Publication date:
29 Jan 2026
Author(s):
Rik Turner, Chief Analyst, Cybersecurity

On the Radar: Rein Security offers a library for app runtime context

Summary

Catalyst

Rein Security offers an application security platform, delivered in software-as-a-service (SaaS) mode without requiring the deployment of software agents and designed to provide the context necessary to address security issues detected within an application, as well as the controls to do so.

Omdia view

While application security (AppSec) is a longstanding part of the cyber landscape, it has undergone dramatic changes in recent years in response to broader movements in technology such as cloud computing, new application development (AppDev) practices, and, more recently, the rise of new forms of AI. These trends mandated new ways of delivering AppSec and, given the continued evolution of computing technologies generally, the pressure on the security sector to keep up will, in all likelihood, only increase.

In this environment, the advent of an AppSec platform offering broader visibility, specifically bringing in application runtime context, is a salutary development that should not only attract the attention of developers and SecOps teams, but also spur other vendors to come up with new ways to address the need to secure application environments. Omdia expects to see Rein pick up customers and make headway in this increasingly crowded market.

Why put Rein on your radar?

Rein's application security platform offers easy, configuration-free deployment and delivers broad visibility into an application's runtime context, enabling both proactive (cf. Application security

posture management (ASPM)) and reactive (ADR) security capabilities. It can be deployed in observe mode, then switched to enforcement once the customer is comfortable with its capabilities.

Market context

A dominant theme running through the history of AppDev in the first quarter of this century has been acceleration. We can point to at least two major trends or "movements" that contributed to the speeding up:

- The Agile development process replaced rigid, linear processes based on extensive documentation with iterative cycles. This resulted in faster time-to-market and increased adaptability to changing requirements, as well as boosting team productivity and enhancing collaboration.
- Meanwhile, the emphasis on componentization and reusability in the service-oriented architecture (SOA) movement led to a fundamental change: AppDev went from a process akin to the work of a medieval scribe, laboring at a desk to write out an entire document from beginning to end in longhand, to something more like a child constructing a building with Lego bricks.

The security issues raised by the need for speed

The combination of these two approaches has certainly brought about a huge increase in the speed with which apps are brought to market, but it has also had implications for security. Most of the "bricks" nowadays used in assembling application code are open-source components (i.e., code snippets and libraries), which are the result of the work of informal communities and often have a volunteer maintainer. These initiatives lack any governance structure or accountability, and as such, the presence of vulnerabilities in the code and the amount of time they may remain there are entirely aleatory.

The rise of software composition analysis (SCA)

This situation led to the emergence of a new software security discipline. While application security testing (AST) tools had been in use since the 1990s in both their static (SAST) and dynamic (DAST) variants, SCA arose in the 2000s as an offshoot of SAST, designed specifically to find and catalog all the open-source components in an app, gauge their vulnerabilities and prioritize them for remediation based on their criticality.

The need for API security

The enterprise application environment has also become more complex since the turn of the millennium, with application programming interfaces (APIs) now enabling countless interactions between systems (apps, databases, web services, libraries, and so on).

Here too, there are security concerns, both with the APIs that an organization's developers are writing themselves, where vulnerabilities may crop up, but also with other, third-party APIs their apps will use once in production.

Infrastructure-as-code further expands an app's attack surface

Meanwhile, AppDev has undergone further acceleration thanks to the emergence of infrastructure-as-code (IaaC) platforms to automate server provisioning in a cloud computing context. IaaC is yet

another place where vulnerabilities can be introduced, and so it also requires the code to be inspected for security purposes.

Enter vibe coding, agents, and Model Context Protocol (MCP)

Over the last year, we have seen yet another new actor come onto the scene, namely the ability to generate app code using AI. The vibe coding trend spells further acceleration in AppDev, of course, but also an additional source of security concerns, since the code generated at still greater speed currently can and does contain as many vulnerabilities as the human-created variety.

Moreover, the rise of agentic AI tends to further complicate the situation, given the degree of access and autonomy that can be granted to agents. A badly configured agent, or indeed a compromised one, introduces a new type of risk into the environment.

The advent of MCP servers, through which agents gain access to external resources such as local files, databases, web APIs, and development tools in real-time, only compounds the problem. MCP servers enable access to multiple types of resources, including ones that the organization deploying them does not control.

Product/service overview

The originality of Rein's approach to the challenges of both proactive and reactive AppSec is in the way it has engineered its platform to run not as a separate process from the application, but rather as part of the execution process itself. This approach brings a range of benefits over alternative technologies, including:

- Technology that lives in the kernel space of the operating system and relies on kernel access to understand what is happening in the environment, such as endpoint detection and response (EDR), is susceptible to attack by threat actors, as well as constituting a single point of failure (as evidenced by the non-malicious CrowdStrike incident of July 2024).
- The much-vaunted eBPF technology, which enables security vendors to run programs in the sandboxed, privileged context of the operating system kernel, lacks full visibility, in that it does not see how the application code is executed, nor what HTTP requests are coming in. In other words, it provides only the kernel perspective, lacking the broader context of the app runtime. As such, it cannot underpin a stack trace (i.e., it is unable to provide a detailed report showing the sequence of function or method calls a program made leading up to an error or crash). The fact that Rein can show the executed functions is of particular importance. Whereas best-in-class eBPF-based reachability platforms show whether a library is loaded into memory, Rein argues that its platform goes two steps further, taking both function and actual execution into account. Rein also points out that it can and does block malicious attempts to exploit vulnerabilities.
- ASPM platforms have gained popularity over the last couple of years due to their aggregation of multiple AppSec capabilities. They tend to use interactive or dynamic app security testing (IAST or DAST) tools to simulate, aka mimic how an app might behave, but this is not the same as seeing how it actually behaves at runtime, because it does not see 100% of requests, responses, and resources called. IAST and DAST observe how an app behaves, but Rein argues that this is still not full visibility.

Rein makes the point that, unlike EDRs, it does not require the deployment of agents. These are pieces of software that operate as external systems, analyzing the behavior of a host system (e.g., an application) and often feeding information to a back-end platform, where decisions are made regarding the performance or security of the host. If allowed by the team that deployed them, EDRs can also take response actions based on decisions made by the back-end platform.

The library/dependency approach

Instead of agents, Rein deploys a library (i.e., a pre-written combination of code, functions, scripts, and routines that resides in the application's user space without privileges and the ability to intrude on the kernel space). This library becomes a dependency (i.e., it forms part of the application rather than operating as an external process). Rein offers three potential deployment modes for the library:

- Via the CI/CD pipeline, where it can be added to a company's configuration package, such as Dependabot, enabling all applications to be updated automatically to contain it.
- Within a framework for package delivery, such as Cantina in Node.js.
- In the base package that a developer uses to build an application.

The central thesis behind the development of Rein's platform is that, rather than prioritizing endless findings based on generic context (Common Vulnerability Scoring System (CVSS), etc.), the objective is to know what is impactful, which Common Vulnerabilities and Exposures (CVE) is actually executed, whether it is executed by a public-facing API, and so on. While some of this can be achieved with IAST or DAST platforms, they essentially predict how the app/API/code will behave, whereas Rein knows because it sees how they actually execute.

The Rein library is added at the build phase of an application's lifecycle rather than in the development phase. It is available as a library in any number of languages, including NPM and Yarn, that a customer can initialize or call, with no configuration, privileges, or software development kit (SDK) required, and is designed to:

- Carry out the discovery of all available resources, including shadow and zombie APIs within the environment
- Track all asset usage
- Inspect all HTTP requests, as well as determine which piece of code is executing that action
- Determine which libraries are reachable or unreachable.

Baselining components and APIs

This information provides the context for AppSec decisions. It enables teams to baseline libraries and APIs, as well as to gauge an app's posture and triage alerts by their urgency, building an "allow list" for certain types of behavior and thus providing an additional control for existing vulnerabilities.

Rein can be used for security posture (i.e., proactive security), to guarantee that an app does not deviate from its expected behavior, as well as for app detection and response (ADR, aka reactive security), determining whether behaviors observed in the production environment constitute malicious activity and recommending or actually taking the necessary remedial action.

Company information

Background

Rein was co-founded in 2024 by CEO Matan Bar Efrat and CTO Netanel Rubin. Both men are alumni of Israeli military intelligence (aka Unit 8200), where Bar Efrat ended his time as Cyber and Data Product Manager, while Rubin was a security researcher. Bar Efrat went on to be Director of Sales and Business Development for Europe at cyber range vendor Cyberbit, then held the same role at defence contractor Elbit Systems.

Meanwhile, Rubin went to security heavyweight Check Point as a vulnerability researcher, then became CTO for five years at cyber consultancy Holcy, which was acquired by security software vendor Lab42 in 2021, where he rose to become CTO, staying till mid-2024.

Current position

Rein identifies three major use cases for its platform in new accounts:

- API security, where traditional providers of runtime security for APIs look at network data, an approach that does not factor in the impact of a request on the application and its resources. Rein argues that the fact that such products lack execution context further limits their detection capabilities regarding malicious actions. It further notes that, whereas they guess which APIs exist within an app based on network sampling, its platform actually sees the routes traffic is taking and can therefore know which APIs are present.
- SCA, where there is a need to address first- and third-party code, but now also agentic code, finding all the libraries that are vulnerable, those that have been executed, and those that have been actively exploited (i.e., they have a CVE associated with them).
- MCP and agentic security, which offers both visibility and detection and response capabilities.

In the first two cases, Rein is often competing against and even replacing existing tools, such as Salt Security or Akamai (in API security) and Snyk (in SCA). The third is almost virgin territory, given the relative newness of agentic AI generally and MCP in particular.

That said, there are already vendors offering MCP gateways to secure these servers, though their visibility is limited to incoming prompts, which leaves them blind to a number of ways in which problems in an agentic environment can metastasize. Rein, by contrast, has the ability to baseline and understand the two most critical parts in the non-deterministic world ushered in by AI: not only what the prompt looks like, but also who performed the action and what its impact is. In other words, it has the necessary context, knowing which resources are accessed and by whom, which in turn allows for better baselining and protection. The Rein platform can also block internal MCPs from accessing undesirable external resources.

In any case, Rein's strategy is to land and expand (i.e., gain a foothold in an account, then take on other AppSec requirements as the customer becomes more familiar with its capabilities).

The charging mechanism for the Rein platform is a combination of a per-API endpoint fee and usage-based charging (i.e., the number of APIs and the number of requests they handle).

Key facts

Table 1: Data sheet: Rein Security

Product/Service name	Rein	**Product classification**	Application security
Version number	n/a	**Release date**	January 2026
Industries covered	The platform is a horizontal offering, with current customers in financial services, insurance, critical infrastructure, and B2B SaaS	**Geographies covered**	North America, EMEA
Relevant company sizes	Large and midmarket enterprises	**Licensing options**	Annual subscription, changed on a per-API endpoint basis, with usage-based licensing
URL	https://www.reinsec.io/	**Routes to market**	Currently direct, with plans to develop a channel in 2026
Company headquarters	New York, NY, US and Tel Aviv, Israel	**Number of employees**	24

Source: Omdia

Analyst comment

As mentioned earlier, the AppSec market is a crowded space characterized not only by a myriad of abbreviations describing different technical capabilities (e.g., the xAST spectrum, SCA, ASPM, ADR, etc.) but also a continual stream of new start-ups vying for attention, not to mention funding and customers.

Part of the challenge facing Rein, therefore, is to make its voice heard over the cacophony in the AppSec market. This can be done through thought leadership via blogs, whitepapers, webinars, conference presentations, and so on, but also by signing up poster customers who are prepared to go on the record to explain why they chose Rein.

Beyond that, there is also the need to articulate clearly what the Rein platform does and where it fits in a security team's arsenal. Part of the challenge here, of course, is that it is a multi-purpose product

that spans a variety of use cases, and Omdia applauds the vendor's decision to highlight three specific cases to start with.

Rein must avoid pigeonholing as an SCA, API, or indeed an agentic AI/MCP security platform, even while making clear that it can cover all three of those requirements.

The vendor is similarly wise to avoid being classed as an ASPM platform. Even though such products promise a comprehensive "Swiss army knife" of AppSec features and functions, they are fundamentally different in how they operate compared to Rein, such that it is better for the latter to position itself as an alternative to their approach.

Appendix

On the Radar

On the Radar is a series of research notes about vendors bringing innovative ideas, products, or business models to their markets. On the Radar vendors bear watching for their potential impact on markets as their approach, recent developments, or strategy could prove disruptive and of interest to tech buyers and users.

Further reading

Content Security Gateway Appliances, Software, and SaaS Market Tracker – 3Q25 Analysis (January 2026)

Enterprise Cybersecurity Operations (SecOps) – Half-yearly analyst update call – 2H25 (January 2026)

"Dawn of the mega-platform: Palo Alto Networks' Cortex Cloud spans full continuous security protection lifecycle" (February 2025)

Market Landscape: Application Security Posture Management (ASPM) – An Update (August 2024)

Author

Rik Turner, Chief Analyst, Cybersecurity

askananalyst@omdia.com

Citation policy

Request external citation and usage of Omdia research and data via citations@omdia.com.

Omdia consulting

We hope that this analysis will help you make informed and imaginative business decisions. If you have further requirements, Omdia's consulting team may be able to help you. For more information about Omdia's consulting capabilities, please contact us directly at consulting@omdia.com.

Copyright notice and disclaimer

CONTACT US

omdia.com

askananalyst@omdia.com