that number has grown larger every month. The consequences of a data breach for organizations can be significant. A recent Forrester study reported the cost of a data breach to run anywhere between \$90-\$305 dollars per record^v.

Examples of High Visibility Credit Card Data Breaches

Hannaford Brothers

In March 2008, a the supermarket chain reported a security breach affecting all of its 165 stores in the Northeast, 106 Sweetbay stores in Florida and a smaller number of independent groceries that sell Hannaford products. The company is currently aware of about 1,800 cases of reported fraud related to the security breach. Credit and debit card numbers were stolen during the card authorization transmission process, but no personal information was divulged. In total, 4.2 million records could be compromised.

Ralph Lauren and HSBC

In April 2004, U.S. Secret Service agents found Ralph Polo Lauren customers' credit card numbers in the hands of Eastern European cyber thieves who created high-quality counterfeit credit cards. Victims are from the U.S., Europe, Asia and Canada, among other places, Several Cuban nationals in Florida were arrested with more than 200,000 credit card account numbers.

In order to mitigate the ongoing problem of security breaches, the PCI DSS Council included new mandates in Version 1.1 that are directed at securing the source code of custom applications. And in order to accomplish PCI compliance, IT departments need to closely manage their code, including all open source software and potential vulnerabilities.

PCI DSS Requirements Version 1.1

When compared to other regulations such as HIPAA and Sarbanes-Oxley, the PCI DSS is considered one of the more comprehensive data security standards today. Version 1.1 of the Data Security Standard requires vendors to:

- Build and Maintain a Secure Network
 - Requirement 1: Install and maintain a firewall configuration to protect cardholder data
 - Requirement 2: Do not use vendor-supplied defaults for system passwords and other security parameters
- Protect Cardholder Data
 - Requirement 3: Protect stored cardholder data
 - Requirement 4: Encrypt transmission of cardholder data across open, public networks
- Maintain a Vulnerability Management Program
 - Requirement 5: Use and regularly update anti-virus software
 - Requirement 6: Develop and maintain secure systems and applications

- Implement Strong Access Control Measures
 - Requirement 7: Restrict access to cardholder data by business need-to-know
 - Requirement 8: Assign a unique ID to each person with computer access
 - Requirement 9: Restrict physical access to cardholder data
- Regularly Monitor and Test Networks
 - Requirement 10: Track and monitor all access to network resources and cardholder data
 - Requirement 11: Regularly test security systems and processes
- Maintain an Information Security Policy
 - Requirement 12: Maintain a policy that addresses information security

And within these twelve requirements are sixty-plus very detailed sub-requirements that have been mandated by the PCI Security Council to help organizations implement and prepare for compliance.

It's in the Details: Requirements as Related to Undocumented Open Source

Of the twelve, there are three requirements that are impossible to meet if undocumented open source software is not identified and managed properly – thereby making PCI compliance unattainable.

Requirement 3: Protect Stored Cardholder Data – Encryption is a critical component of cardholder data protection. If an intruder circumvents other network security controls and gains access to encrypted data, without the proper cryptographic keys, the data is unreadable and unusable to that person.

Applications play a key role in protecting stored cardholder data. Vulnerabilities within the application's code base could be harmful to the security of the private data. For most of today's web and software applications, cryptography and its applications are increasingly relevant. Consider the significant amount of information users transmit and receive over the course of day and the potential ways that information could be exploited.

The open source community has shown strong dedication for the encryption tools that can be used daily to protect your privacy and reduce some of the risks for information abuse. They are all easily available and enterprise-ready for your developers to download and use. They also fit the profile of the type of open source projects that can often go undocumented.

Popular Open Source Cryptography Tools for Wide Range of Uses

Project	Description
PGP Gnu Privacy Guard	Popular for file encryption, a staple of the open source crypto toolkit, it is a command line tool that allows users to generate and manage their own crypto keys, those of others, and encrypt/decrypt data. It is PGP compatible.
Seahorse	User friendly Gnome front-end to GnuPG that gives users control over their keys as well as integration with the Nautilus file manager. Useful for managing the keys of people you know, signing them, and exporting/importing from key servers.
Off-the-Record Messaging	Library for secure instant messaging. A plugin is available for the Gaim instant messaging client, and can be used on Windows as well as Unix. OTR generates public/private keys for various instant messaging accounts, and can then exchange these with others for secure chat, provided they're running the OTR plugin.
Revelation	Gnome application for securely storing and managing all logins and passwords.
Dm-crypt	transparent disk encryption subsystem in Linux kernel versions 2.6 and later. It is part of the device mapper infrastructure, and uses cryptographic routines from the kernel's Crypto API.

Requirement 4: Encrypt transmission of cardholder data across open, public networks - Sensitive information must be encrypted during transmission over networks that are easy and common for a hacker to intercept, modify, and divert data while in transit.

Undocumented open source code could be potentially very harmful to the safety of cardholder data. Organizations need to be able to document the encryption tools they are using in case there are issues with the utility. For example in 2006, developers of the open-source GnuPG encryption software reported a security flaw that could allow an attacker to sneak malicious code into a signed e-mail message. GnuPG is included with several versions of Linux as well as FreeBSD, and is also used widely used by the IT security industry.

The vulnerability allows an attacker to take a signed message and insert additional code, which then appears to the recipient as if it were part of the digitally signed content.

An attacker can intercept the message during transmission, inject additional data, and then the person who verifies the signature would be told it is a valid, unaltered message. The attacker could potentially alter a text file, like a business contract, or an executable file attached to the message.

It affects all versions of GnuPG prior to 1.4.2.2, and users are advised to upgrade at once to that release. The GnuPG team uncovered the flaw while testing the patch for a previous vulnerability reported in February of 2006. Upgrading to the 1.4.2.2 release fixes the problem and more information is on the GnuPG web site.

Most open source projects are great at monitoring themselves, as this example shows. When the community realizes it needs a patch, that patch can literally be up within hours. But unlike commercially supported third-party products, only 1 out of every 10 open source project has a vendor offering commercial support^{vi}. As a result, organizations that use open source components are largely “on their own” when it comes to patches, upgrades, vulnerability assessment and similar tasks that are part of a normal commercial service contract.

Requirement 6: Develop and maintain secure systems and applications - Unscrupulous individuals use security vulnerabilities to gain privileged access to systems. Many of these vulnerabilities are fixed by vendor-provided security patches. All systems must have the most recently released, appropriate software patches to protect against exploitation by employees, external hackers, and viruses.(see Appendix for detailed account of Requirement 6.)

The core regulation addressing the need to secure applications at the source code level, introduces the basis of a secure application, the necessary steps that need to be taken in order to be PCI compliant, and the verification of compliance required.

The key to Requirement 6 is putting the appropriate processes in place to both make compliance possible and exhibit proof of said processes. While the PCI DSS Council was very specific in identifying exactly what needs to be done to reach compliance, an organization must work granularly in order to ensure the security of their applications from the unforeseen exposure due to undocumented code within their custom applications.

Undocumented open source software presents the greatest risk to PCI compliance with regards to Requirement 6.5 and 6.6 below:

Requirement 6.5 - 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:

6.5.1 Unvalidated input

6.5.2 Broken access control (for example, malicious use of user IDs)

6.5.3 Broken authentication and session management (use of account credentials and session cookies)

6.5.4 Cross-site scripting (XSS) attacks

6.5.5 Buffer overflows

6.5.6 Injection flaws (for example, structured query language (SQL) injection)

6.5.7 Improper error handling

6.5.8 Insecure storage

6.5.9 Denial of service

6.5.10 Insecure configuration management

Requirement 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
- Installing an application layer firewall in front of web-facing applications.

In 2007, Palamida’s Professional Services team evaluated between 300M to 500M lines of code and compiled a list of the Top 5 Most Overlooked Open Source Vulnerabilities. The list was compiled based on the combination of which projects were most often undocumented and associated, known vulnerabilities. Just these Top 5 projects, should an organization have them embedded inside their applications, will put them at risk of non-compliance for section 6.6.

Top 5 Most Overlooked Open Source Vulnerabilities 2007

Project	Vulnerability	Vulnerability Type
Apache Geronimo 2.0	CVE-2007-4548	Allows attackers to bypass authentication requirements and deploy arbitrary code
JBoss Application Server 3.2.4 to 4.0.5	CVE-2006-5750	Allows remote users to read or modify arbitrary files
LibTiff before 3.8.2	CVE-2006-3464	Allows attackers ability to deploy arbitrary code
Net-SNMP 5.2.x before 5.2.2, 5.1.x before 5.1.3, 5.0.x before 5.0.10.2	CVE-2005-4837	Allows remote attackers ability to cause denial of service
Zlib before 1.2.3	CVE-2005-2096	Allows remote attackers ability to cause denial of service

If identifying and managing all undocumented open source software within your organization’s custom applications is not a top priority, then achieving compliance with Requirement 6.5 and 6.6 is virtually impossible, since open source code is just as likely as proprietary code to be vulnerable to all of the common code vulnerabilities listed within the Standard.

Bottom Line - What Can Organizations Do to Avoid Security Breaches?

Application security cannot be adequately addressed until application development and security teams agree to work together in understanding and managing security. Neither role benefits from a siloed existence. For instance, quality assurance and virus prevention are not mutually exclusive in application development.

Development teams commonly (and mistakenly) believe that security is solely the responsibility of security professionals. For many companies the development process has solely focused on designing, architecting, coding and testing applications to ensure that they fulfill functional requirements. Particularly with today's strained budgets and resources, applications are not adequately tested for coding defects, vulnerability assessment, or conditions that could leave the applications exposed to external attacks.

Simultaneously, the mandate for security teams has been focused on defending the perimeter against external attacks. They have made significant investment in firewalls, web-based authentication, intrusion detection, and identity management systems. Security professionals are not coders and often do not realize that decisions made during the development process have significant material impact on the deployed applications they are mandated to protect.

Therefore, neither team can adequately cover application security problems alone. Hackers, who often know coding methodologies better than security professionals and know security better than programmers, are exploiting this gap.

It is the responsibility of security and development teams to ensure that their developers use processes that produce secure software. Working together, these two departments need to insert application security into the overall security strategy for their companies by:

- Conducting code-level security reviews in addition to penetration tests for their internally developed code before deployment
- Insisting that code-level audits have been conducted by outsourced development and business partners
- Ensuring that all other third-party code included in their software applications is identified and tracked for security flaws and updated version information
- Ensuring that internally developed applications have adequate checkpoints that enable thorough audit trails

Conclusion

The identification, monitoring and management of undocumented open source software in order to ensure that no code goes unnoticed in applications or running behind firewalls has not been managed well, to date. But with the increasing scrutiny of the problems associated with application-layer attacks, increased customer data breach, and policies such as PCI DSS, the need for automated application security and governance systems for open source code use becomes a key part of companies engineering, IT and application security priorities. By implementing new application security software systems, companies will be able to decrease the total cost of ownership of open source technology, increase customer and partner confidence, maintain competitive advantage and ensure regulatory and corporate compliance.

Appendix

6.1 Ensure that all system components and software have the latest vendor-supplied security patches installed. Install relevant security patches within one month of release.

6.2 Establish a process to identify newly discovered security vulnerabilities (for example, subscribe to alert services freely available on the Internet). Update standards to address new vulnerability issues.

6.3 Develop software applications based on industry best practices and incorporate information security throughout the software development life cycle.

6.3.1 Testing of all security patches and system and software configuration changes before deployment

6.3.2 Separate development, test, and production environments

6.3.3 Separation of duties between development, test, and production environments

6.3.4 Production data (live PANs) are not used for testing or development

6.3.5 Removal of test data and accounts before production systems become active

6.3.6 Removal of custom application accounts, usernames, and passwords before applications become active or are released to customers

6.3.7 Review of custom code prior to release to production or customers in order to identify any potential coding vulnerability. Establish a process to identify newly discovered security vulnerabilities (for example, subscribe to alert services freely available on the Internet). Update standards to address new vulnerability issues.

6.4 Follow change control procedures for all system and software configuration changes. The procedures must include the following:

6.4.1 Documentation of impact

6.4.2 Management sign-off by appropriate parties

6.4.3 Testing of operational functionality

6.4.4 Back-out procedures

6.5 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:

6.5.1 Unvalidated input

6.5.2 Broken access control (for example, malicious use of user IDs)

6.5.3 Broken authentication and session management (use of account credentials and session cookies)

6.5.4 Cross-site scripting (XSS) attacks

6.5.5 Buffer overflows

6.5.6 Injection flaws (for example, structured query language (SQL) injection)

6.5.7 Improper error handling

6.5.8 Insecure storage

6.5.9 Denial of service

6.5.10 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
- Installing an application layer firewall in front of web-facing applications.

Note: This method is considered a best practice until June 30, 2008, after which it becomes a requirement.

About Palamida, Inc.

Palamida is the industry's first application security solution exclusively for Open Source Software that uses component-level analysis to quickly identify and track undocumented code and associated security vulnerabilities as well as intellectual property and compliance issues, enabling development organizations to cost-effectively manage and secure mission critical applications and products.

For more information visit www.palamida.com

ⁱPrivacy Rights Clearinghouse: a Chronology of Data Breaches, June 10, 2008

ⁱⁱBriggs, Linda L. "Application Security Comes Under Attack," Application Development Trends June 2006:1

ⁱⁱⁱIn 2007, Palamida Services team audited between 300M to 400M lines of code for F500 to venture-backed companies across, multiple industries. Results were tabulated based on blind audit results.

^{iv}Privacy Rights Clearinghouse: a Chronology of Data Breaches, June 10, 2008

^vDignan, Larry, "What that Data Breach Will Really Cost You," ZDNet Between the Lines May 2007: 8

^{vi}Based on Palamida research conducted February 29, 2008 - March 4, 2008, examining support structure for 3,168 popular open source projects