Realtime publishers

"Leading the Conversation"

The Essentials Series

PCI Compliance

sponsored by

ALERTLOGIC

by Rebecca Herold

Copyright Statement

Addressing Application Vulnerabilities with PCI Log Management Compliance

By Rebecca Herold, CIPP, CISSP, CISA, CISM, FLMI

Applications with significant and dangerous vulnerabilities continue to be pushed into production. More and more application vulnerabilities are being exploited, purposefully and accidentally, resulting in not only bad press for organizations but also costly incident response activities and personally identifiable information (PII) privacy breaches.

Meeting the PCI DSS requirements for logging can benefit businesses by putting into place logs that reveal vulnerabilities within applications that could have lead to information security incidents and privacy breaches if they were not discovered. Such vulnerabilities can also result in significant fines for PCI DSS non-compliance. Too many information security and IT practitioners do not clearly understand potential great negative impacts that poorly engineered and vulnerable applications can have upon the business.

Application Flaws Impact Business

Early in my career, I was a systems analyst and applications programmer. I followed a stringent applications testing procedure as part of the company's change management policy when moving new and updated programs into the production environment. During this testing, we checked not only input and output but also the logs to ensure the application was communicating with only the other databases, common modules, applications, and other systems resources as intended. The logs were of great value to our testing activities.

For example, one time while our team was testing changes for a payment processing application, the immediate outputs on the screens for the corresponding inputs all looked valid. However, upon checking the logs of the application following the test run, we saw that the application was actually applying the payments to the wrong database—a mirror of the production database that was created to serve as a backup but was not to be used during the actual payment processing cycle. The net result of this error was that the payments in that backup database were overwritten at the end of the application processing cycle by the valid customer database, negating the customer payments. If this had not been caught, and the application had gone into production, we would likely have had a very large number of angry customers contacting the company upon receiving bills for amounts that they had already paid! Many of the customers would likely have taken their business elsewhere.

It is very easy to make application errors, especially if there is a tight timeline for putting the application into production; application testing often gets shortchanged as a result. Following log management procedures to verify the application has executed as intended helps to keep errors from being put into production and facilitates catching errors in production applications as soon as possible, helping to keep mistakes and vulnerability exploits from negatively impacting business.

PCI DSS Log Compliance Can Help Improve Application Security

PCI DSS log management requirements can be used to mitigate the possibility of putting buggy applications into production and catch vulnerabilities within applications already in production. For example, log management procedures can be used to

- Troubleshoot problems and bugs in applications using the logs
- Analyze data flows and identify data leaks and vulnerabilities
- Identify inappropriate links to other systems components that may put PII at risk
- Determine inappropriate storage points that put PII at risk
- Identify erroneous storage of prohibited data, such as PII

There are many ways in which logs can be used to improve applications security while helping to support compliance with PCI DSS log management requirements. Kevin Beaver, founder and principal information security consultant of Principle Logic, LLC, talked with me about five such methods.

1. Remote Code Execution Vulnerabilities

Coding errors, or poor quality code, can create vulnerabilities that could allow an attacker to run arbitrary, system-level code on the vulnerable server and access sensitive information. A couple of examples of code that, in the past, have contained such vulnerabilities include Invision Board and the PayPal cart. How can logs point to a remote code execution vulnerability within an application?

According to Kevin, "Server logs could certainly point out that at least a connection was made and when it happened. More specifically, application and/or database logs could point out that certain system calls were made, certain data was accessed, or specific data was added, modified, or transferred."

2. SQL Injection Vulnerabilities

SQL injection vulnerabilities within applications can allow an attacker to retrieve information from a Web server's database. Depending upon how the application is engineered, the attack can vary from basic information disclosure to remote code execution and even possibly total system compromise. Could logs indicate whether an application is vulnerable to SQL injection, and if so, how?

Kevin answers, "Sure, all you'd have to do is look for specific SQL commands such as SELECT, CONNECT, and UNION being submitted to the server or application. You may also have log entries for SQL errors that are being generated and displayed back to the user of the application."

3. Format String Vulnerabilities

It can be a great challenge to secure applications that interact with Web-based customers, such as e-commerce applications—especially when so many of those trying to use the application may in fact have malicious intent. Applications written in C can be a particular challenge to secure because of the security that is often traded to gain efficiency. Something that often results is format string vulnerability.

Since the discovery of the format string vulnerability, many hackers have exploited it to gain root access to these vulnerable systems. An application that does not adequately filter user input, such as the format string parameter in certain Perl or C functions that perform formatting (such as C's printf() command), will create the ability for a malicious attacker to perform a Denial of Service (DoS) attack, read confidential data, or perhaps even write to sensitive data bases. Could logs indicate whether an application has format string vulnerabilities?

Kevin replies, "Yes, potentially; if the application has logging capabilities. As with the remote code problems mentioned earlier, you may at least be able to see basic connections being made from the application."

4. Cross Site Scripting

Poorly engineered applications may be vulnerable to cross site scripting (XSS). This attack is typically accomplished when the victim executes a malicious URL that appears to be legitimate, and then goes to that URL only to have malicious code executed in their browser. Examples of products that have been vulnerable to this type of exploit in the past include Yahoo Mail and Google search, just to name a couple.

I asked Kevin whether logs could indicate if an application has format string vulnerabilities. "Yes, definitely. An application could be coded to log anything that's out of the ordinary. That said, if developers are writing logging capabilities such as this, then they've hopefully addressed the basic XSS problems to begin with! You can also set up Web servers and other systems to trigger on suspect input such as <script> tags."

5. Username Enumeration

Sometimes applications are coded in such a way that they allow for username enumeration. Applications can be prone to username enumeration exploits because of the inadequate ways in which they verify end user–supplied application input. Attackers may exploit this weakness to determine valid usernames. This can assist attackers in brute-force password cracking or other types of attacks.

Username enumeration can be exploited when the back-end validation code tells the attacker whether the supplied username correct or not. The attacker can then try different usernames until the attacker finds valid ones by noting the different error messages.

Kevin has recent experience with this type of exploit. "I just came across this very vulnerability in a Web application. Detection of this vulnerability is as simple as looking for large numbers of 'incorrect user' and 'incorrect password' errors and distinguishing between the two to determine that something's going on. This could be done at the server or the application level depending on how the site and/or application works."

An IT Security Expert Practitioner's Perspective

A.B., an IT security expert practitioner with more than 25 years of experience within large multi-national companies (A.B.'s corporate rules do not allow him to have his name or his company's name published; however, his great expertise and vast experience offers some valuable lessons to readers), offers some good points and examples about how he has successfully used logs to help identify vulnerabilities within applications.

> *Consider this scenario: The log file of a web server that is a front end to a database shows that a remote user has accessed files that are outside of the web server's environment. For example if the web server's environment is under the directory tree (e.g., c:/applications/database/webserver/) and you see users in the application log fetching files from the root level of the c: drive (e.g., "c:/"), then something is very wrong. The developers should consider that they did not do an adequate job of URL input validation.*
>
> *Here's another example; a programming pet peeve of mine. Suppose a log file shows that a hacker was able to subvert an application's input validation routine. The command he used is in the log file. Analysis of the log shows that the hacker abused a character that was supposed to cause an illegal input error message. When the developers look at the program, they discover that the abused character was not in the list of illegal input characters, an easy mistake to make after long hours of programming. But here's my pet peeve: The programmers took the wrong approach in error checking. By scanning the input characters for illegal characters, they take the risk of not checking for **ALL** of the illegal characters (which is what happened). Instead, it is best to ensure that all characters in the input are legal (e.g., scan for legal characters). If the program finds a character that is not legal, then the application should signal an error message. With this approach, if a legal character is erroneously programmed as illegal (e.g., an apostrophe), Mr. BigBoss will complain. Better an annoyed legitimate user than a successful hacker.*
>
> *Another example is a log file showing the mysterious creation of a privileged account on a web server. There is a possibility that a hacker was able to craft input to break out of the web environment to the command line. If the web server was running under a privileged account, he could have created an administrator account. Until recently SQL, by default, allowed SQL commands to execute DOS commands. A bonehead idea from a security point of view. I think Microsoft reconsidered that idea and fixed this vulnerability.*

Prepare for PCI DSS Compliance and Increase the Security of Applications

Kevin indicated the following are significant application vulnerabilities that he often runs across, "Username and password enumeration is a biggie; it is common to find these vulnerabilities within e-commerce applications. Another one is basic URL manipulation and hidden field manipulation. Both of these types of vulnerabilities can be detected in Web logs and help prevent malicious unauthorized use of the application."

Creating effective log management practices supports PCI DSS compliance and helps to protect the business against using applications that contain vulnerabilities such as those described, in addition to others. When creating your log management procedures to detect application vulnerabilities, include checks to look for the following:

- Remote code execution, especially for Web applications that process credit cards
- Username and password enumeration vulnerabilities
- Incorrect database updates
- SQL injection vulnerabilities
- Format and input string validation vulnerabilities
- Cross site scripting
- Applications inappropriately running under privileged accounts
- Multiple application sessions using the same username

Do not consider this a comprehensive list but rather a starting point to help you identify which application logs to use within your organization. These practices will not only help protect your business, they will also help to support PCI DSS log management requirements.