

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 <u>Sweet</u> <u>Security</u> 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

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

augments to their existing CSPM solutions, or as standalone solutions, depending on the specifics of their environment and coverage capabilities.

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

What is Sweet Security?

<u>Sweet Security</u> 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 <u>detection methodology</u> is genuinely unique among anomaly based detections, with some pros and cons.

Where Sweet Fits in the CNAPP Landscape



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 <u>Aqua</u> (runtime) and Orca (posture) having a partnership, despite offering the same features on paper.

<u>Sweet</u> 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.
- 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.



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.

Story +: Al

- 1 Attacker downloads and installs AWS CLI The attacker begins by downloading the AWS CLI installation package using curl from https://awscli.amazonaws.com/awscliexe-linux-x86_64.zip . They then unzip the package and install it to /tmp/aws-cli .
- 2 Attempt to create 'eviladmin' IAM user Using the installed AWS CLI, the attacker attempts to create a new IAM user named eviladmin . This action is executed with the command /tmp/aws iam create-user --user-name eviladmin .
- 3 Listing and accessing \$3 buckets The attacker lists available \$3 buckets using /tmp/aws s3 ls . They then proceed to list objects in specific buckets, including iamgescann erlatio, cf-templates-yr0nd6y05xkd-us-east-1, and config-bucket-455067489959.
- 4 Downloading sensitive files from S3 The attacker downloads files from various S3 buckets, including a file named secure.txt from the latio.tech-test bucket and soc ials.txt from a bucket named wooooothisisasecurityriskboy.
- 5 Broad AWS resource enumeration The attacker executes a series of AWS CLI commands to enumerate various AWS resources, including IAM users (aws iam list-users), EC2 instances (aws ec2 describe-instances), S3 buckets (aws s3 ls), Secrets Manager secrets (aws secretsmanager list-secrets), and CloudTrail trails (aws clo udtrail describe-trails).
- 6 Accessing EC2 instance metadata The attacker interacts with the EC2 instance metadata service at http://169.254.169.254, attempting to retrieve the IAM security credentials associated with the instance.
- 7 Checking for AWS credentials in environment The attacker uses the env command to list environment variables, specifically grep'ing for AWS-related variables, likely to confirm the presence of the compromised credentials.
- 8 Web application vulnerability exploitation A suspected Local File Inclusion (LFI) attack is detected, with the attacker attempting to execute AWS CLI commands through a web application vulnerability at the /app/result endpoint.

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.



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.

Amounts of Attack Types			Top 3 Tar	rget Services	Тор	3 Sources		Authenticated vs Unauth	henticated
	• LFI 6		O ingres	ss-nginx-controller	10 🔘	10.0.1.137	7		
13 Total	XSS 3 Command Injection 2		📀 insect	ure-js	2	10.0.2.21	4	13 Total	Authenticated 10 Unauthenticated 3
	 SQL Injection 		(insect	ure-app	1 ()	10.0.3.70	2	- Chur	
Sroup By Attack type Service	Response status code Sourc Severity ~ Response Status C	e None	ASP v Service v	Risks v Accoun	t v Last Time:	stamp + Add 1	ilter Clear All	ලි C 🕅 🕅	E Customize columns
Group By Attack type Service Q Search Attack type ~ Attack Type Service	Response status code Sourc Severity - Response Status C e Service	e None ode ~ OW, Respo	ASP - Service - Source	Risks - Accoun Destination	t ✓ Last Time:	istamp + Add 1	liter Clear All OWASP	তি C ایٹر (۱) Last Timestamp الآ	≣≣ Customize columns Path
iroup By Attack type Service Q. Search: Attack type ~ Attack Type Ser SQL injection H	Response status code Source Severity Response Status C e Service p) Q insecure-js	e None ode ~ OW. Respo • 200	ASP • Service • Source () 10.0.3.70	Risks ~ Accound Destination	t - Last Time:	stamp + Add 1	ilter Clear All OWASP A03:2021	ت ن بار (۱) Last Timestamp الا و about 7 hours	E Customize columns Path
Attack type Service Q. Search Attack type Attack Type Service SQL Injection IH XSS C	Response status code Source Severity > Response Status C e Service m Insecure-js tcol Insecure-app	e None ode ~ OW. Respo • 200 • 200	ASP × Service × Source ③ 10.0.3.70 ④ 10.0.2.21	Risks v Accoun Destination POST kBs-ingress RoST kBs-ingress	t Last Time: h-ingressn-82d2act	stamp + Add 1	liter Clear All OWASP A03:2021 A03:2021	C Inf (1) Last Timestamp IF about 7 hours about 7 hours	E Customize columns Path / /app/result
Attack type Service Q. Search Attack type Service Attack Type Service SQL injection Ha XSS CG SQL injection Ha	Response status code Source Severity Response Status C Response Status C	e None ode ~ OW. Respo • 200 • 200	ASP Service Source Source () 10.0.3.70 () 10.0.2.21 () 10.0.1.137	Risks × Accoun Destination POST kBs-ingress POST kBs-ingress	t - Last Time: h-ingressn-82d2act -ingressn-82d2act	stamp + Add 1 5541-d0ca6e97 <i>j</i> 5541-d0ca <i>jap</i> 5541-d0ca6e97 <i>j</i>	liter Clear All OWASP A03:2021 A03:2021	C Jt (1) Last Timestamp LF o about 7 hours o about 7 hours o about 7 hours o about 7 hours	Path / /app/result /

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. 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

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

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.**

AWS Load Balancer Controller creates security group with open SSH access ['sg-0847/0cabec46038f'] The AWS Load Balancer Controller created a new security group and authorized inbound SSH access from anywhere (0.0.0.0/0) on port 22. This action was performed using an assumed role ass-load-balancer-controller.		⊃ Comments ● Open ~
Story Initial security group and subnet queries The AWS Load Balancer Controller queried existing security groups and subnets in the VPC (vpc-03b5241a2e7F5sc2d). This is likely part of its initialization process to understand the current network configuration. Creation of new security group A new security group sag-0847f6cabec40038f was created with the name kts-erlang-erlangss-78826f9136. This group was tagged with cluster and service information, indicating it's managed by Kubernetes. Authorization of open SSH access immediately after creation, the new security group was configured to allow inbound SSH traffic (port 22) from any IP address (0.0.0.0/ 0.). This is a potentially risky configuration as it exposes SSH to the internet. Additional security group modifications The controller made changes to another security group set_09efffc09cfd148eb, revoking and then authorizing specific ingress rules. These changes appear to be related to load balancer target group bindings. Continuous security groups, likely to verify and manage the configurations it was inclementation.	Event general details Consumer	Time created Apr 18, 2025, 22418 Time ended Apr 19, 2025, 22423 Time resolved Resolve
Incident Impact A new security group sg-8847facabec46838f was created in the AWS account 4558c7489959 . The new security group allows inbound SSH access from any IP address (0.0.0.0/0) on port 22. Modifications were made to existing security group sg-89efffc80cfd148eb , potentially affecting its ingress rules.		

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

Critical High He	dium 😐 Low											
5.3K (100%) Total		174 (3.319 Loaded	:)		94 (1.79%) Executed	45 (0.86%) Inbound		19 (0.36%) Fixable		0 Exploit in the	e Wild	
Group By Vulneral	ollities Packages	OS In	lages	Workloads Attack	Labels None							
Q Search Risk	Reasoning: The vulnerab despite low exploitation p workload accepts inboun loaded in runtime, mainta vulnerability in a core ORI significant threat to data	ility remains robability in d connection ining the hig W functional security	highly cri the wild (as and the h risk. Th ty repres	tical (9.4) (-0.4). The Sw a package is e SQL injection vents a	eet Score 👻 Attack Labels 👻	+ Add filter Clear All Workload	Risk Indicato	rs	Status	(1) 1¢ لغ	≣≣ Custom	ize columns
	023-22578			sequelize	confusedcrib/insecure-is							
☐ ⊕ CVE-20		0.4 V	-0.4	JS 4.44.1	latest	Insecure-js / Insecure-js	Fixable +2		• Open			۲
	019-10748 S	.9.4	0.4	JS 4.44.1 JS 4.44.1	latest confusedcrib/insecure-js latest	 insecure-js / insecure-js insecure-js / insecure-js 	Fixable +2 Fixable +2		Open Open			۲
	019-10748 S 022-37434 S	.9.4 4 .8.6 4	-0.4 -0.4 1.2	JS 4.44.1 JS sequelize 4.44.1 (° zib1g 17.2.11.dfsg-2+deb11u1	latest confusedcrib/insecure-js latest confusedcrib/insecure-java latest	 insecure-js / insecure-js insecure-js / insecure-js insecure-java / insecure-java 	Fixable +2 Fixable +2 Executed +	2)	Open Open Open			
	019-10748 S 022-37434 S 024-2961 S	.9.4) ↓ .9.4) ↓ .8.6) ↓ .8.5) ↑	-0.4 -0.4 1.2 1.2	JS 4.44.1 JS sequelize 4.44.1 (2 zlib1g 1:1.2.11.dfsg-2+deb11u1 (2 libc6 2.28-10+deb10u2	latest confusedcrib/insecure-js latest confusedcrib/insecure-java latest confusedcrib/insecure-js latest	insecure-js / insecure-js insecure-js / insecure-js insecure-ja / insecure-ja insecure-ja / insecure-ja	Fixable +2 Fixable +2 Executed + Executed +	2	Open Open Open Open Open			*
	019-10748 S 022-37434 S 024-2961 S 023-22579 S	.9.4 ↓ .8.6 ↓ .8.5 ↑ .8.4 ↓	-0.4 -0.4 -1.2 -1.2 -0.4	JS 4.44.1 JS sequelize 4.44.1 (°) zlib1g 1.1.2.11.dfsg-2+deb11u1 (°) 2.28-10+deb10u2 JS sequelize 4.44.1	tatest confusedcrib/insecure-js latest confusedcrib/insecure-js latest confusedcrib/insecure-js latest confusedcrib/insecure-js latest	insecure-js / insecure-js insecure-js / insecure-js insecure-java / insecure-js insecure-js / insecure-js insecure-js / insecure-js	Fixable +2 Fixable +2 Executed + Fixable +2	2	 Open Open Open Open Open 			*
 	119-10748 S 122-37434 S 124-2961 S 123-22579 S 119-10752 S	.8.6 4 · · · · · · · · · · · · · · · · · ·	-0.4 -0.4 1.2 -1.2 0.4 1.6	JS 4.441 JS 8-8401 4.441	Untext confusedcrib/insecure-js Untext confusedcrib/insecure-java Untext confusedcrib/insecure-js Untext confusedcrib/insecure-js Untext confusedcrib/insecure-js Untext	insecure-ja / insecure-ja insecure-ja / insecure-ja insecure-java / insecure-ja insecure-java / insecure-ja insecure-ja / insecure-ja insecure-ja / insecure-ja insecure-ja / insecure-ja	Fixable +2 Fixable +2 Executed + Fixable +2 Fixable +2 Fixable +2	2	Open Open Open Open Open Open Open Open			۰ ۵

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.

Group By	Vulnerabilities	Packages	OS	Images	Workloads	Attack Labels None
Q Search	Risk Reasor Risk despite worklo: loaded vulnerabili signific	ing: The vulnera low exploitation ad accepts inbou in runtime, maint bility in a core Ol ant threat to data	bility ren probabili nd conne raining th RM funct security	nains highly cr ity in the wild ections and th e high risk. Th tionality repres	ritical (9.4) (-0.4). The e package is ne SQL injection sents a	Sweet Score 🖌 Attack Labels 🗸
	CVE-2023-225	78	S. 9.4	↓ -0.4	JS sequelize	confusedcrib/insecure-js latest
	CVE-2019-1074	8	S. 9.4	↓ -0.4	JS 4.44.1	confusedcrib/insecure-js latest

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.



																		1	nves	tiga	ation		Rem	ediati	on	Ov	ervie	ew																
Swe	eet s	Score	Rea	soni	ng																																							
The The	vulne SQL	erabili	y rem on vu	ains	highly	y cri in a	tical core	(9.4 0R	l) de M fu	spit	e low onali	ity n	epre	atio	n pr ts a	obal sigr	oility ifica	in th nt th	e wil reat	d (- to d	0.4) lata s	The	e woi rity	rkloa	d acc	cepts	inbo	ounc	l con	nect	ions	and	the p	back	age	is loa	aded	l in ru	untin	ne, n	nainti	ainir	ng the	high
Tec	hnie	al in	pact																																									
Suco	essf	ul exp	loitati	on a	lows	read	ling,	mod	difyi	ng o	r del	etine	g da	ta fr	om	the	datat	ase	Dep	penc	ding	on d	atab	ase p	ermi	issio	ns, a	ttac	ker c	ould	pote	ntial	ly ex	ecut	te ar	bitra	ry SI	QLC	omm	and	s affe	ectin	ng dat	abas
integ	rity	and co	onfide	ntial	ty																																							
0																																												
Gra	pn																																											
																. (69	Inbo	und (Conn	nectio	ins																						
																•	6	Inbo	und (Conn	nectio	ins																						
																• • ((S) ternal	Inbo	und (Conn	nectic	ins																						
																- - (- Ir	ternal	Inbo	und (Conn	nectio	ins																						
																- - - - -	ternal	Inbo	und (Conn	nectio	ins i						Comm	and inje	tion +														
																- - - - -	ternal	Inbo	und (Conn	nectio 	ons D		– Fixa	ble —	•	e G	Comm	and inje	tion +														
															i	nse	ternal	Inbo	und (Loade	Conn	nectio — (ueliz	ze	– Fixa	ble — CV	E-2	6 023	Comm -22	and inje	tion +														
															1	i lr	ternal	Inbo 	und (Conn sd —	nectio — (sequ	uns Juelia 14.1	ze	- Fixa	ble — CV Exp	E-2	023 0111y p	Comm -22 proba	and inje 578 bility	tion +														

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.

Grou	ip By	Is	sue Nar	ne (3)	Re	source	None				
۹	Sea	arch	Issue	name	~	Accou	nt 🗸	Severity	*	Last Seen	Clear All
	>	Public	ly expos	sed wor	kload	l with crit	tical exec	uted vuln	erabilit	ies with availa	able fix
	>	Incide	nt has b	een cre	ated	on a pub	lic facing	workload	with c	critical execut	ed vulnerabilities
	>	Plaint	ext secr	ets and	critic	al execut	ed vulne	rabilities f	ound a	on public-faci	ng workload
					Т	oxic	Com	nbina	tior	าร	

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.



npliance				
By framework CIS K8s Benchmark v1.9.0 v				
Passed Checks CIS K8s Benchmark v1.9.0		Compliance Posture by Assets		
20 passed out of 31 checks 65%	6	75.89% of assets are fully compliant	75	.89%
Q. Search				
Name	Passed checks			
 4 - Worker Node Security Configuration 	19 of 23			
 4.1 - Worker Node Configuration Files 	6 of 10			
> 4.1.1 - Ensure that the kubelet service file permissions are set to 600 or more restrictive	1 of 1		Compliance Resources	100% Passed
> 41.2 - Ensure that the kubelet service file ownership is set to root:root (Automated)	1 of 1		Compliance Resources	100% Passed
 4.1.3 - If proxy kubeconfig file exists ensure permissions are set to 600 or more restricti 	0 of 1	Severity	Compliance Resources	66% Failed

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.





	Exposed Endpoints Data S	ervices Al Services Exte	ernal Services Internal Endpoi	nts
Services wi	th Risks		Services Handling Se	ensitive Data at Risk
-			7 PII	
9	Unauthenticated	6	 Unauthenticated 	5
Group By H	act Sarrica (21) Nana			
	Service (21) None			
Q Search First Seen	Host ~ Method ~ Last Seen Show IP Host:	Path × Resource × Te × No × × + Add filte	echnologies - Application - r Clear All	Sensitive Data 🗸
Q Search First Seen ≻ ≝irre in	Host v Method v Last Seen Show IP Host: secure-java	Path - Resource - Te × No - × + Add filte 14 E	echnologies v Application v r Clear All	Sensitive Data 🗸
Q Search First Seen > define in > to a	Host ~ Method ~ Last Seen Show IP Host: secure-java	Path × Resource × Te × No • × + Add filte 14 E 59-4a93b187 3 En	echnologies - Application - r Clear All indpoints	Sensitive Data 🗸
Q Search First Seen > ₫ in > ເ a > ⓒ in	Host × Method × Last Seen Show IP Host: secure-java ws-cloudtrail-logs-4550674899 secure-js	Path → Resource → Te × No → × + Add filte 14 E 59-4a93b187 3 En 2 En	echnologies v Application v r Clear All indpoints indpoints	Sensitive Data ↓
Q Search First Seen →	Host v Method v Last Seen Show IP Host: secure-java ws-cloudtrail-logs-4550674899 secure-js secure-app	Path × Resource × Te × No × × + Add filte 14 E 59-4a93b187 3 En 2 En 3 En	echnologies ~ Application ~ r Clear All indpoints indpoints indpoints	Sensitive 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.

ititie5										
Secrets Nor	n Human Identities Hu	ıman Identities								
Secrets			Secrets by status		Secrets b	by technologies To	op 5		Most Severe In	sights
21	Other Plaintext	20	21	OpenResolved	21 0 📥 3	3 Openind 3	A 2 🔶 1		🤌 Critical	1
Total			Total	Closed	0 0 Plainte	ext 0 Plaintext	0 Plaintext 0 Plaintex	t	🤌 High	4
rroup By Name Q. Search Tar (۱) Status マ ×	Workload Redacte rget v Type v + Add filter Clea	rd Value Type C Namespace ~ R	Cluster Namespace Resource ~ Workload	None Managed	Used 🗸 Plain text -	✓ Has Value ✓	Account ~	60.5	±, ↓↑ (1) ≣≣ C	ustomize columns
Group By Name Q. Search Tar (1) Status ~ × Account	Workload Redacte rget + Type + + Add filter Clea Name	d Value Type C Namespace - R r All Source	Cluster Namespace	None ✓ Managed ✓ Last Detected 4F	Used -> Plain text	 Has Value Redacted Va 	Account ~	ලි 🗘 ප් Severest Ins	ર્મ્ડ ∔↑ (1) ≣≣ α Status	ustomize columns
Stroup By Name Q. Search Tai (1) Status × Account Acsolor48, 4550674, 4500674, 4500676, 4500676, 4500676, 45000000000000000000000000000000000000	Workload Redacte rget ↓ Type ↓ + Add filter Clea Name ^^ 合 GF_SECURITY	Id Value Type C Namespace - R r All Source	Cluster Namespace Resource - Workload Last Workload © prometheus-g	None Managed • Last Detected 4F O 3 minutes ago	Used v Plain text of Target	 Has Value Redacted Va 	Account ~ Insight Count	ම් ට ප් Severest Ins	لي لئ (۱) على من	ustomize columns
Broup By Name Q. Search Tai (1) Status < ×	Workload Redactr rget V Type V + Add filter Clea Name A GF_SECURITY A REQ.PASSWO	d Value Type C Namespace ~ R All Source Ta Manifest Ta Manifest	Cluster Namespace Resource V Workload Last Workload O prometheus-g O prometheus-g	None Managed « Last Detected 1F 3 minutes ago 3 minutes ago	Used v Plain text = Target -	 Has Value Redacted Va - 	Account ~ Insight Count 0 0	ت C د Severest Ins	L L1 (I) EE C Status • Open	ustomize columns

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.



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.



Incidents over time	• Critic	al • High • Other - Learning time	Last week op	en incidents		Incident status	0	
2			▲ Critical	1 🗈 Ma	anual O	13	Open Under Inve	1: stigation
- 0 	12 May 13 May 14 May	15 May 16 May	17 May	5 🔁 Ma	anual O	Total	Closed Bad Practic	ce (
All Sensor	Logs	Account y Labels y Datast	tion Time 💦 Hide learning	nerind Tage v	L Artist filtrar			izo columos
All Sensor Q Search Severity Clear All	Logs v Type v Resource v Name v	Account - Labels - Detect	ction Time 🛛 Hide learning	period Tags v	+ Add filter	0 0 4 (↓↑ (1)	iize columns
All Sensor Q. Search Severity Clear All S Type	Logs • Type • Resource • Name • Name	Account v Labels v Detect	tion Time D Hide learning	period Tags v Findings	+ Add filter Account	ලි 🗘 🕹	J↑ (1) Eli Custom Source Entity	ize columns Status
All Sensor Q. Search Severity Clear All S Type Q. S. Sensor	Logs V Type V Resource V Name V Name AWS CLI Installation and Credential Abuse w	Account v Labels v Defect Last finding time 1F rith O about 6 hours ago	Labels	period Tags v Findings 227	+ Add filter Account 455067489959 455067489959	ت ن بالم	J↑ (1) Source Entity O insecure-app	ize columns Status • Open
All Sensor Q Search Severity Clear All S Type S. Sensor Clear All Clear All Cl	Logs V Type V Resource V Name V Name V AWS CLI Installation and Credential Abuse w Admin user 'adminboi' attaches 'SecurityAu	Account v Labels v Detect Last finding time 4F vith ' O about 6 hours ago dri O about 6 hours ago	tton Time Didde learning	period Tags v Findings 227 15	+ Add filter Account \$25067489959 \$55067489959 \$55067489959 \$455067489959 \$455067489959	ت د ب Cluster ی securitytestin	↓↑ (1) Source Entity insecure-app & adminboi	ize columns Status • Open • Open

<u>Sweet Security</u> 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 <u>strong ASPM platform</u> alongside a <u>runtime CNAPP (or CADR)</u> 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.

