What is a Runtime CNAPP Anyways?

A Deep Dive on Sweet Security

James Berthoty | Latio Tech

This post was completed in collaboration with the team at Sweet Security who let me use their product to show their runtime CNAPP capabilities and asked me to speak honestly about the pros and cons of the platform. Also, to see a demo of Sweet and many other Cloud Security tools, tune in to the Cloud Security Showdown today!

It's exciting to see Cloud Native Application Protection Platforms (CNAPPs) evolve beyond their endless posture management (CSPM) origins. The idea of a runtime-first CNAPP is appealing to anyone **who expects their cloud security solution to actually detect and stop ongoing attacks** rather than focusing on only the discovery of posture related issues and potential vulnerabilities.

In 2025, endless posture scanning is played out and security teams need ways to reduce noise and make cloud security alerts actionable. That concept of runtime actionability has implications for both posture and runtime detection events. I suspect people will use these new breeds of runtime oriented solutions as either augments to their existing CSPM solutions, or as standalone solutions, depending on the specifics of their environment and coverage capabilities.

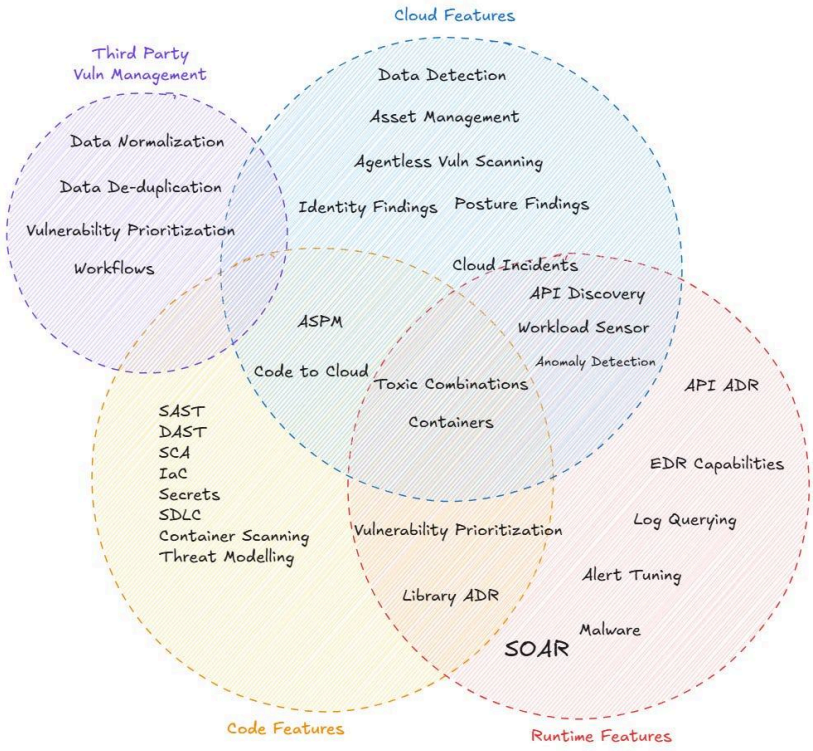
In 2025, endless posture scanning is played out and security teams need ways to reduce noise and make cloud security alerts actionable.

In this report, we'll first talk about where "runtime CNAPPs" fit in the overall landscape of security products, then discuss what makes Sweet Security's offering exciting for cloud security, and finally, what use cases would not be a good fit.

What is Sweet Security?

Sweet Security is positioned in the growing cluster of runtime-first CNAPP solutions, with capabilities growing into the category I refer to as Cloud Application Detection Response (CADR). **Sweet is meant to enable meaningful detection and response in cloud environments.** They stand out for how they approach detection and response, especially when it comes to reducing alert noise and helping SOC teams understand what actually happened during an incident. Additionally, their detection methodology is genuinely unique among anomaly based detections, with some pros and cons.

Where Sweet Fits in the CNAPP Landscape

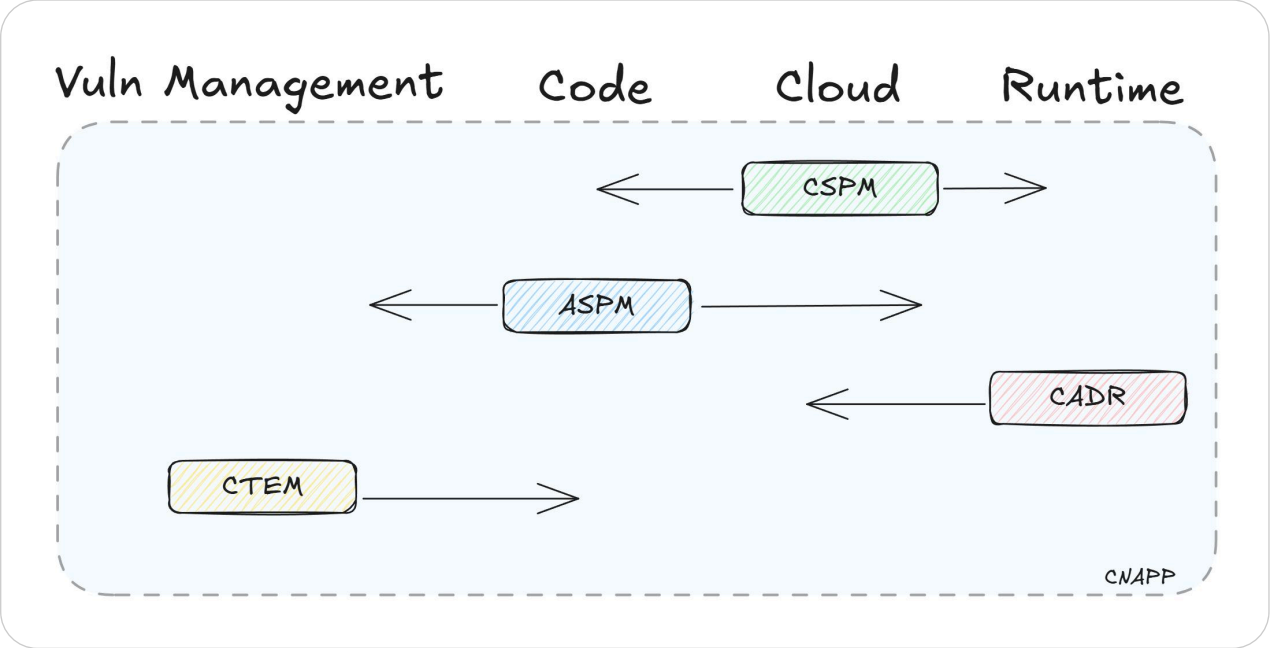


The wonderful world of CNAPP feature checkboxes

Before getting into the product specifics, it's worth briefly reviewing the broader CNAPP category. Many platforms in this space attempt to unify cloud posture management (CSPM), vulnerability management (CTEM), code scanning (ASPM), and runtime protection (CADR). **The problem is, most of them either spread themselves too thin, or bolt on features that lack depth.** Larger platforms have long failed to be the single pane of glass for everything, and that has led to weaker performance at the edges. **As a result, many teams end up pairing specialized tools rather than relying on single platforms.** The go-to example of this is Aqua (runtime) and Orca (posture) having a partnership, despite offering the same features on paper.

Sweet fits into the landscape as a strong option on the runtime side. It's not trying to be everything from code to cloud to third party vulnerability management, but instead offers meaningful detection capabilities, vulnerability discovery & prioritization, and API security capabilities.

Personally, I'm a fan of this approach, and what is becoming a new category in Cloud Application Detection and Response (CADR). Instead of trying to pretend that all of these offerings are one massive tool (CNAPP), **we can acknowledge that we're really talking about four distinct clusters of capabilities,** each of which are built for different security users. Every vendor chooses particular features from these clusters to implement, but it's overly reductionistic to refer to them all as a single thing and assign a grade.



At the end of the day, securing large complex cloud systems requires some combination of four capabilities:

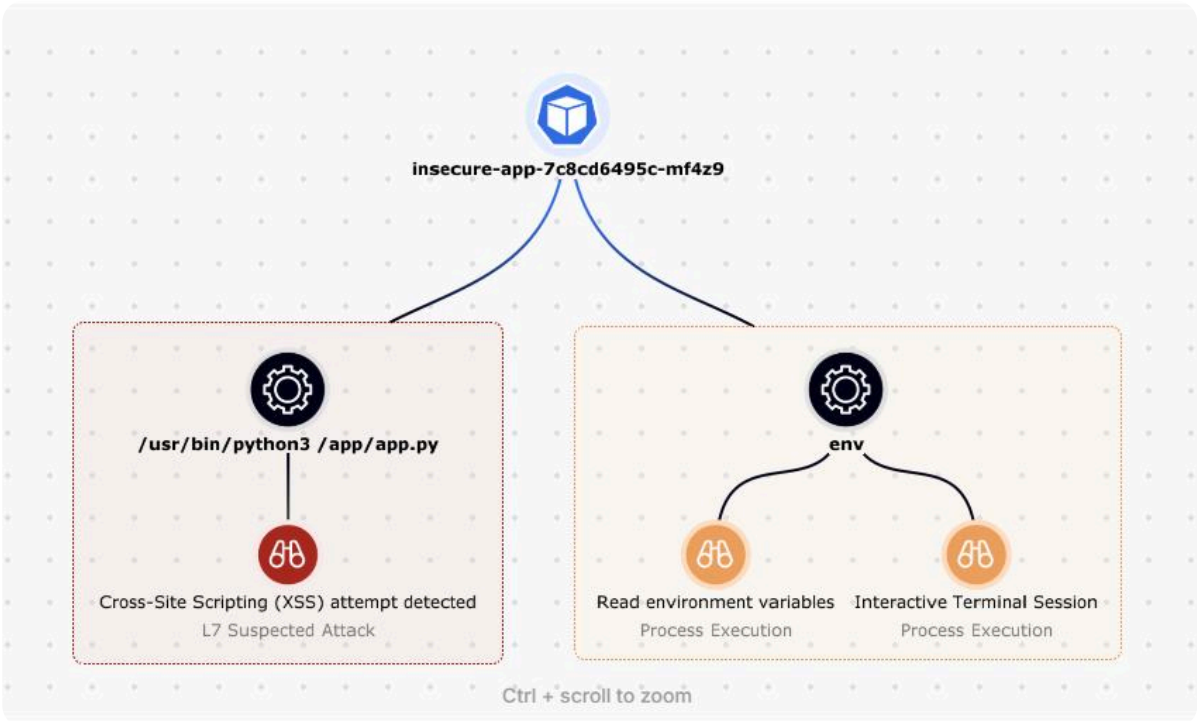
1. The ability to deeply scan and contextualize their **code findings (ASPM)**. These tools are for helping **developers** fix issues with code.
2. **Visibility of cloud assets** with information from vulnerabilities to technologies being deployed **(CSPM)**. These tools are for helping **cloud engineers** fix misconfigurations in their cloud environments.
3. Best in class **cloud workload protection (CADR)**. These emerging tools are to make cloud security operations achievable - whether for traditional **SOC teams** or emerging product security incident response teams.
4. A place to **consolidate vulnerability data** for reporting and actioning (CTEM). These are for **vulnerability management teams** in large distributed environments. Whether or not this should be merged with CSPM I'll leave up to the reader, I could go either way.

Unfortunately, CNAPP just means some amount of these four things is happening, which is why I prefer keeping the acronyms distinct. But in this case, I use Runtime CNAPP and CADR interchangeably - enabling best in class cloud workload protection will come with some CSPM functionalities that makes the tools competitive.

Whether or not this should be merged with CSPM I'll leave up to the reader, I could go either way.

A brief note on identity and DSPM, I usually lump these features up into CSPM because they are extensions of the same data, but it's fair play to argue for them as separate categories.

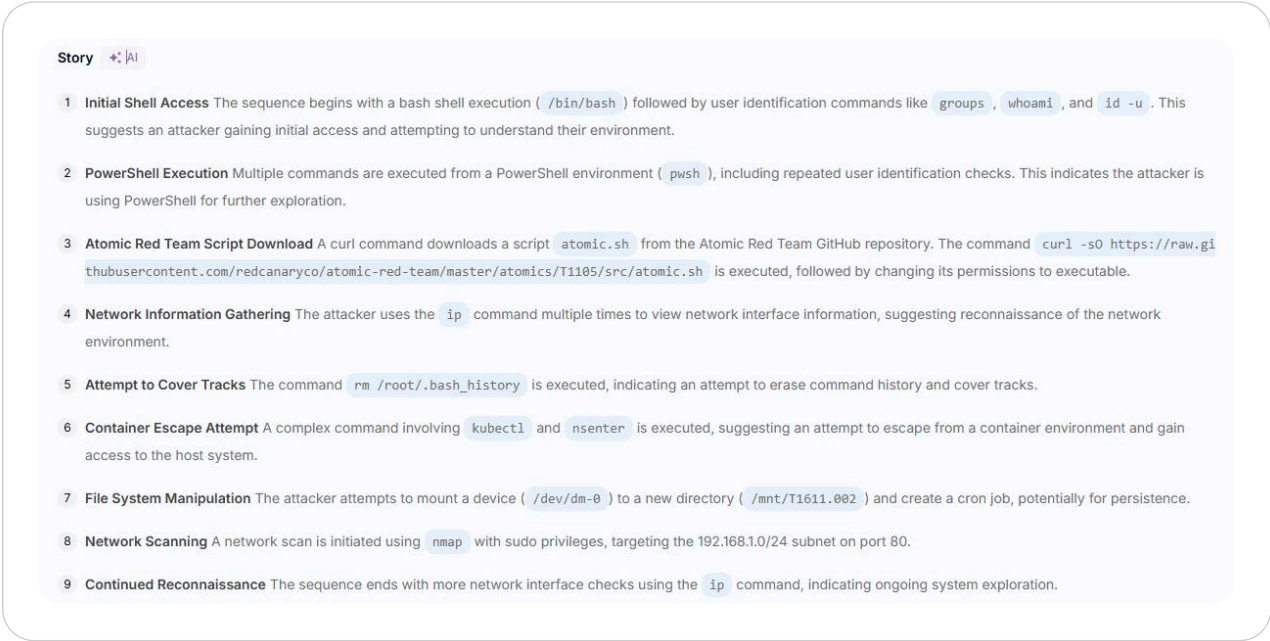
Runtime Detection: **Where Sweet Shines**



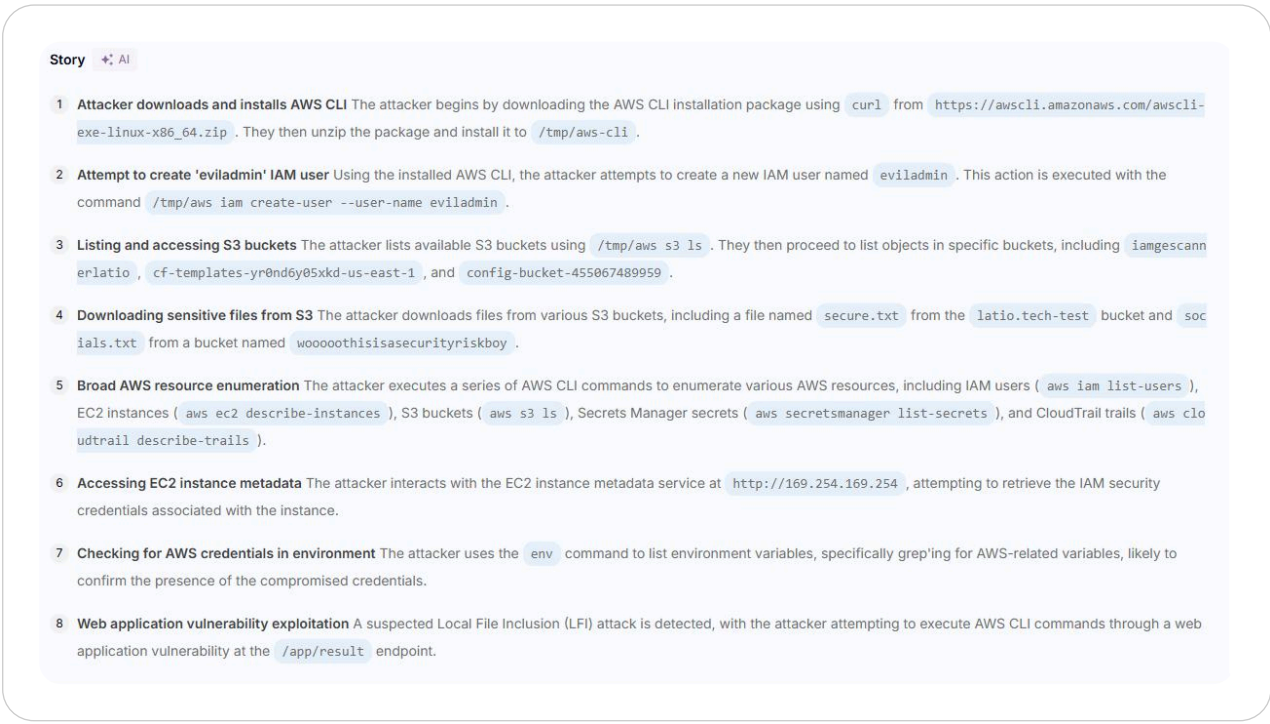
Combining API attacks with container attacks

Let's start where Sweet was the strongest. In testing, Sweet stood out for its approach to **incident correlation.** Where other detection tools tend to drown you in alerts, often firing off dozens of findings for a single attack sequence, Sweet does a much better job summarizing what happened into a single understandable attack chain. It ties together bash execution, container drift, file tampering, and network behavior into a coherent incident view.

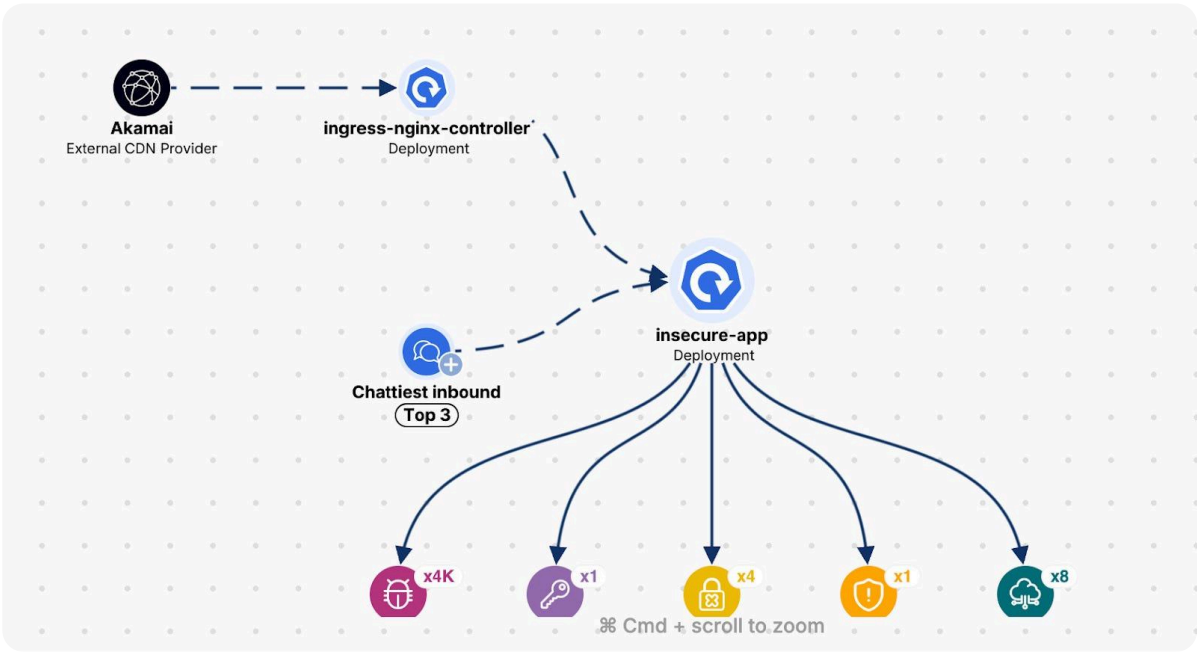
If the main failure of operationalizing cloud security operations is the SOC not understanding cloud alerts, Sweet does a great job breaking down the attack.



For example, when running a series of Atomic Red Team tests, including container escapes and file manipulations, **Sweet correctly identified each technique and grouped them into a single, understandable attack chain.** The tool provided clear details about what was run, which pods were involved, and how the activity unfolded. It also offered full process logs for investigation.



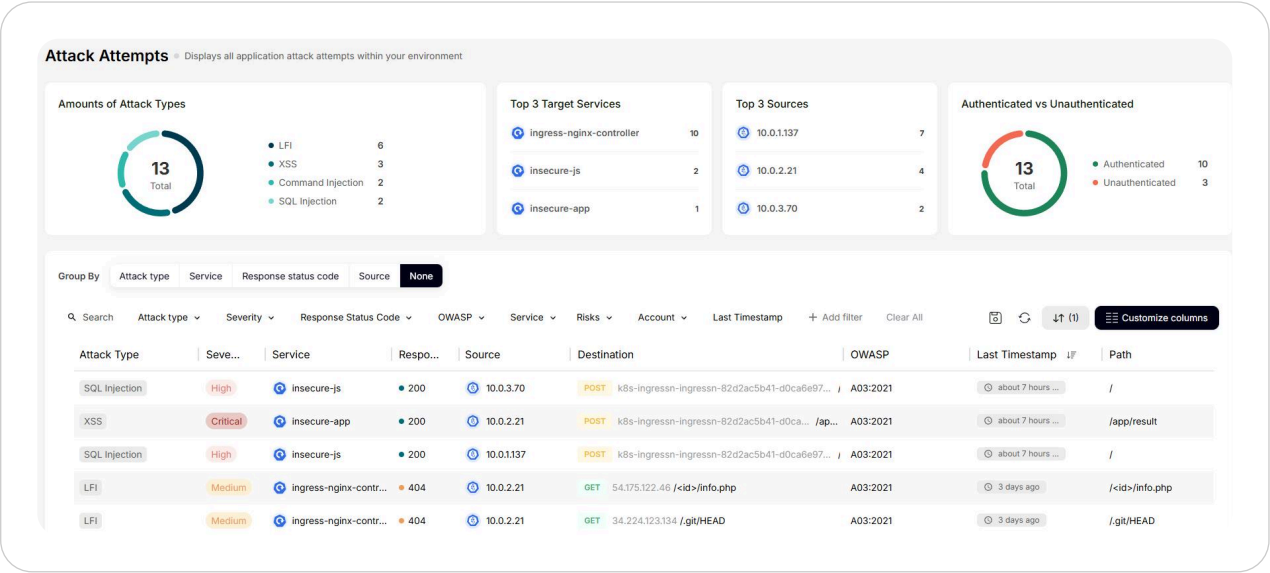
The most fun part of testing Sweet is how it will call me out if it notices the attack not working. The drift and manual detections correlate behind the scenes to deftly craft an accurate story of what happened, even when that involves making mistakes. In the SOC, **this context is critical and the difference between pinging the DevOps team one time with a complete story of what happened and fifteen times trying to track down a false positive.** I also managed to get my test environment infected by a real attacker, but it was immediately clear what was happening in the environment.



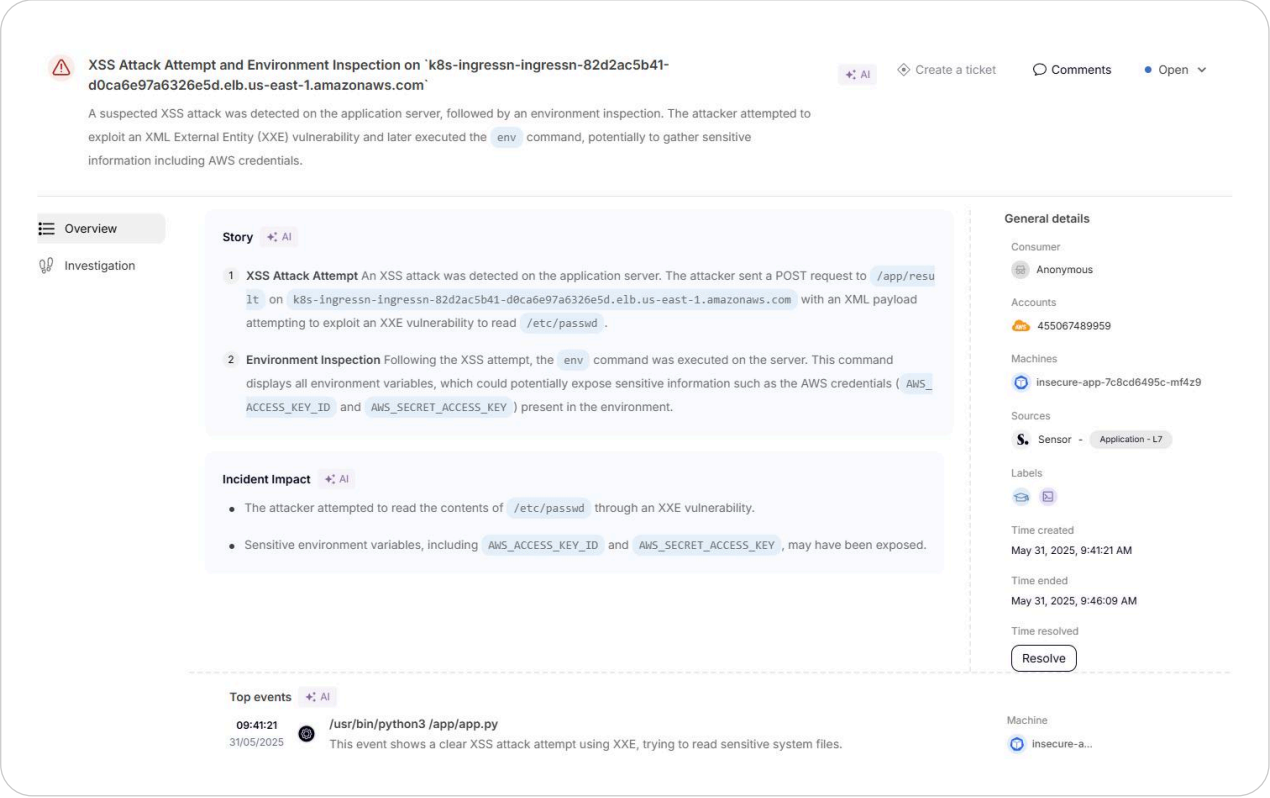
Understanding how network flows work in your application

It's also worth mentioning how the detection capabilities cross between API, cloud and workload layers. Sweet applies this detection methodology across all of the attack layers to identify when attacks pivot into the cloud from the workload.

The ability to create custom detection rules is essential to security operations teams as well. You can define behaviors, like outbound calls to suspicious IPs, that should trigger alerts, allowing for tuning and proactive guardrails. SOAR capabilities are limited, but this is far ahead of where most security teams are at. **Basic capabilities like killing processes work well** but teams looking to build fully automated response pipelines should look towards the Torq integration.



Despite being a newer feature, application layer attack detection worked really well and was well summarized. In the above screenshot, I saw both some of my own application attacks, as well as some web crawlers trying to exploit non-existent PHP services. **In keeping with the noise reduction, my actual attacks triggered stories, while the unsuccessful crawlers were logged without triggering incidents.**

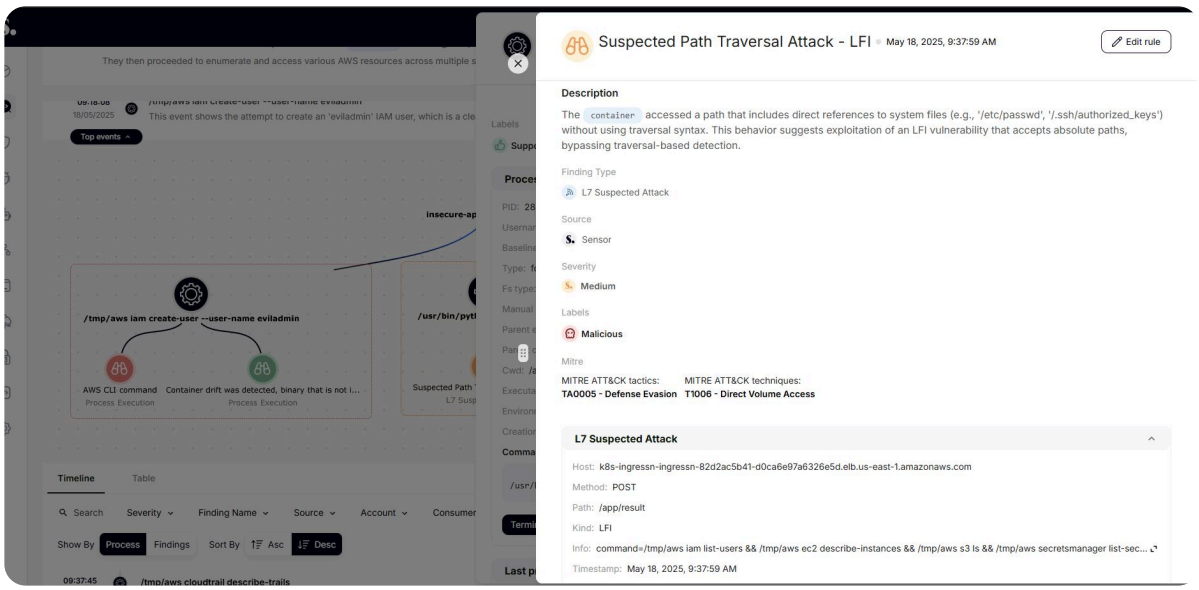


If you're new to cloud security, you may not instantly recognize how cool the above screenshot attack story is. In isolation, a DevOps team member might look at the environment variables in a running pod, triggering an alert that an analyst has to go and hunt down; however, **a DevOps team is never running an XML injection payload before doing that.** This is a critical example of where the application layer enables meaningful response.

The ability for SecOps teams to get actionable application insights is what CADR is all about, and it's awesome to see more from vendors.

It's also worth briefly mentioning that unlike many other runtime focused providers, Sweet also provides support for Windows OS, enabling them to be a one stop replacement for your runtime security.

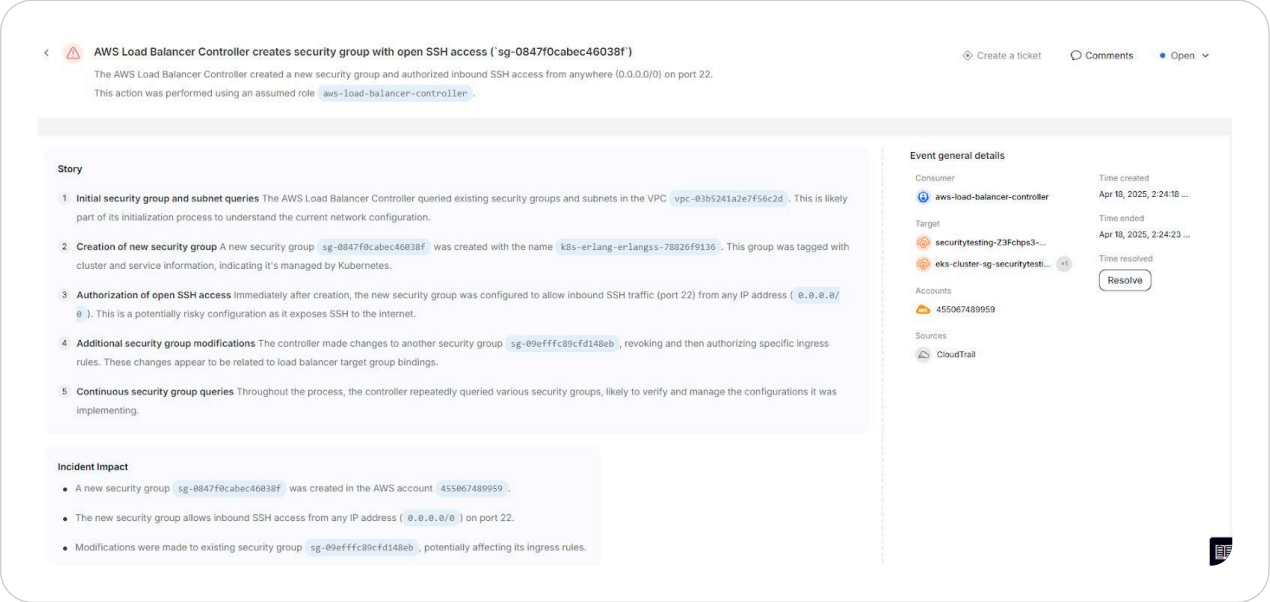
Where There's Still Room to Grow on Detections



The attack summaries could also have pros and cons when striking that delicate balance between too much noise and not enough. Every meaningful test I conducted got a story created; however, some individual events would trigger a finding. To be clear, the sensor always detected the event, so custom alerting would still have caught the attack, but it wasn't always rolled up into an entire incident. To be honest, due to the noisiness of most cloud detection tools, I think Sweet struck a good overall balance when deciding what to alert about as critical. Every security tool has to strike a balance on deciding what to service, and Sweet did a good job.

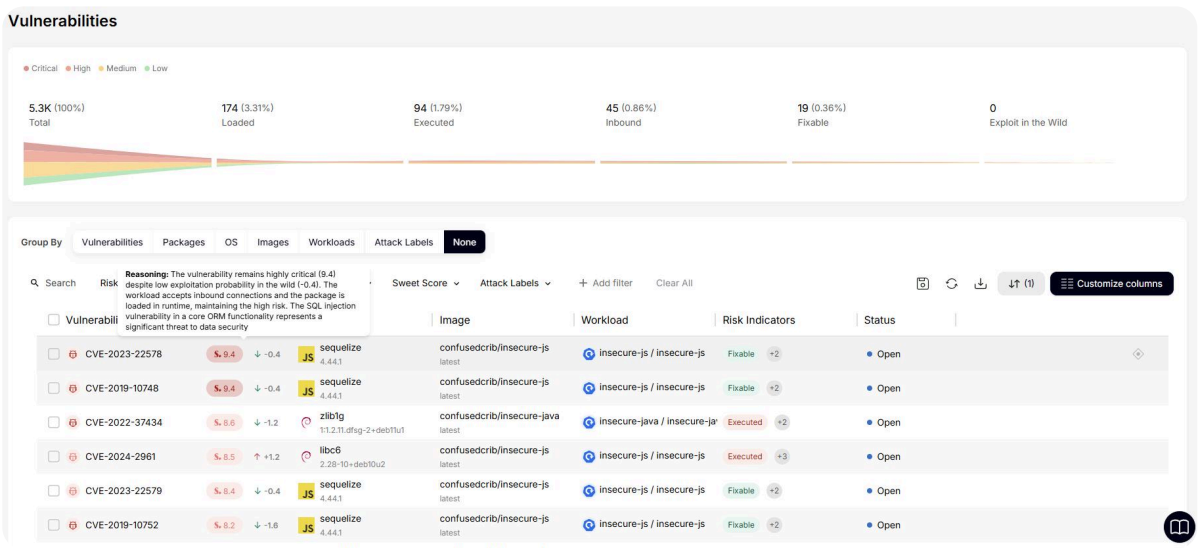
I think Sweet struck a good overall balance when deciding what to alert about as critical.

While it wasn't in the UI at time of testing, I'm extremely excited for Sweet's soon to be launched **potential misconfiguration detection.** One of the key challenges in cloud security in the SOC is differentiating interesting events from impactful ones; Sweet picks up a lot of activity that the security team might be interested in, but isn't necessarily related to an attack. Automatically delineating between these events is awesome and helps **give security teams real time visibility without clogging up their alerts.**



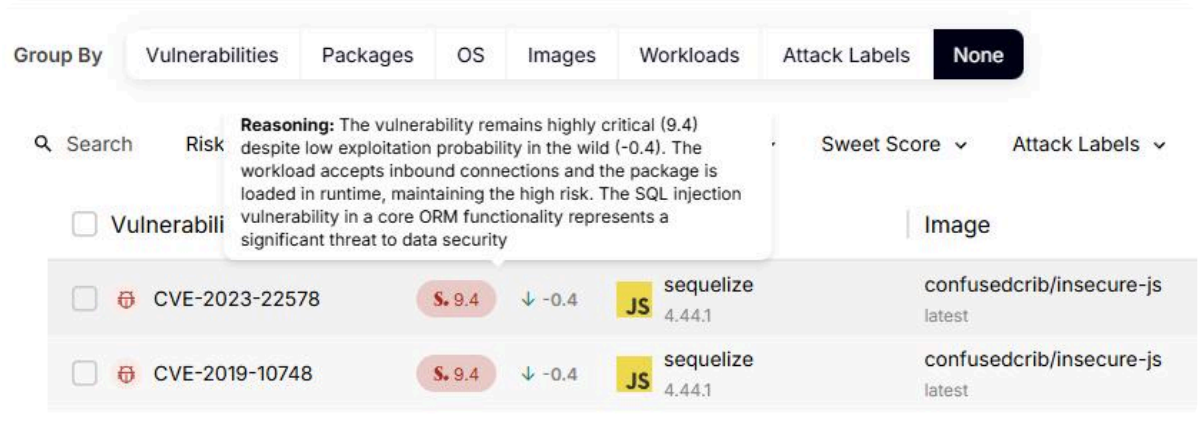
For example, in this test case I opened my cluster to the internet, and Sweet alerted me to it as a potentially suspicious change (in this case it was a classic oopsie in my terraform). Similarly, Sweet alerted me when I made some impactful IAM and S3 configuration changes. I'm excited for when these are surfaced as interesting misconfiguration findings rather than alerts.

Vulnerability Insights: **Better than Average**



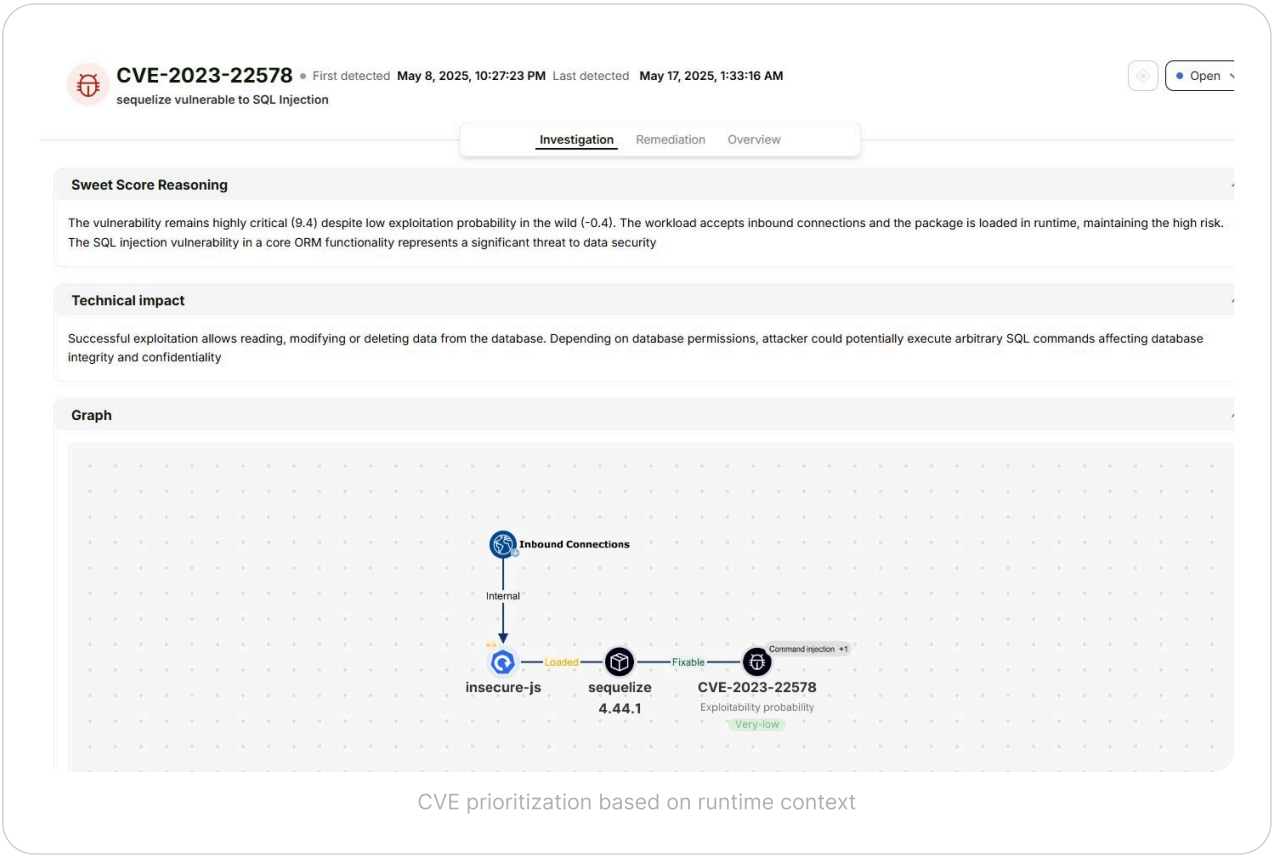
Vulnerability Prioritization

Sweet includes vulnerability discovery and management capabilities amplified by runtime data and LLM based prioritization. At a basic level, Sweet can prioritize with the standard loaded/executed and network reachability distinctions. For the execution detection, reliability seemed based on language, and I didn't see any function level reachability happening. This data carries into detection by optionally integrating your container registry.



Accurate prioritization reasoning even without function level reachability

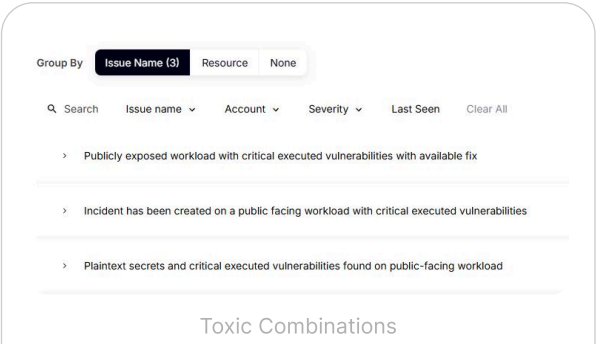
However, the Sweet scoring was surprisingly accurate, understanding the nature of the packages and their common usages in applications, and prioritizing based on the context of the app.



CVE prioritization based on runtime context

With remediation guidance, the effected image layer is given, but the LLM remediation guidance was usually not that helpful.

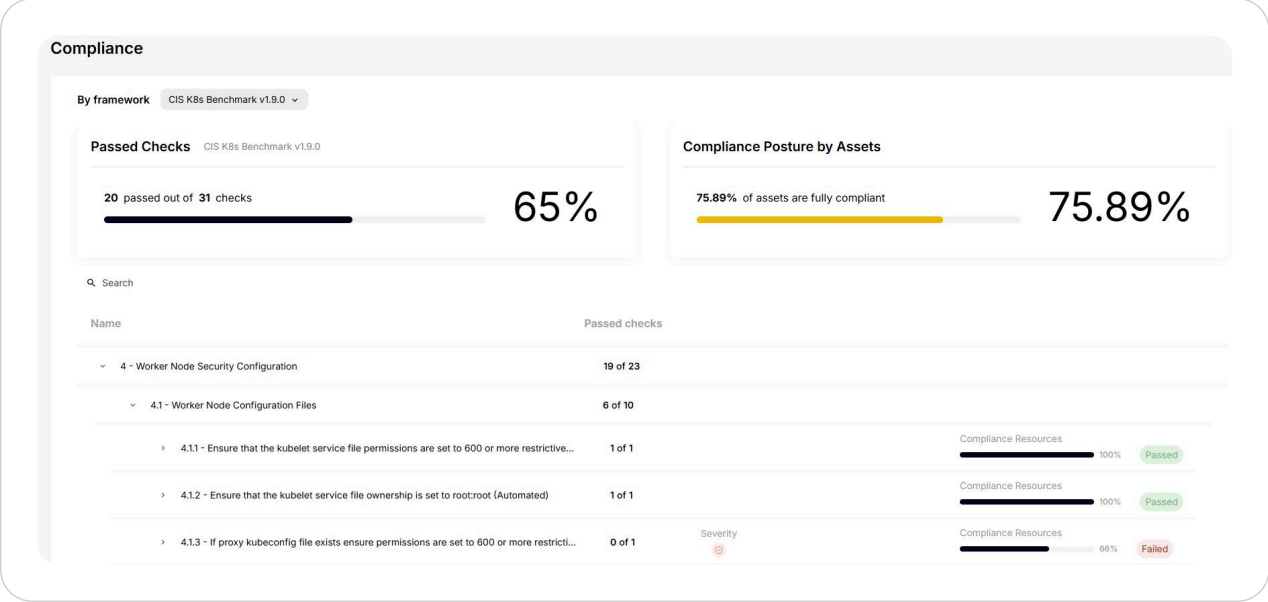
Some critical context for fixing container vulnerabilities, like base image versions, fix availability per distro, or whether a vulnerability is fixable in an upstream layer is missing. These are the kinds of details teams need in order to actually remediate issues at scale - and details most CNAPPs don't provide.



Toxic Combinations

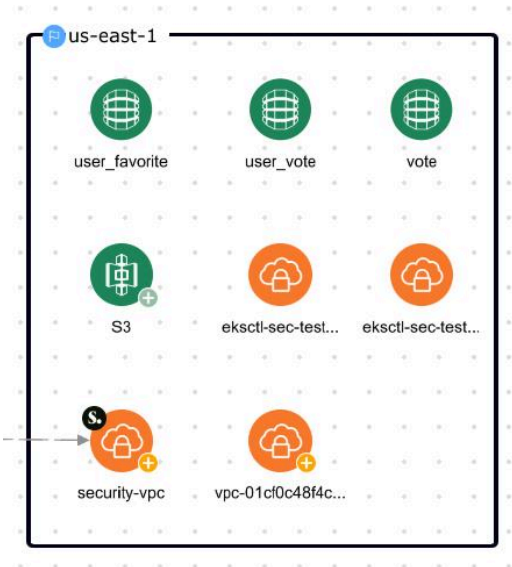
Prioritization is present, and Sweet shows the patterns to highlight "toxic combinations" of exposed services and critical vulnerabilities. The implementation here is functional but basic. It's also worth mentioning the biggest feature gap on the vulnerability side - the lack of agentless scanning.

Posture and Asset Management



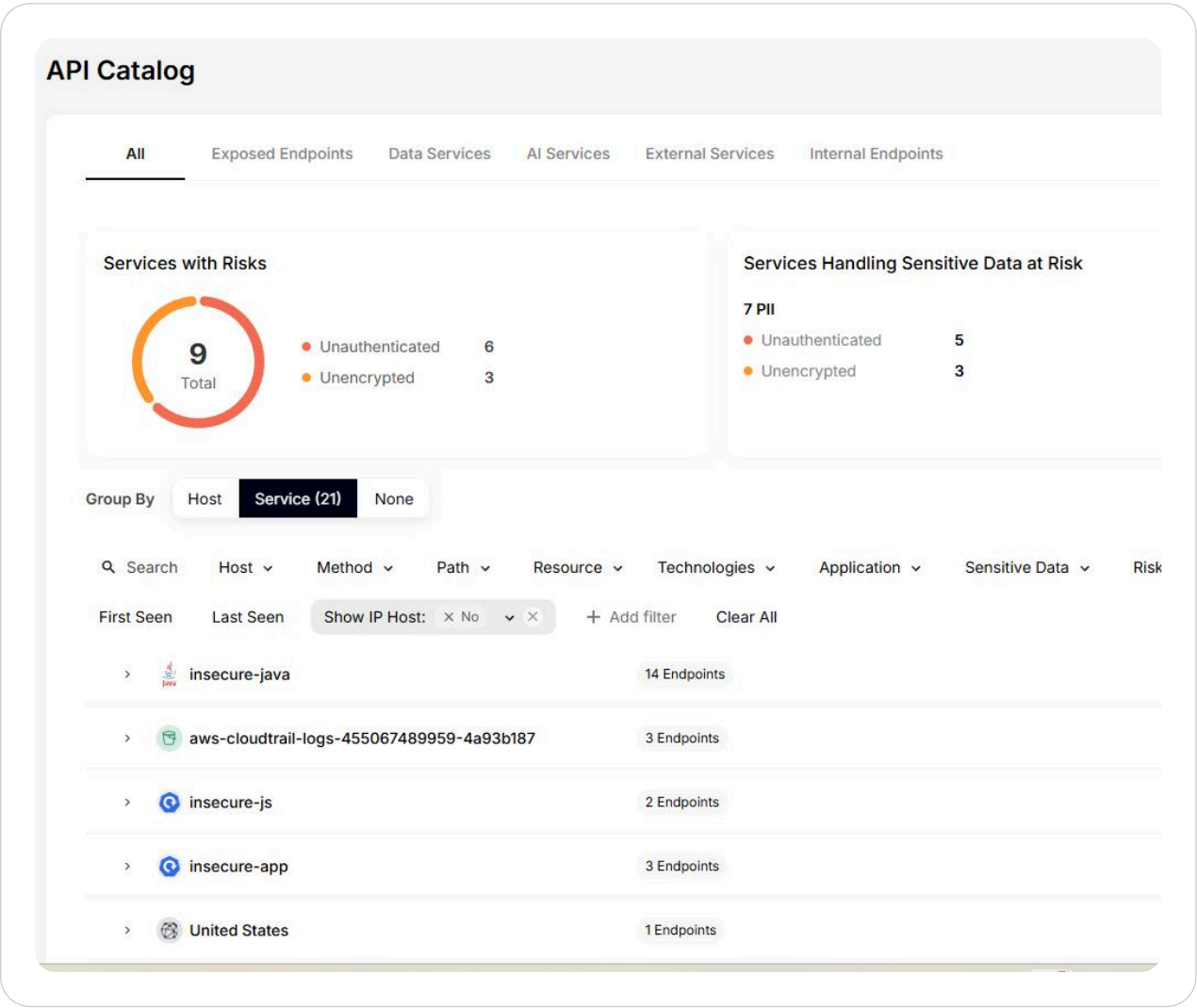
Sweet includes some basic CSPM functionality, including compliance checks and hardening recommendations, but the depth is limited. Teams used to richer policy management or better search and customization will find these features underwhelming. Asset visibility is also skewed toward containerized environments, with gaps in coverage for unmanaged assets unless a sensor is deployed.

The topology view provides a helpful visual of how services connect with the sensor showing service to service traffic. I appreciated how all meaningful AWS assets are mapped in the topology instead of only showing workloads - it was great to see my critical lambda functions grouped in with the clusters. Unfortunately, only basic data exists for workloads without the sensor installed.



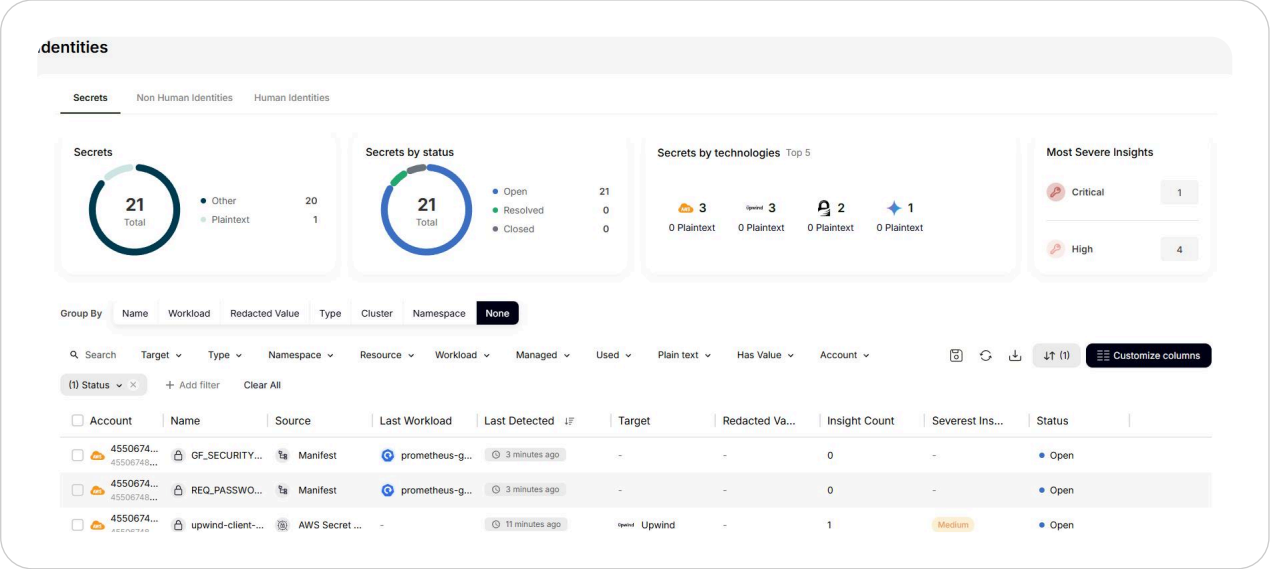
the biggest feature gap on the vulnerability side - the lack of agentless scanning.

API Catalog and Identity Data



It's been exciting to see API security grow into a feature of CADR, as gathering this application layer data has long been a missing point of context for understanding cloud workloads. **Understanding API catalogues and regular traffic is foundational to understanding how workloads are operating** and detecting application layer attacks. In testing, Sweet was able to helpfully group by either hostnames or service names to know exactly what API endpoints a service was surfacing.

Understanding API catalogues and regular traffic is foundational to understanding how workloads are operating and detecting application layer attacks.



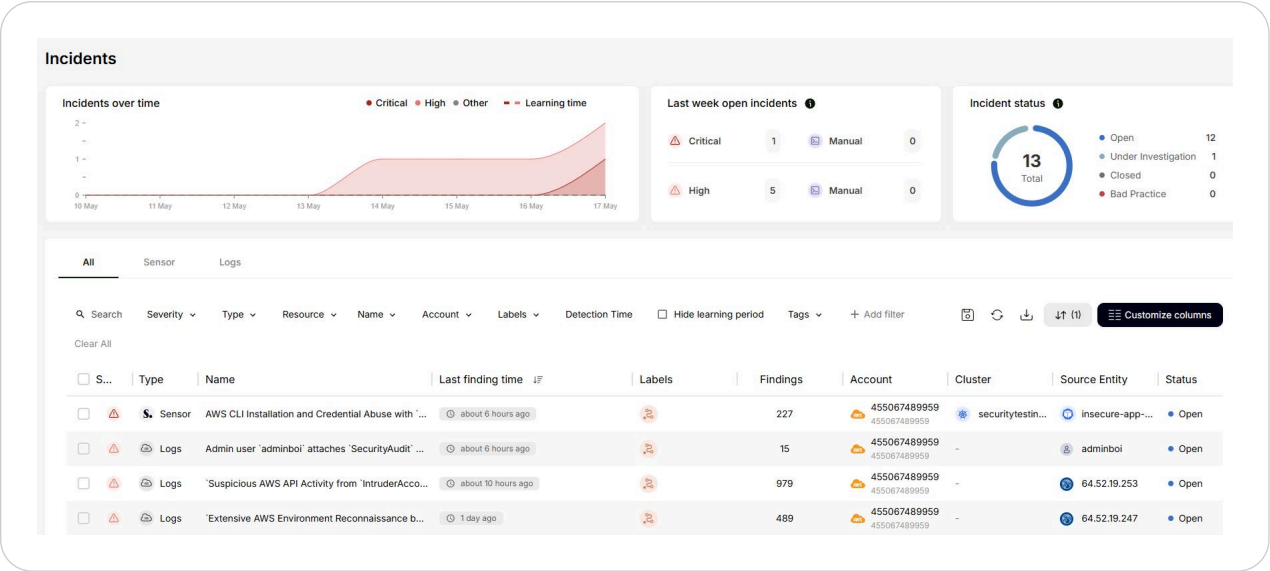
Identity was an unexpected but strong functionality in the platform. Sweet does a good job highlighting non-human identities and associated risks thanks to its runtime visibility. It's not building behavior-based IAM policies yet, but the foundational data is solid - especially the NHI detection.

Integration and Workflow

Sweet offers the standard integrations for event forwarding, ticketing, and notifications. Nothing particularly novel here, but it covers the basics well enough for most SOC workflows. The workflow capabilities aren't incredibly mature, but probably assume that is happening more in other tools.

Its ability to correlate attacks and reduce noise is a clear strength.

Final Take



Sweet Security is a strong runtime detection platform, particularly for teams looking for better incident context and alert fidelity. Its ability to correlate attacks and reduce noise is a clear strength. Especially for teams looking for a simple workload protection tool to enable their SOC and Cloud Security teams to better respond to runtime alerts, they have a strong offering.

As I've written elsewhere, **I'm not a believer in the all-in-one CNAPP Megazord** that gobbles up the security budget; however, if you're looking for that full feature list CNAPP replacement, i.e. something that has a check the box offering on everything from your asset and vulnerability management, compliance posture, IaC scanning, and code security, Sweet does not have all of these capabilities. The runtime piece is more mature than most of the larger competition, but the other areas are less developed.

Rather than being a drawback, I think these solutions are especially well paired with robust application security scanning solutions that surface vulnerability findings to developers. **As part of a layered approach that includes a solid ASPM or shift left application security tools, Sweet is the runtime complement to secure the applications once they're deployed.** I believe teams with cloud native architectures should adopt a strong ASPM platform alongside a runtime CNAPP (or CADR) in order to get true code to cloud coverage and application protection. For larger enterprises with highly distributed environments I typically recommend separate CSPM, CADR, and ASPM platforms to allow each to really get operationalized to its fullest extent.

Overall, Sweet is a great tool for teams looking to get runtime protection of their cloud environments. It's not the best tool for teams looking primarily for basic cloud asset visibility and vulnerability management, or for establishing code to cloud pictures for vulnerability remediation; however, I don't want to sell the CSPM features short either, they're pretty good. My practitioner preference has always been keeping these tools distinct where possible: CSPM, ASPM, and CADR. I would certainly include Sweet in any evaluation for runtime oriented cloud security capabilities.

Let's back up and answer the question, "what is a runtime CNAPP?" The answer is a tool that helps you deeply understand your cloud workloads, and that you can trust to find attackers in your environment. And that's Sweet.

The answer is a tool that helps you deeply understand your cloud workloads, and that you can trust to find attackers in your environment.