PREVOTY

A Quick Guide to Runtime Application Self-Protection (RASP)

Prevoty, Inc. HQ
11911 San Vicente Blvd #355
Los Angeles, CA 90049

prevoty.com
info@prevoty.com
310.499.4983
@prevoty

How RASP Complements the Security Ecosystem

Gartner first defined runtime application self-protection (RASP) as a security technology built or linked into an application runtime environment to control execution and prevent real time attacks.[1] Before RASP entered the security market, the industry's offerings provided protections on the network layer and on the host, but lacked active protection at the application layer. With the exception of a WAF, there were no production environment protections to provide controls at runtime. With 28 technologies and growing, the application security space isn't trivial -- nor is it well defined.

RASP is an emerging tool that typically falls under the "Runtime Testing and Protection" or "Application Self Protection" category. As a newcomer, RASP is important not only in its own function but also in how it differs from and/or interacts with the other technologies in the ecosystem. Unique points:

- The RASP approach differs from traditional tools because it is tightly coupled with the application code traditionally susceptible to malicious exploit and operates in production.
- RASP can automatically adapt to any language or environment and uses contextual awareness -- not blacklists and whitelists -- to detect threats.
- Instead of blindly guessing that a particular payload will (or will not) be able to exploit an unknown part of the application code before the data is sent to the application, RASP technology inspects the complete (and oftentimes transformed data) in the context of how the application will use it -- if and only if the application will attempt to use the data.

No matter what, RASP supplements and even improves the effectiveness of other tools. **And most of the time, modern RASP technologies are easier to deploy, provide a more uniform set of controls regardless of programming language, and perform with higher accuracy.** The following table aims to define the various technologies in the application security ecosystem that potentially interact with RASP, and how:

1 Gartner IT Glossary: Runtime Application Self-Protection (RASP). http://www.gartner.com/it-glossary/runtime-application-self-protection-rasp

Technology	Interaction
Web application firewall (WAF) or next-generation web application firewall	Since most firewalls are inserted in the data and control path as passive elements to ensure compliance, RASP offers a critical layer of attack prevention behind a WAF or next-generation WAF. WAFs monitor and block traffic by applying rules. Several next-generation network firewall solutions (Dell Sonicwall, Palo Alto Networks) include some basic WAF functions and/or application awareness as they sit as a bump in the wire on the architecture. Since most firewalls use patterns and heuristics, they offer limited assurance against application attacks. Rather, they are useful as the first layer of defense. RASP is not a proxy nor does it block traffic; instead, it neutralizes malicious or malformed payloads and specific inputs to serve as a last line of defense. Because it sits within the application, it has access to attack details, including unsanctioned database activity. Leading solutions from vendors like F5, Imperva enable this today and view RASP as a complementary technology. See more in "WAF vs. RASP" on Page 8.
Dynamic application security testing (DAST)	Leading DAST solutions like that of Whitehat Security provide visibility into vulnerabilities. The interaction between DAST and RASP is simple. RASP can be used to prioritize vulnerabilities before running any testing tool, guiding developers on how best to minimize risk and ensuring effective secure coding practices. RASP can be also be installed on an application in production and turned on in protection mode. Attacks and abnormal inputs are cleaned and mitigated in real time, and proof of runtime threat activity is reported to a dashboard like WhiteHat Security's Sentinel.
Security incident and event management (SIEM)	RASP sits inside the application and enriches attack data with critical insights into the where any transformation or exfiltration attempt took place, where, and by what bad actor, greatly improving a SIEM's security analytics with runtime intelligence. Most leading vendors like Splunk allow for logs and files to be visually displayed as well as run through data analytics engines to determine patterns. All RASP solutions should have the ability to generate monitoring and protection logs in the following formats: CEF, LEEF and JSON. The first two target SIEMs like ArcSight, QRadar and Nitro. By generating JSON logs, a RASP's output can be ingested by more modern/flexible SIEMs such as Splunk or even Elasticsearch.

RASP Value & Use Cases

RASP entered the market as a progressive alternative to the application security status quo. So what problem(s) does RASP solve, and how? Below, we outline RASP's main values to the enterprise:

SMARTER RESPONSES

RASP Value & Use Cases

Response teams do not have insight into application security events in production and thus cannot accurately correlate pre-production vulnerability findings with runtime attack data. Furthermore, there is even more limited visibility into access or exfiltration attempts for applications and databases moving to the cloud. There is also a significant amount of noise generated by testing tool results, application firewall activity, and vulnerability reports. RASP can help security operations and application development teams filter through the noise and better allocate resources using runtime intelligence.

RASP delivers correlated network, application and database security logs for smarter, faster responses and powerful visibility into an organization's actual runtime exposure to risk. Application security monitoring using RASP is a new capability that has been designed to give enterprises the ability to determine which applications are actually under attack in real time (and how) -- effectively improving risk management and remediation efforts. In short, application monitoring via RASP answers the following questions:

Who

Identify the origin of the threat

IP address, session info (with user ID), cookie

What

Provide details of the nature of the threat

Contents of the payload, payload intelligence

Where

Where the exploit happened

URL for web applications, stack trace for SQL queries

When

When did the attack take place

Timestamp (down to the nanosecond)

Specifically, new application-level insights and forensics can also catch authentication, authorization and transactional fraud. Detailed information on all database queries issued by specific applications allow for detailed audit trails and support root cause analysis for data breaches.

BETTER DEFENSES

Real-Time Vulnerability Mitigation

Remediation efforts are unable to verify and mitigate 100% of application security vulnerabilities found in the secure software development life cycle. Nevertheless, enterprises are often pushing applications into production with known vulnerabilities that cannot be remediated due to a lack of access to the code base, legacy frameworks, and other roadblocks. These vulnerability exception procedures can be costly and extremely risky.

RASP implementations are uniquely positioned to help enterprises protect applications at runtime, neutralizing known vulnerabilities and protecting against previously unknown threats and zero-day attacks. Depending on the nature of the deployment, RASP can also transform or block content and database queries so that everything the application processes is safe.

Many organizations use RASP to embed a last line of defense that travels with the application, whether in the cloud, on-premise, in pre-production or in production. As an automated, technical control for compliance requirements, RASP takes the pressure off of development by performing real-time vulnerability mitigation.

FASTER RELEASES

Scalable, DevOps-Friendly Application Security

Until RASP, application security and DevOps were frequently at odds. The increasingly distributed, agile nature of application development and deployment has made preventing a breach complex and challenging. Virtualization, containerization, microservices and the cloud make applications and data ubiquitous -- impossible to monitor consistently and accurately across different platforms and environments. Worse yet, pre-production testing requirements create bottlenecks in the software development process. Some vulnerabilities can even block release, which can be problematic in a rapid-release, Continuous Integration (CI) or Continuous Deployment (CD) DevOps cycle.

Some RASPs can be woven directly into the DevOps build/deployment process so that applications can safely and be deployed into production without any delays. RASP's detection and prevention features can be embedded directly into every application release as part of the automated CI/CD pipeline, which means applications can automatically self-protect no matter where it sits in the SSDLC. Security tests are no longer tied to release schedules and remediation is prioritized using production attack data.

Common Use Cases for RASP

These following scenarios showcase more specifically how security admins, DevOps teams and developers use RASP to address some of the challenges facing application security today:

Use Cases
1 - Reduce vulnerability backlog
2 - Real-time visibility into production attacks
3 - Faster application releases with scalable DevSecOps
4 - Provide Runtime Intelligence for DevOps
5 - Protect legacy applications
6 - Last line of defense in a layered security model
7 - Optimize the Secure Software Development Lifecycle (SSDLC)
8 - Improve Security Operations & Response
9 - Protection for applications anywhere & everywhere
10 - Reduce AppSec risk & increase compliance

Common Use Cases for RASP

These following scenarios showcase more specifically how security admins, DevOps teams and developers use RASP to address some of the challenges facing application security today:

Cross-Site Scripting

Cross-Site Request Forgery

DOM Cross-Site Scripting

SQL Injection

XML Injectlon

JSON Injection

Database Access Violation (advanced SQL Injection)

Command Injection

Link Spam

XML External Entity Injection

Weak Authentication (Basic Auth)

Broken Session Management

Weak Browser Cache Management

Logging Sensitive Information (credit card numbers and email addresses

Insecure Transport

Protocol Uncaught Exception

Insecure Direct Object References

Security Misconfiguration

Sensitive Data Exposure Unvalidated Redirects and Forwards

HTTP Response Splitting

HTTP Method Tampering Unvalidated Redirects

Path Traversal

Unauthorized Markup (trying to inject prohibited HTML tags)

Unauthorized Media (trying to inject links to prohibited media sites)

RASP in Action: Passive and Active

All RASPs should function in two different yet complementary modes: monitoring and protection.

1. When in passive monitoring mode, a RASP solution should utilize very limited application resources such as CPU and memory (RAM). It should also add minimal latency. While monitoring, a RASP should be able to generate similar logging events as if it were in passive mode. This allows organizations to build or access a security analytics report or "heatmap" of where real-world attacks are hitting the application, with little to no false positives.

2. When in active protection mode, a RASP solution should still utilize limited application resources to detect threats while automatically mitigating attacks in real-time and preventing database exfiltration. It should not require significant resources to tune or configure, or require cumbersome rule sets or definition lists. It should add minimal latency to an application. While in active mode, a RASP should generate actionable intelligence about real-world attacks, as well as what action was performed to neutralize the malicious or malformed payload.

Conclusion

This guide was designed to help security decision-makers think critically about the capabilities of RASP solutions, compare and contrast RASP against other tools (e.g. WAFs, penetration testing, application security testing, etc.), and come up with questions to ask vendors. Not all RASP solutions are created equal; each provides different levels of visibility, performance, scalability, accuracy, and ease of implementation / maintenance.

Keep in mind that RASP originated as a solution not simply to test for application security risks, but to mitigate real-time threats to production applications. It has also evolved to provide powerful capabilities for database monitoring and application attack visibility for improved forensics and faster remediation. As the category expands, so does the range of effectiveness. It's important to carefully consider each solution's capabilities to ensure applications are protected with no impact on operations or performance and plenty of vendor support.

Key RASP differentiators include: methodology, coverage, speed, and ease of deployment. For instance, in today's DevOps-enabled business climate, new fully-automated RASP technology that plays nice with Continuous Integration / Continuous Deployment cycles and reports real-time, actionable security analytics have a significant edge over traditional RASP tools. Therefore, it's important to judge a RASP's capabilities as it relates to each enterprise's unique challenges and use cases while considering the key differentiators. We recommend that readers use this document as a tool to shape their research when evaluating competitive offerings.

To read the **full, unabbreviated Guide to RASP** which contains technical components of RASP (including different methodologies), compliance information, specific evaluation criteria, core requirements and a scoring rubric for comparing RASP technologies, please visit into.prevoty.com/guide-to-rasp.

RASP Will Continue to Evolve

RASP technology is a powerful and innovative component in making applications more secure given increasing software deployments and distributed application architectures. With RASP, we are finally able to address attacks in production and explore ways to embed security functions into the applications themselves. Once a nascent category undergoing development and expansion, RASP has gained momentum as a viable, enterprise-grade security solution and a popular choice for achieving DevSecOps alignment. Enterprises using RASP are currently unlocking powerful insights and making smarter application development, security operations and vulnerability remediation decisions.