Security Standards Council™

Standard: Data Security Standard (DSS)
Requirement: 6.6
Date: February 2008

Information Supplement: Requirement 6.6 Code Reviews and Application Firewalls Clarified

Release date: 2008-04-15

General

PCI DSS Requirement 6.6 provides two options that are intended to address common threats to cardholder data and ensure that input to web applications from untrusted environments is inspected "top to bottom." The details of how to meet this requirement will vary depending on the specific implementation supporting a particular application.

Forensic analyses of cardholder data compromises have shown that web applications are frequently the initial point of attack upon cardholder data, through SQL injection in particular.

The intent of Requirement 6.6 is to ensure web applications exposed to the public Internet are protected against the most common types of malicious input. There is a great deal of public information available regarding web application vulnerabilities. The minimum vulnerabilities to consider are described in Requirement 6.5. (Refer to the References section for other sources of information on web application testing.)

Proper implementation of both options would provide the best multi-layered defense.

PCI SSC recognizes that the cost and operational complexity of deploying both options may not be feasible. Further, one or the other option may not be possible in some situations (no access to source code, for example). However, it should be possible to apply at least one of the alternatives described in this paper, and proper implementation can meet the intent of the requirement.

This document provides guidance to assist in determining the best option, which can vary depending on products in use, how an organization procures or develops its web applications, and other factors within the environment.

Requirement 6.6 Option 1 – Application Code Reviews

The application code review option does not necessarily require a manual review of source code. Keeping in mind that the objective of Requirement 6.6 is to prevent exploitation of common vulnerabilities (such as those listed in Requirement 6.5), several possible solutions may be considered. They are dynamic and pro-active, requiring the specific initiation of a manual or automated process. Properly implemented, one or more of these four alternatives could meet the intent of Option 1 and provide the minimum level of protection against common web application threats:

1. Manual review of application source code
2. Proper use of automated application source code analyzer (scanning) tools
3. Manual web application security vulnerability assessment
4. Proper use of automated web application security vulnerability assessment (scanning) tools

All of these must be designed to test for the presence of web application vulnerabilities as indicated under "General" above. Note that a vulnerability assessment simply identifies and reports vulnerabilities, whereas a penetration test attempts to exploit the vulnerabilities to determine whether unauthorized access or other malicious activity is possible.

Manual reviews/assessments may be performed by a qualified internal resource or a qualified third party. In all cases, the individual(s) must have the proper skills and experience to understand the source code and/or web application, know how to evaluate each for vulnerabilities, and understand the findings. Similarly, individuals using automated tools must have the skills and knowledge to properly configure the tool and test environment, use the tool, and evaluate the results.

If internal resources are being used, they should be organizationally separate from the management of the application being tested. For example, the team writing the software should not perform the final review or assessment and verify the code is secure.

There are several ways to perform a code review or application assessment that would provide the same (or better) protection provided by an application firewall (discussed below as Option 2). Two examples that may meet the intent of the first option are:

1. An organization's having in place, as part of its software development life cycle (SDLC), a process whereby web application source code undergoes an independent security review. The security review should consist of examining applications for controls that address common web application vulnerabilities. These code reviews may be implemented as manual or automated processes.
2. Use of a manual process or specialized tools to test for the presence of exposed vulnerabilities and defects in an executing web application. This approach involves creating and submitting malicious or non-standard input to the application, thus simulating an attack. The responses to that input are examined for indications that the application may be vulnerable to certain attacks.

The reviews or assessments should be incorporated into the SDLC and performed prior to the application's being deployed into the production environment. The SDLC must incorporate information security throughout, per Requirement 6.3. Change control processes must ensure that software developers are not able to bypass the code review/application assessment step and deploy new software directly into the production environment. Change control processes must also enforce the correction and retesting of vulnerabilities before implementation.

While the final sign-off/approval of the review/scan results must be done by an independent organization, it is recommended that reviews and scans also be performed as early as possible in the development process. Tools should be made available to software developers and integrated into their development suite as much as practical.

Vendors may bundle code scanning or vulnerability assessment capabilities in products that have other objectives, such as validating conformance to architectural best practices related to a specific language. When evaluating these tools, it is important to confirm the tool's ability to test for the common web application vulnerabilities before assuming it can be used to meet the intent of Requirement 6.6. Also, to meet new and emerging threats, tools should have the ability to incorporate new analysis rules. Individuals performing manual reviews or assessments must stay current with industry trends to ensure their evaluation or testing skills continue to address new vulnerabilities.

Requirement 6.6 Option 2 – Application Firewalls

In the context of Requirement 6.6, an “application firewall” is a web application firewall (WAF), which is a security policy enforcement point positioned between a web application and the client end point. This functionality can be implemented in software or hardware, running in an appliance device, or in a typical server running a common operating system. It may be a stand-alone device or integrated into other network components.

Typical network firewalls are implemented at the perimeter of the network or between network segments (zones) and provide the first line of defense against many types of attacks. However, they must allow messages to reach the web applications an organization chooses to expose to the public Internet. Network firewalls usually are not designed to inspect, evaluate, and react to the parts of an Internet Protocol (IP) message (packet) consumed by web applications, and therefore public applications frequently receive uninspected input. As a result, a new logical security perimeter is created—the web application itself—and security best practices call for messages to be inspected when they cross from an untrusted into a trusted environment. There are many known attacks against web applications and, as we are all aware, web applications are not always designed and written to defend against those attacks. Adding to the risk is the availability of these applications to virtually anyone with an Internet connection.

WAFs are designed to inspect the contents of the application layer of an IP packet, as well as the contents of any other layer that could be used to attack a web application. It should be noted, however, that Requirement 6.6 is not intended to introduce redundant controls. IP packet content adequately inspected (i.e., providing equivalent protection) by network firewalls, proxies, or other components do not have to be re-inspected by a WAF.

IP packet structure follows a layered model, with each layer containing defined information that is acted upon by specific network nodes or components (physical or software-based) supporting the flow of information through the Internet or intranet. The layer containing the content that is processed by the application is called the application layer.

Increasingly, WAF technology is integrated into solutions that include other functions such as packet filtering, proxying, SSL termination, load balancing, object caching, etc. These devices are variously marketed as “firewalls,” “application gateways,” “application delivery system,” “secure proxy,” or some other description. It is important to fully understand the data-inspection capabilities of such a product to determine whether the product could satisfy the intent of Requirement 6.6.

Note that compliance is not assured by merely implementing a product with the capabilities described in this paper. Proper positioning, configuration, administration, and monitoring are also key aspects of a compliant solution. Implementing a WAF is one option to meet Requirement 6.6 and does not eliminate the need for a secure software development process (Requirement 6.3).

Recommended Capabilities

A web application firewall should be able to:

- Meet all applicable PCI DSS requirements pertaining to system components in the cardholder data environment.
- React appropriately (defined by active policy or rules) to threats against relevant vulnerabilities as identified, at a minimum, in the OWASP Top Ten and/or PCI DSS Requirement 6.5.
- Inspect web application input and respond (allow, block, and/or alert) based on active policy or rules, and log actions taken.
- Prevent data leakage—meaning have the ability to inspect web application output and respond (allow, block, mask and/or alert) based on the active policy or rules, and log actions taken.
- Enforce both positive and negative security models. The positive model ("white list") defines acceptable, permitted behavior, input, data ranges, etc., and denies everything else. The negative model ("black list") defines what is NOT allowed; messages matching those signatures are blocked, and traffic not matching the signatures (not "black listed") is permitted.
- Inspect both web page content, such as Hypertext Markup Language (HTML), Dynamic HTML (DHTML), and Cascading Style Sheets (CSS), and the underlying protocols that deliver content, such as Hypertext Transport Protocol (HTTP) and Hypertext Transport Protocol over SSL (HTTPS). (In addition to SSL, HTTPS includes Hypertext Transport Protocol over TLS.)
- Inspect web services messages, if web services are exposed to the public Internet. Typically this would include Simple Object Access Protocol (SOAP) and eXtensible Markup Language (XML), both document- and RPC-oriented models, in addition to HTTP.
- Inspect any protocol (proprietary or standardized) or data construct (proprietary or standardized) that is used to transmit data to or from a web application, when such protocols or data is not otherwise inspected at another point in the message flow.

  Note: Proprietary protocols present challenges to current application firewall products, and customized changes may be required. If an application's messages do not follow standard protocols and data constructs, it may not be reasonable to ask that an application firewall inspect that specific message flow. In these cases, implementing the code review/vulnerability assessment option of Requirement 6.6 is probably the better choice.
- Defend against threats that target the WAF itself.
- Support SSL and/or TLS termination, or be positioned such that encrypted transmissions are decrypted before being inspected by the WAF. Encrypted data streams cannot be inspected unless SSL is terminated ahead of the inspection engine.

Additional Recommended Capabilities for Certain Environments

- Prevent and/or detect session token tampering, for example by encrypting session cookies, hidden form fields or other data elements used for session state maintenance.
- Automatically receive and apply dynamic signature updates from a vendor or other source. In the absence of this capability, there should be procedures in place to ensure frequent update of WAF signatures or other configuration settings.
- Fail open (a device that has failed allows traffic to pass through uninspected) or fail closed (a device that has failed blocks all traffic), depending on active policy. Note: Allowing a WAF to fail open must be carefully evaluated as to the risk of exposing unprotected web application(s) to the public Internet. A bypass mode, in which absolutely no modification is made to the traffic passing through it, may be applicable in some circumstances. (Even in “fail open” mode, some WAFs add tracking headers, clean up HTML that they consider to violate standards, or perform other actions. This can negatively impact troubleshooting efforts.)
- In certain environments, the WAF should support Secure Sockets Layer (SSL) client certificates and proxying client authentication via certificates. Many modern web applications use client SSL certificates to identify end users. Without this support, these applications cannot reside behind an application firewall. Many modern application firewalls will integrate with Lightweight Directory Access Protocol or other user directories and can even perform initial authentication on behalf of the underlying application.
- Some ecommerce applications may require FIPS hardware key store support. If this is a consideration in your environment, make sure that the WAF vendor supports this requirement in one of their systems and be aware that this feature may drastically increase the cost of the solution.

Additional Considerations

While WAFs can protect against many security threats, they may also expose technical problems within an infrastructure. Be sure to watch out for the following issues that may hinder successful deployment:

- Sites that rely on unusual headers, URLs, or cookies may require special tuning. WAFs often enforce maximum sizes for these components. Additionally, the signatures they look for may filter out specific strings perceived as “exploits” that in fact may be perfectly valid for a specific application.
- Content that does not conform to HTML/HTTP RFCs or is otherwise “unusual” may also be blocked without tuning of the default filters. This could include anything from overly large file uploads to content submitted in foreign character sets or languages.

- DHTML, Asynchronous JavaScript and XML (AJAX), and other dynamic technologies may require special consideration, testing, and tuning. These applications sometimes assume they have access to a web site in a way is perceived as malicious by a WAF.
- Applications that require information about the underlying network session, such as client IP address, may require modification if the WAF acts as a reverse proxy. Generally these WAFs will place client-side information into an HTTP header, which existing applications may not expect.

Important Considerations

- Code reviews and application vulnerability assessments described in this document should be performed prior to implementing the application in production.
- If a WAF “fail open” or “bypass mode” is being considered, specific procedures and criteria defining the use of these higher-risk modes should be established prior to implementation. Web applications are not protected while these modes are active, and long periods of use are not recommended.
- The impact of web application firewall changes must be assessed for potential impact to relevant web applications, and vice versa.
- Communicate timing and scope of production web application firewall changes to all affected parties throughout the organization.
- Adhere to all policies and procedures including change control, business continuity, and disaster recovery.
- Changes to the production environment should occur during a monitored maintenance window.

Additional Sources of Information

This list is provided as a starting point for more information on web application security.

- OWASP Top Ten
- OWASP Countermeasures Reference
- OWASP Application Security FAQ
- Build Security In (Dept. of Homeland Security, National Cyber Security Division)
- Web Application Vulnerability Scanners (National Institute of Standards and Technology)
- Web Application Firewall Evaluation Criteria (Web Application Security Consortium)

About the PCI Security Standards Council

The mission of the PCI Security Standards Council is to enhance payment account security by driving education and awareness of the PCI Data Security Standard and other standards that increase payment data security.

The PCI Security Standards Council was formed by the major payment card brands American Express, Discover Financial Services, JCB International, MasterCard Worldwide, and Visa Inc. to provide a transparent forum in which all stakeholders can provide input into the ongoing development, enhancement, and dissemination of the PCI Data Security Standard (DSS), PIN Entry Device (PED) Security Requirements, and the Payment Application Data Security Standard (PA-DSS). Merchants, banks, processors, and point-of-sale vendors are encouraged to join as Participating Organizations.