SOLUTION BRIEF

riverbed
Think fast.

Why web application firewalls matter

TO GUARD AGAINST MODERN THREATS, TRADITIONAL FIREWALLS AREN'T SUFFICIENT

Web application firewalls (WAF), like Riverbed SteelApp Web Firewall, have a specific purpose: to protect services from attacks that attempt to exploit vulnerabilities in web applications. While general purpose firewalls (both traditional and next-generation) may also provide a degree of such protection, they're typically intended for protecting enterprise networks. WAFs include additional protection capabilities that traditional enterprise firewalls lack, and concentrate these capabilities at the application layer.

WAFs add necessary protection against contemporary threats

WAFs are most often deployed in front of public-facing web servers, and commonly as a way to maintain compliance with PCI DSS Requirement 6.6. This is not the only place or reason for a WAF, however. Internal applications may also require the additional protection of a WAF. When making the decision, consider the business impact: if an internal application were to be compromised, how would that affect the ability of business operations? Business-critical applications can exist anywhere within a network, private and public. They often contain a mix of custom code and third-party components; these themselves can be a mix of modern and legacy code. Finding and mitigating every weakness and potential attack vector can be very difficult. A WAF provides immediate protection from known and emerging threats in ways that traditional and next-generation firewalls do not.

WAFs increase the resiliency of web applications in several ways. Sometimes it might not be possible to modify the code of an application that contains known vulnerabilities or is known to leak confidential information. Specific rules in the WAF can mitigate these weaknesses. Many WAFs can offer configuration suggestions based on the output of a vulnerability scan report, which can help identify additional potential attack vectors that must be closed. WAFs have keen awareness of proper and improper URLs, parameters, cookies, and form values; close inspection of these elements during any web session can help identify and block malicious behavior. Over time, a WAF can recommend and apply additional policies to the baseline protection profile.

WAFs belong in front of any application that is of interest to an attacker—any application that has the potential to admit improper access to confidential corporate data, customer records, or personally identifiable information. Applications that present a logon screen are especially attractive, as credential theft enables an attacker to penetrate deep into an organization's network. A WAF can greatly increase the difficulty of successfully conducting such attacks.

Common compensating controls implemented by WAFs

At a minimum, a WAF implements controls that help mitigate the OWASP Top Ten web vulnerabilities[1], commonly seen in many off-the-shelf and in-house applications:

[1] https://www.owasp.org/index.php/Top_10_2013-Top_10

- **Injection.** Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

- **Broken authentication and session management.** Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.

- **Cross-site scripting.** XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

- **Insecure direct object references.** A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.

- **Security misconfiguration.** Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.

- **Sensitive data exposure.** Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

- **Missing function level access control.** Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.

- **Cross-site request forgery.** A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.

- **Using components with known vulnerabilities.** Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.

- **Unvalidated redirects and forwards.** Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

Traditional packet filtering and stateful inspection firewalls lack the necessary application-level awareness to address the above threats, because they focus their attention on the information in packet headers, not payloads. Next-generation firewalls often contain some degree of application awareness, but this is mostly as a means to identify potential originating applications of packets flowing through the firewall's engine. Even next-generation firewalls can't apply the kinds of controls necessary to address the above threats. The next section explores how modern WAFs implement their unique protection mechanisms.

How WAFs differ from traditional and next-generation firewalls

Most WAFs include some kind of subscription to an online service that offers a continuously updated list of attack signatures specifically tuned for addressing evolving threats against web applications. Using this list, the WAF deploys a set of protection rules (commonly called "deny-listing,"[2] a negative security model) to block any access attempts that match the signatures. Because attackers frequently modify their techniques to evade detection, it's important to keep the signature list updated as often as possible. Deny-listing rules can also block inadvertent data leakage, such as credit card numbers, social insurance numbers, and other personally identifying information.

Some WAFs also provide a mechanism for defining acceptable behavior (commonly called "allow-listing," a positive security model) and automatically denying everything else. Such rules can be applied at different levels—across the entire site, on a page or group of pages, and even on specific page elements. Policies can constrain the application behavior to follow only a defined set of workflows or execution paths. The layout of an application can be obfuscated, making it impractical for an attacker to attempt to find vulnerabilities in the received HTML. The ability to constrain an application's behavior is one area in which WAFs are especially superior to even next-generation firewalls in improving an organization's overall security posture.

It should be noted that techniques like these will take time to develop and test, and often are individually customized for each web application. A common criticism against WAFs is that the complex rule sets and the effort required to create and maintain them outweigh any protection offered. An important feature of some WAFs is a learning mode, which inspects "normal" traffic over time and constructs a recommended baseline protection profile. SteelApp Web Firewall is the only WAF on the market that extends this functionality into a "shadow mode." Here, the WAF will generate reports indicating how it would apply the rules to production traffic and, consequently, demonstrate what's allowed and what's denied, providing greater assurance that the configuration won't break applications.

SteelApp's rules are constructed by a variety of "handlers." The fundamental one is a session handler. This establishes a unique secure session between the WAF and each individual browser. A cryptographic cookie containing a session ID is transferred to the browser; every communication between the WAF and the browser contains this cookie. Two examples of the many handlers that make use of this are:

- **Cookie jar handler.** Once a session is established between a browser and a server, the WAF stores all the actual cookies associated with each browser and relies only on the session handler cookie to identify individual browsers. This renders the web server session state opaque and thwarts attempts to eavesdrop, decode, or inject malicious cookies into sessions, because the cookies no longer traverse the network between the browser and the WAF.

- **Virtualize form field handler.** This handler encrypts form variables In POST requests from a browser, using the session handler's ID as the encryption key. This technique renders field names unpredictable, thwarting a number of attacks including cross-site request forgeries. It also increases the difficulty of extracting user IDs and passwords, because they are no longer identifiable as such in eavesdropped traffic.

To implement its protection policies, a WAF must see plain-text HTML and HTTP. If a web server is encrypting traffic using SSL/TLS, the WAF must be installed on the web server so that it can observe traffic pre-encryption and post-decryption. This is not an optimal deployment choice, because a WAF is likely to impede the processing performance of the server. For this reason, a WAF is often paired with an application delivery controller (ADC), and the ADC assumes responsibility for handling SSL/TLS. Traffic between the browser and the ADC is encrypted; traffic between the ADC and the web server is plain text. The WAF on the ADC observes the plain text traffic and applies its policies.

[2] While commonly used elsewhere, this paper avoids the terms "blacklist" and "whitelist."

Finally, it's important to keep performance in mind when considering a WAF deployment. WAFs inspect both inbound and outbound traffic between browsers and servers, and at a deeper level than traditional or next-generation firewalls. Even the payloads of single packets often aren't sufficient to identify, detect, and block attacks; it's necessary to observe tens, hundreds, perhaps thousands of packets. Considerable CPU resources might be required to process the sometimes complex rules in a WAF. If installed directly on a web server, a WAF can negatively affect the performance of applications. A distributed WAF architecture like SteelApp Web Firewall separates the rule evaluation and enforcement functions. A "decider" process that runs on a farm of general-purpose servers evaluates traffic against rules. A lightweight "enforcer" module on the web server simply allows or denies the traffic, following instructions from the decider farm. As pure software, the decider process can scale up or down as needed, without any upper-limit constraints imposed by hardware appliance form factors.

WAFs and PCI-DSS compliance

Unlike traditional or next-generation firewalls, WAFs can help achieve regulatory compliance if required. For example, PCI DSS Requirement 6.6 provides two options[3] 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 options are:

- **Option 1: application code reviews.** An organization's software development life cycle should include 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. Typically, such reviews are augmented with 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.

- **Option 2: web application firewalls.** 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 payloads contained in web application sessions, 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.

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. WAFs are particularly suitable in situations where application code review is impractical or impossible.

[3] https://www.pcisecuritystandards.org/pdfs/infosupp_6_6_applicationfirewalls_codereviews.pdf

Additional information

OWASP best practices: Use of web application firewalls
https://www.owasp.org/index.php/Category:OWASP_Best_Practices:_Use_of_Web_Application_Firewalls

Gartner: Web application firewalls are worth the investment for enterprises
https://www.gartner.com/doc/2673820/web-application-firewalls-worth-investment

Riverbed: Stingray (now SteelApp) User Guide, Chapter 11: Basics of web application security
https://support.riverbed.com/bin/support/download?did=136

About Riverbed

Riverbed Technology is the leader in Application Performance Infrastructure, delivering the most complete platform for location-independent computing. Location-independent computing turns location and distance into a competitive advantage by giving IT the flexibility to host applications and data in the most optimal locations while ensuring applications perform as expected, data is always available when needed, and performance issues are detected and fixed before end users notice. Riverbed's 24,000+ customers include 97% of the Fortune 100 and 95% of the Forbes Global 100. Learn more at www.riverbed.com.