Contrast
SECURITY

EBOOK

WAF and RASP: Raising the bar for application protection

Introduction

Do all application attack attempts merit an equal alarm sound? Or, should “successful application attacks” merit a higher alarm? If that sounds like a no-brainer, then how can we sift through the attack noise and distinguish what is important and relevant to the security of our said application? What is an attack irrelevant to our said application?

This ebook is intended to provide some light reading on two technologies: Web Application Firewall (WAF) - pervasive in use in many networks across companies of all sizes and sectors; and Runtime Application Self Protection (RASP) technology - not as pervasive, but certainly more effective in sounding the right alarm, at the right time, for the successful attacks.

To understand the two technologies and why they are better together, let’s start from the beginning.

Brief descriptions of these two technologies:

What is Web Application Firewall (WAF) technology?

A web application firewall (WAF) is a network-level defense tool designed to filter, monitor, and block HTTP traffic to and from a web application. A WAF differs from a regular firewall, which typically is designed to serve as a safety gate between servers. A web application firewall is able to watch application-level traffic and makes decisions as to allow or disallow that traffic based on the data that is visible over the network. WAF security typically performs SSL termination to watch decrypted traffic for pattern-matching or volumetric attacks.

What is Runtime Application Self Protection (RASP) technology?

Runtime Application Self Protection is an application-level, server-based defense tool that operates from within the application's runtime environment, designed to monitor and analyze both application behavior and context everytime the application is used. By doing so, RASP tools control the application execution, detect vulnerabilities, and prevent real-time attacks that exploit vulnerabilities inside that application – without human intervention.

Brief use cases and limitations of WAF and RASP:

WAF: THE FIRST LINE OF DEFENSE

WAFs act as a shield at the perimeter, inspecting incoming traffic for malicious payloads associated with known attack patterns.

They offer several advantages:

- **Ease of Deployment:**

  WAFs are often deployed at the network level, requiring minimal configuration for existing applications.

- **Perimeter Defense:**

  WAFs positioned at the network or cloud perimeter are uniquely positioned to defend against Distributed Denial of Service (DDOS) attacks and known bad actors.

- **Centralized Management:**

  Security teams can manage WAF rules from a central location, simplifying security policy enforcement.

- **Protection from Common Attacks:**

  WAFs effectively block common attacks like SQL injection and Cross-Site Scripting (XSS) by identifying known attack signatures.

However, WAF also have limitations:

- **Limited Context:**

  WAFs rely solely on the content of the request (payload) for analysis, making them vulnerable to attacks that exploit application logic flaws.

- **False Positives:**

  Due to their reliance on generic signatures, WAFs can generate false positives, blocking legitimate traffic and disrupting application functionality.

- **Vulnerability to Zero-Day Attacks:**

  WAFs are ineffective against new and unknown vulnerabilities (zero-day attacks) until their signatures are added to the WAF rule set.

Brief use cases and limitations of WAF and RASP:

RASP: DEEPER PROTECTION WITH RUNTIME ANALYSIS

Runtime Application Self-Protection (RASP) complements WAFs by providing deeper application security from within. Think of RASP as a zero friction agent for production applications. It automatically hardens the runtime, the libraries, the open source software, and the app server, mitigating top vulnerability classes and stopping zero day exploits.

Here's how RASP offers a distinct advantage:

- **Behavior-Based Detection:**

  RASP monitors application runtime behavior, allowing it to detect and block attacks that manipulate application logic or access unauthorized data, even if the payloads appear benign.

- **Zero-Day Protection:**

  RASP's ability to analyze application behavior makes it effective against zero-day attacks, offering protection even before vulnerabilities are publicly known.

- **Reduced False Positives:**

  By understanding the application's logic, RASP can differentiate between malicious and legitimate behavior, minimizing fals positives compared to WAFs.

However, RASP also presents some challenges:

- **Deployment Complexity:**

  Integrating RASP with existing applications can be more complex than deploying a WAF.

- **Potential Performance Impact:**

  RASP's runtime monitoring can introduce some overhead, requiring careful optimization to avoid impacting application performance. However, Contrast Security's statistics have shown that more than 80% of all requests going through Contrast Protect (Contrast's RASP) are treated in less than 0.5 ms. At least 96% are treated within less than a single-digit milliseconds delay, making RASP just as fast, if not faster, than equivalent WAF treatment times.

THE POWER OF COMBINING WAF AND RASP:

Contrast Security's statistics highlight a critical point: over 180,000 attacks bypass WAFs every week but are successfully blocked by the Contrast Protect RASP*. This demonstrates the limitations of relying solely on WAFs.

By combining WAFs with RASP, organizations can achieve a layered defense:

- **WAFs:**

  Act as the first line of defense, blocking very obvious and common attacks and preventing malicious traffic from reaching the application.

- **RASP:**

  Provides deeper inspection as it can see how input values are being transformed by the application, where these values are being used and whether they hit a vulnerable part of an application, thus providing protection from logic flaws, zero-day attacks, and data breaches that might bypass a WAF.

This layered approach offers several benefits:

- **Comprehensive Security:**

  Addresses a wider range of threats, including known and unknown vulnerabilities.

- **Reduced Risk:**

  Minimizes the attack surface and potential damage from breaches.

- **Improved Efficiency:**

  WAFs can be configured to handle only common attacks and to let other traffic flow through without spending time on analysis. This frees up time for RASP to focus on more complex threats as RASP usually takes less time for analyzing such attacks as it sees what happens and does not have to rely on probability analysis or guesswork like a WAF would.

TWO EXAMPLES

The Deserialization Attacks Example

WAF limitations and how Contrast Security's RASP does it better

Deserialization attacks exploit a fundamental process within many applications. Applications often serialize data – converting objects into a format suitable for storage or transmission. Later, this data is deserialized back into objects for use by the application. Attackers can craft malicious payloads that, when deserialized, trigger unexpected or harmful actions within the application.

- **The Challenge for WAFs:**

  WAFs primarily analyze the content of incoming requests. They might spot an unusually structured data payload, but often lack the context to understand how an application will process that data during deserialization. This means attacks that exploit application logic flaws in the deserialization process can slip through a WAF's defenses.

- **Real-World Example:**

  An attacker might craft a payload that, when deserialized, executes arbitrary code on the server (remote code execution). A WAF, analyzing the payload itself, might see nothing inherently malicious. It wouldn't understand that the deserialization process would lead to the dangerous execution of attacker-supplied code.

Understanding the nature of a serialized object is only possible within the application. Only there is the content revealed and can be analyzed based on its true nature and its usage by the application. The instrumentation-based approach provides:

- **Runtime Insight:**

  Contrast Protect operates within the application, monitoring its behavior at runtime. During deserialization, it observes how the data is processed – where it flows within the application and what actions it influences.

- **Behavioral Anomaly Detection:**

  If the deserialized data triggers suspicious or unexpected actions, Contrast Protect recognizes these as potential attacks, blocking the request from causing further harm, and alerting security teams.

- **Clean Exception Handling:**

  Contrast Protect can neatly integrate with the application's error handling. If an attack is detected, it generates an exception, just like the application would for invalid input. This smoothly halts the abnormal usage, allowing the application to respond gracefully.

TWO EXAMPLES

The Log4Shell Example

The widespread Log4Shell vulnerability (2021) was a prime example. Attackers exploited a weakness in the popular logging library to inject malicious payloads that could take control of systems.

- **WAF Limitations:**
  Until specific signatures were released, WAFs had little chance of stopping Log4Shell attacks.
- **RASP Advantage:**
  Contrast Protect had defenses in place since 2018. Its dataflow analysis capabilities allowed it to identify the abnormal Log4Shell payload behavior during execution, preventing exploitation even before the vulnerability became public knowledge.

What security issues did the Log4Shell vulnerability expose?

Log4j exposed many things when it comes to security issues:

- Organizations struggled (and are still struggling) to find where Log4j was used. They do not have accurate asset management or Software Bills of Materials (SBOMs) of their systems to provide them with the ability to quickly identify problem areas.
- Once organizations identified all their Log4j instances, they were not sure where to start to upgrade, and in most cases found themselves waiting on vendors to upgrade the systems first.
- Only 50% of Log4j has been updated in one year's time. It is still hard to manage third party libraries even when we know there are egregious and critical vulnerabilities.
- The biggest security exposure was the fact that most organizations have ZERO projection when a zero day is released [up until] when they can patch it. Even patching a WAF after a couple of days is minimally effective if it can or already has been bypassed. Organizations must protect their applications from these unknown risks through the use of runtime protections.

Given that the Log4j library is used in close to two-thirds of Java applications ... there is no reason for attackers to stop targeting it. They'll be able to successfully exploit the vulnerability in many cases.

- David Lindner, CISO at Contrast Security

Conclusion

Most security teams see Web Application Firewall as a valuable security tool. However, as guardians of the security of our applications and indeed as guardians of our brand, reputation and business revenue, we need to understand that a WAF might not be enough to detect and stop successful attacks.

A Web Application Firewall can watch data that goes over the network, but its architecture does not enable them to see how that data is actually used. As a result, they sound an equal alarm for all attack attempts without raising the importance for attacks that could actually work. This wastes human effort on investigating issues that don't matter, and results in constant manual tuning or auto-tuning where success depends on information that WAF's simply do not have.

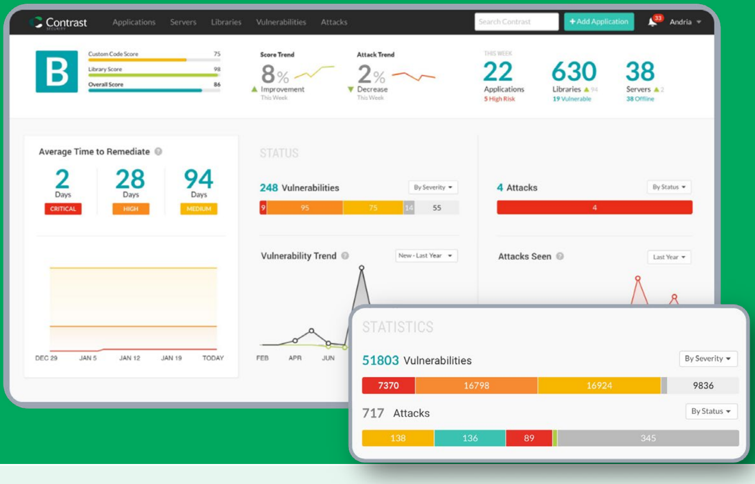
RASP provides a more effective application attack protection derived from actual monitoring and analysis of the application behavior and context everytime the application is used. By doing so, RASP tools control the application execution, detect vulnerabilities, and prevent real-time attacks that exploit vulnerabilities inside that application – without human intervention.

By combining WAF and RASP, organizations can achieve a more comprehensive and future-proof application security posture, significantly reducing the risk of successful cyberattacks.

If you want to learn more or see Contrast Protect - your RASP solution in action, visit **www.contrastsecurity.com** and click on the "Get DEMO" button on the top of every page to schedule a personal introduction to Contrast..

SPOTLIGHT

Contrast Protect with Runtime Application Self Protection



CONTRAST PROTECT PUTS PROTECTION WHERE APPLICATIONS NEED IT MOST

As a complement to WAFs and other traditional perimeter defenses, Contrast Protect is deployed within the application runtime, providing visibility, accuracy, ease of deployment, and instant scalability for applications. Contrast Protect helps organizations protect application vulnerabilities from both internal and external attacks in real time and eliminates the false positives that squander security staff resources.

The time is now to begin using a modern approach to blocking unknown threats that frequently bypass perimeter solutions. Here, Contrast Protect allows security teams to keep their focus on the real and critical risks to running applications instead of chasing false positives. Even further, it keeps organizations compliant with emerging industry standards for effective and modern application security.

Contrast Protect works wherever the application runs—in the data center, cloud, or container. This always-on, embedded simplicity greatly reduces setup effort and costs—which, in turn, enables development teams to move more quickly and re-architect solutions without compromising on security.

CONTRAST PROTECT ENABLES COMPLIANCE WITH MAINSTREAM STANDARDS

Maintaining compliance with the latest industry standards and government regulations helps organizations keep pace with an evolving threat landscape and adhere to minimum best practices for security and network infrastructure—including deployed AppSec defenses.

Contrast Protect helps organizations comply with mainstream industry standards such as the National Institute of Standards and Technology (NIST) and the Payment Card Industry Software Security Standard (PCI-SSS). NIST standards are used for things such as measuring equipment and procedures and quality control. PCI-SSS are new requirements for the secure design and development of modern payment software. RASP technology is already a requirement in NIST 800-53—which covers recommended security-control selection—and already a requirement in PCI-SSS 9.1, 10.2a, and 10.2b which defines security requirements to ensure payment software protection.

* The source of the data are Contrast Security's SaaS environments, additional blocked attacks are reported to on-premise-based, customer-hosted servers which are excluded from these statistics.

240 3rd Street
2nd Floor
Los Altos, CA 94022
Phone: 888.371.1333

Contrast SECURITY

contrastsecurity.com