Modern detection engineering with Google SecOps

Google Cloud Next 25

David French

Staff Adoption Engineer
Google Cloud

John Stoner

Senior Security Consultant
Google Cloud

Agenda

01 Detection Engineering

02 Curated Detections

03 Custom Rules

04 Detection-as-Code

01. Detection Engineering

Fundamental Concepts

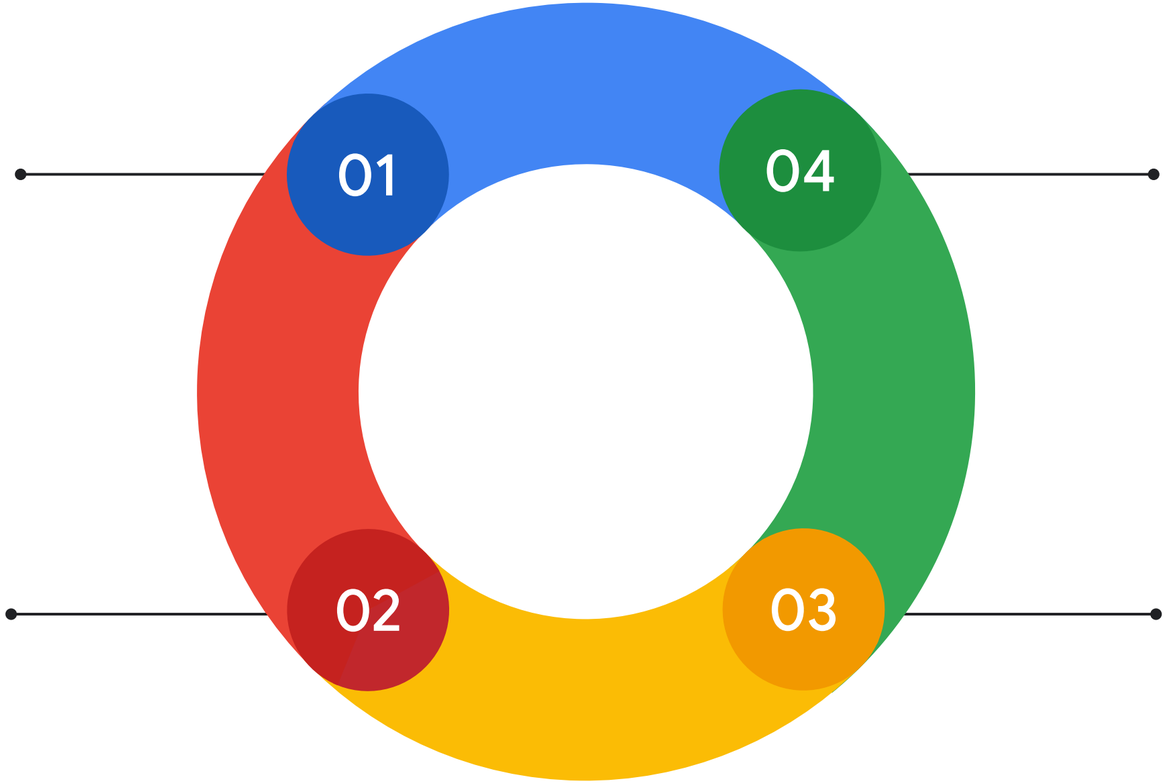
Detection Engineering Workflow

Identify & Prioritize

Understand critical assets. Identify most relevant threats to your organization based on industry trends, past incidents, and potential impact. Assess detection coverage. Identify data sources.

Develop & Test

Develop detection logic to identify specific behaviors, patterns, or anomalies. Test & validate.



Monitor & Iterate

Continuously monitor detection effectiveness. Improve coverage and reduce noise. Incorporate feedback from security analysts, incident responders, and threat intel analysts.

Deploy & Tune

Implement detections in security tools and continuously refine them based on performance and observed activity.

Threat Intelligence Driven Detection Engineering

Informs Prioritization

Identify prevalent threats targeting our industry or technology stack

Understand attacker tactics, techniques, and procedures (TTPs) to focus detection efforts

IOCs are good but TTPs are better!

IOCs provide initial detection opportunities, they are often short-lived.

Focusing on detecting attacker TTPs leads to more resilient detections

Common Language

Leverage the MITRE ATT&CK framework to identify gaps in detection coverage against known adversary TTPs and adjust accordingly

Prioritization of Data Collection

- Prioritize the onboarding of data sets based on threat intelligence
- Configure preprocessing for log events (filter, redact, enrich, etc)
- Log data that protects your most critical information should be prioritized - Crown Jewels
- Detection development starts as soon as the first data is onboarded

02. Curated Detections

Curated Detections

Applied Threat Intelligence

Host and Network based IOCs from Active Security Breaches & Investigations

Mandiant Frontline & Emerging Threats

Derived from Mandiant threat intelligence and findings in the field

Cloud Threats

Detections within GCP cloud infrastructure, Workspace and Chrome Enterprise Platform to protect your information

Ease of Implementation

Rule packages can be toggled to detection and/or alert and analyzed.

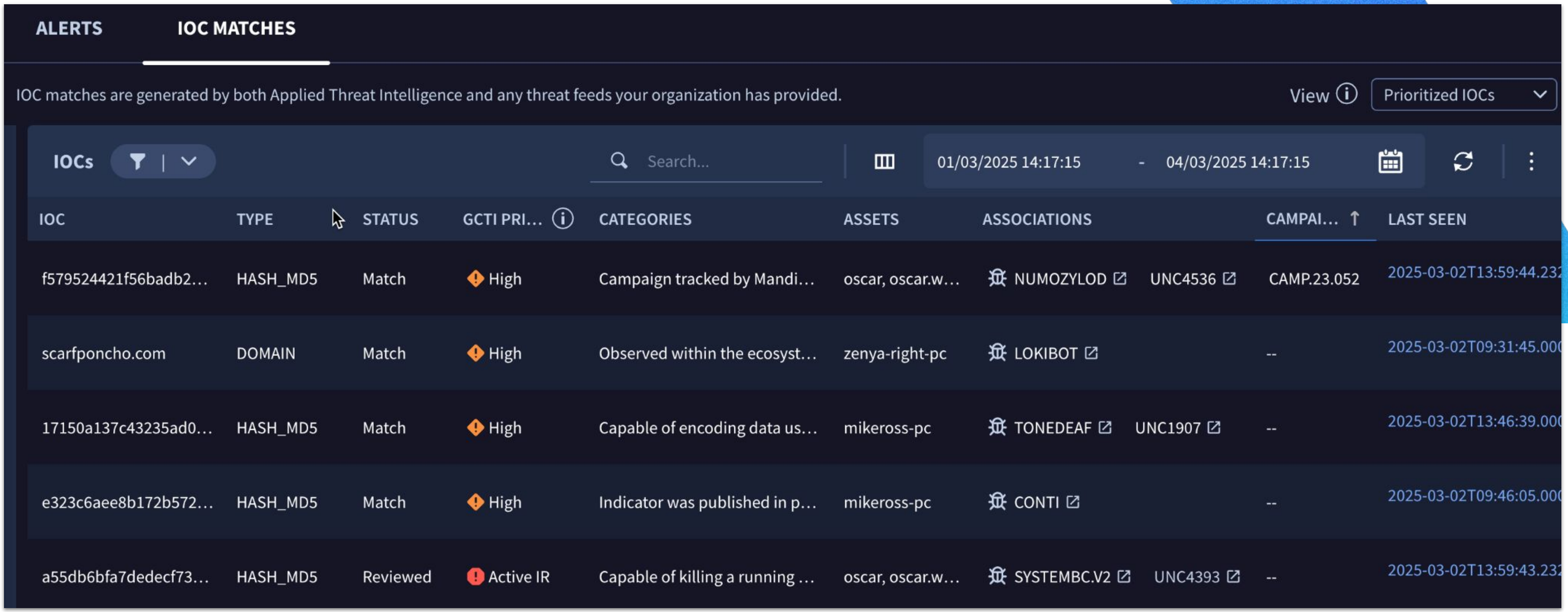
Risk Analytics

User and entity based analytics metrics to uncover outlier behavior

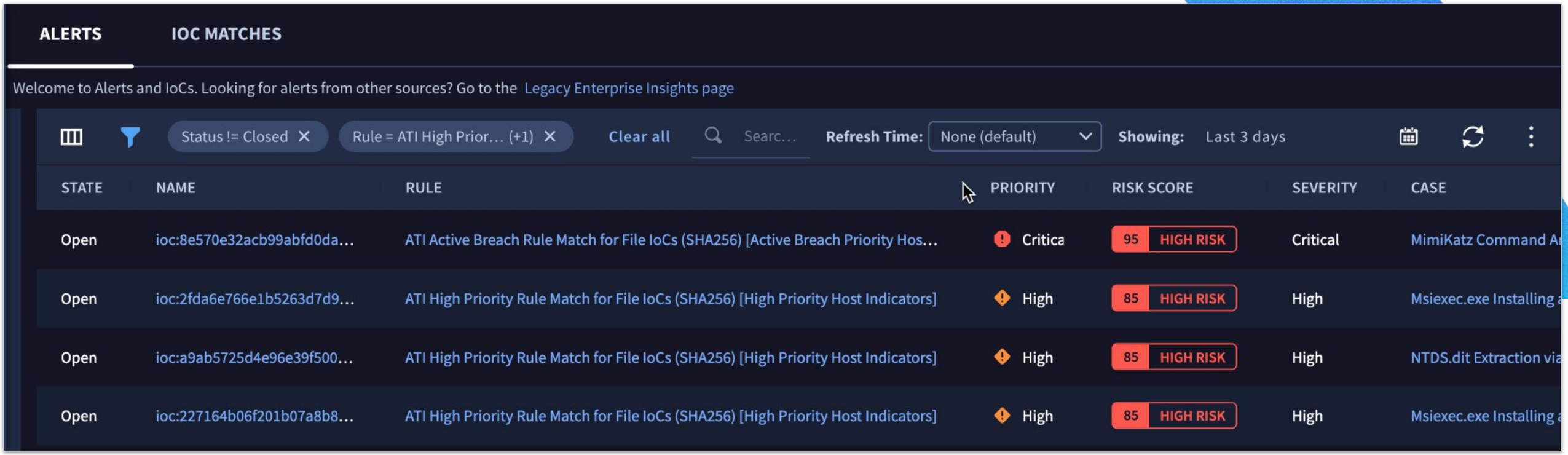
Customize with Exclusions

Provides the ability to tune rule packages

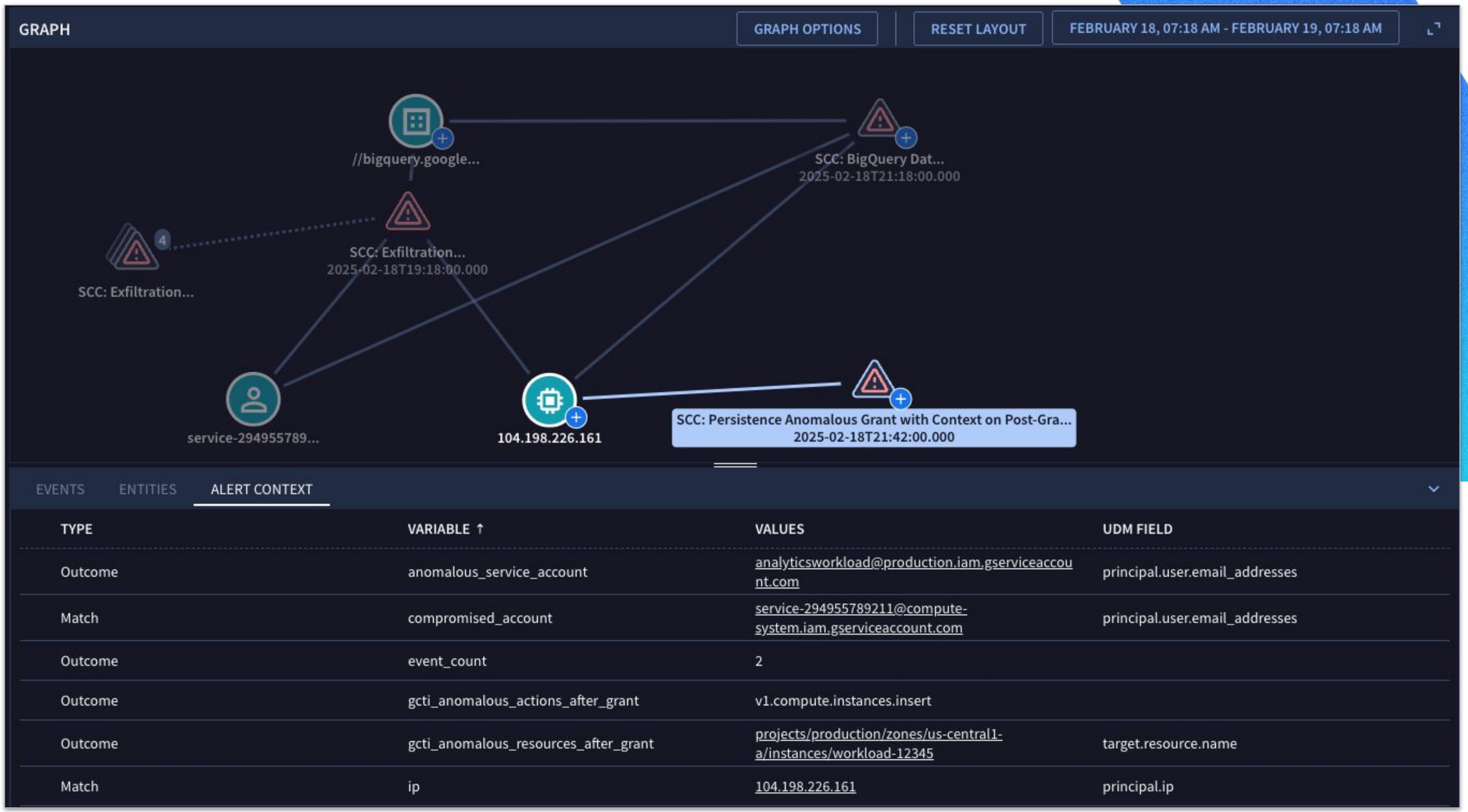
Applied Threat Intelligence



Applied Threat Intelligence

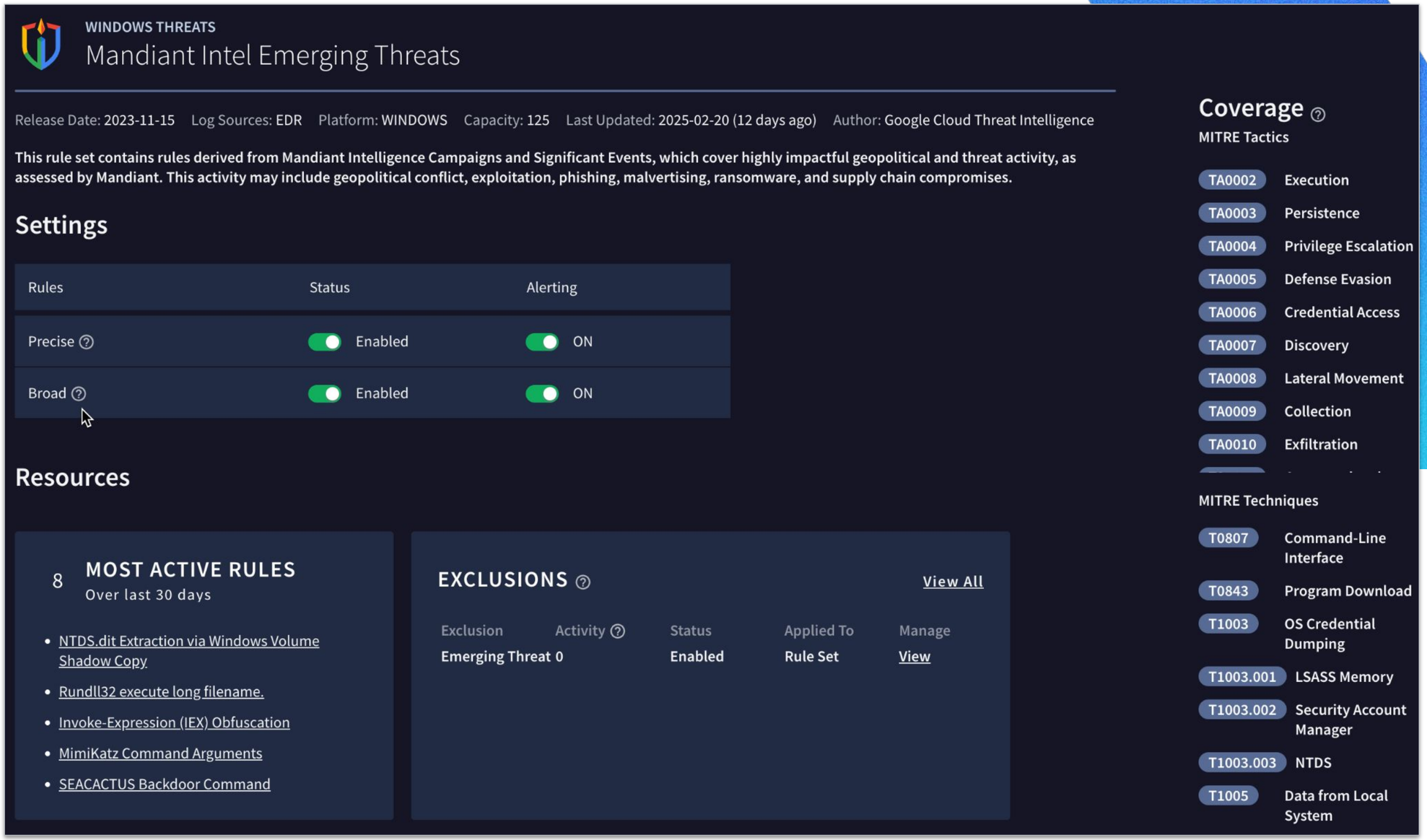


Cloud Threats



TYPE	VARIABLE ↑	VALUES	UDM FIELD
Outcome	anomalous_service_account	analyticsworkload@production.iam.gserviceaccount.com	principal.user.email_addresses
Match	compromised_account	service-294955789211@compute-system.iam.gserviceaccount.com	principal.user.email_addresses
Outcome	event_count	2	
Outcome	gcti_anomalous_actions_after_grant	v1.compute.instances.insert	
Outcome	gcti_anomalous_resources_after_grant	projects/production/zones/us-central1-a/instances/workload-12345	target.resource.name
Match	ip	104.198.226.161	principal.ip

Rule Pack Configuration



03. Custom Rules

Custom Rules

Build Your Own Rules

Rules editor with auto-complete, active syntax checker and integrated rule test capability

Entity Graph Integration

Leverage data within the entity graph to build detections that can use prevalence and GCTI data like TOR exit nodes and Remote Access Tools

Risk Analytics

User and entity based analytics metrics can be leveraged to build your own bespoke rules

Community Detection Content

Contains rules based on real world use cases as well as samples to assist you in building your own rules

Flexible Language

Over 60 functions available to assist you in working with your data

Retrohunt

Run a rule over existing events and generate detections for any matches

Developing Custom Rules in Google SecOps

Demo

Using Functions in Your Rules Rules

Functions provide more flexibility when working with data

Great for handling date/time values, strings, type conversion and much more!

```
$decoded_value =
re.replace(strings.base64_decode(re.capture($process.target.process.comman
d_line, `(?i)(?:-enc|-ec|-en)\s*(\S*)`)),`\0`, "")
(
    re.regex($decoded_value, `WebClient`) nocase or
    re.regex($decoded_value, `WebRequest`) nocase
)
```

Community-Driven Detection Content

1. **Provides a starting point**

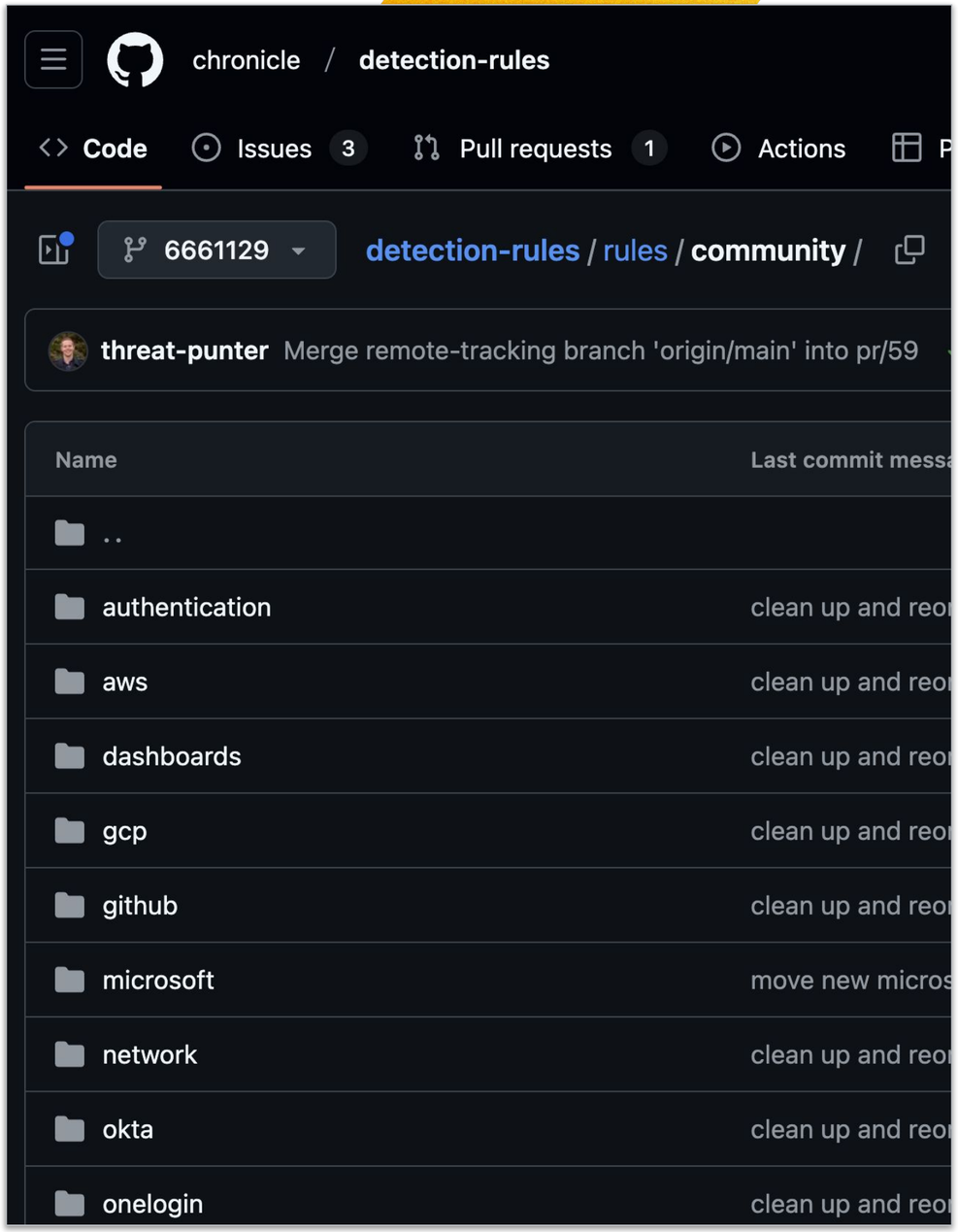
   Use these rules for threat hunting and for inspiration when building your own custom detections. Test, tune and customize them.

2. **Demonstrates YARA-L usage**

   Many of these rules use the latest capabilities of YARA-L and are used for educational purposes.

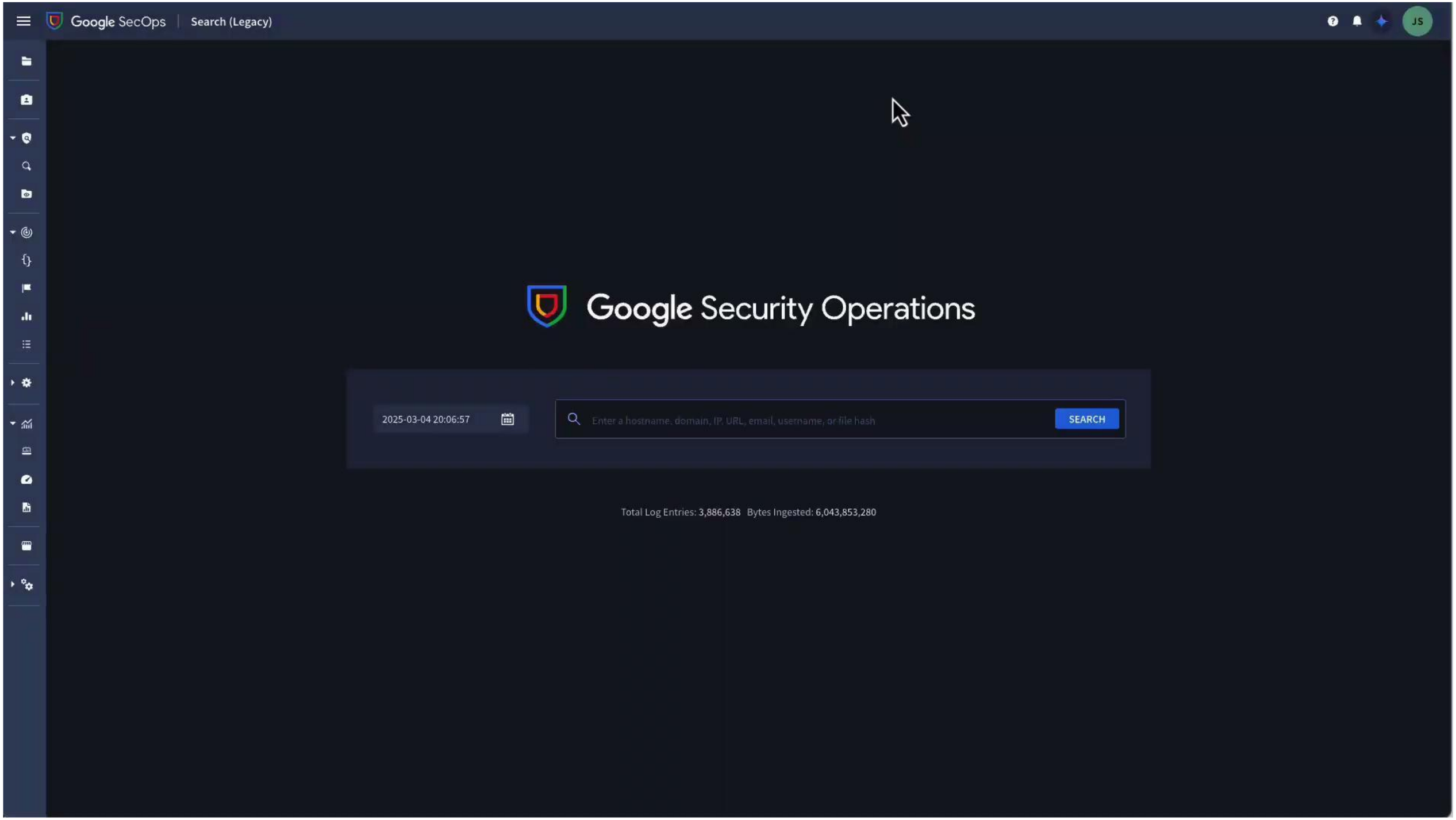
3. **We welcome contributions!**

   Contribution guidelines, a rule style guide, and example pull requests can be found in the GitHub repository.



Search to Rules Using Gemini AI

Demo

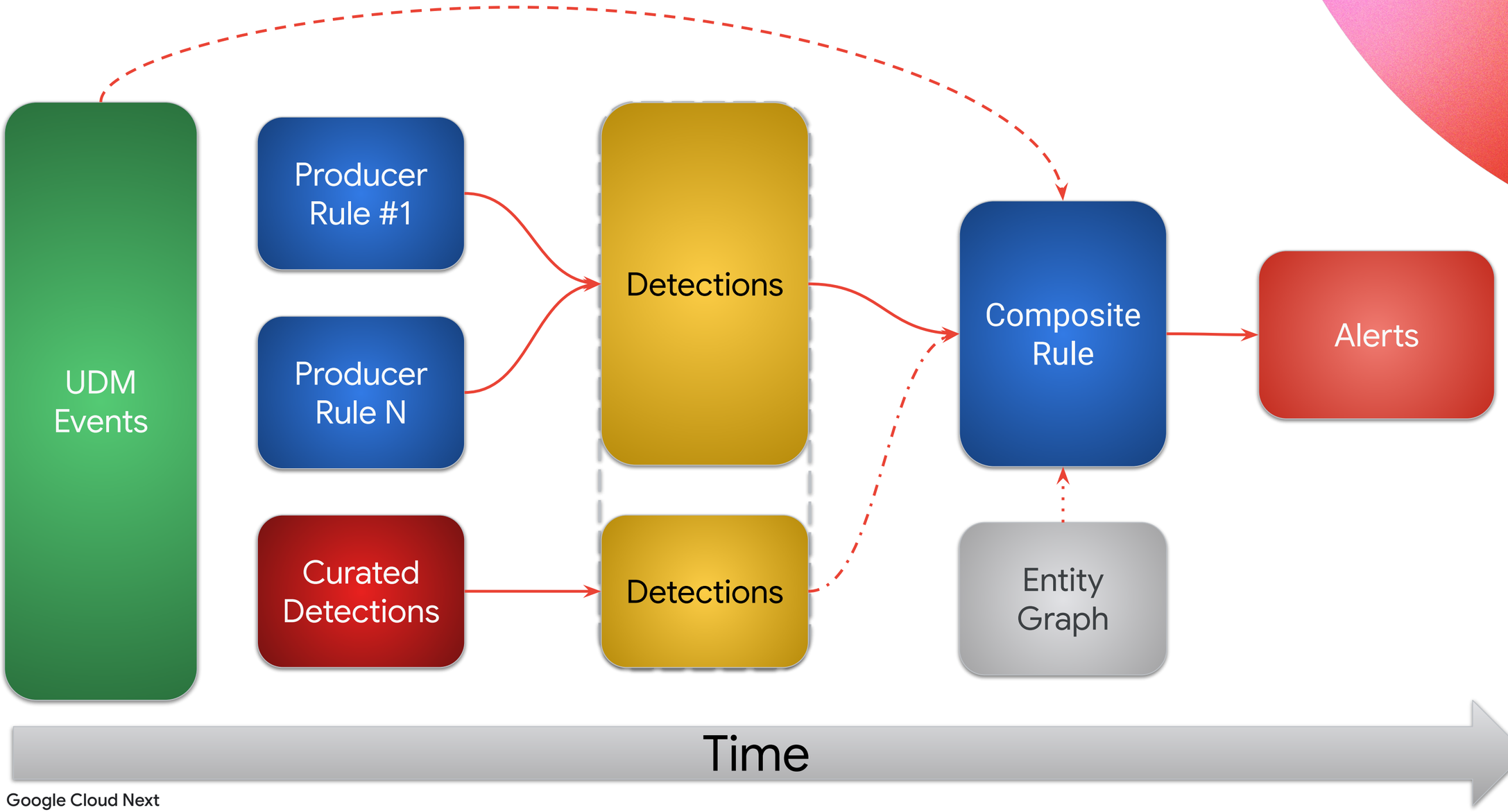

Google SecOps
Search (Legacy)
JS
Google Security Operations
2025-03-04 20:06:57
Enter a hostname, domain, IP, URL, email, username, or file hash
SEARCH
Total Log Entries: 3,886,638 Bytes Ingested: 6,043,853,280

Composite Rules

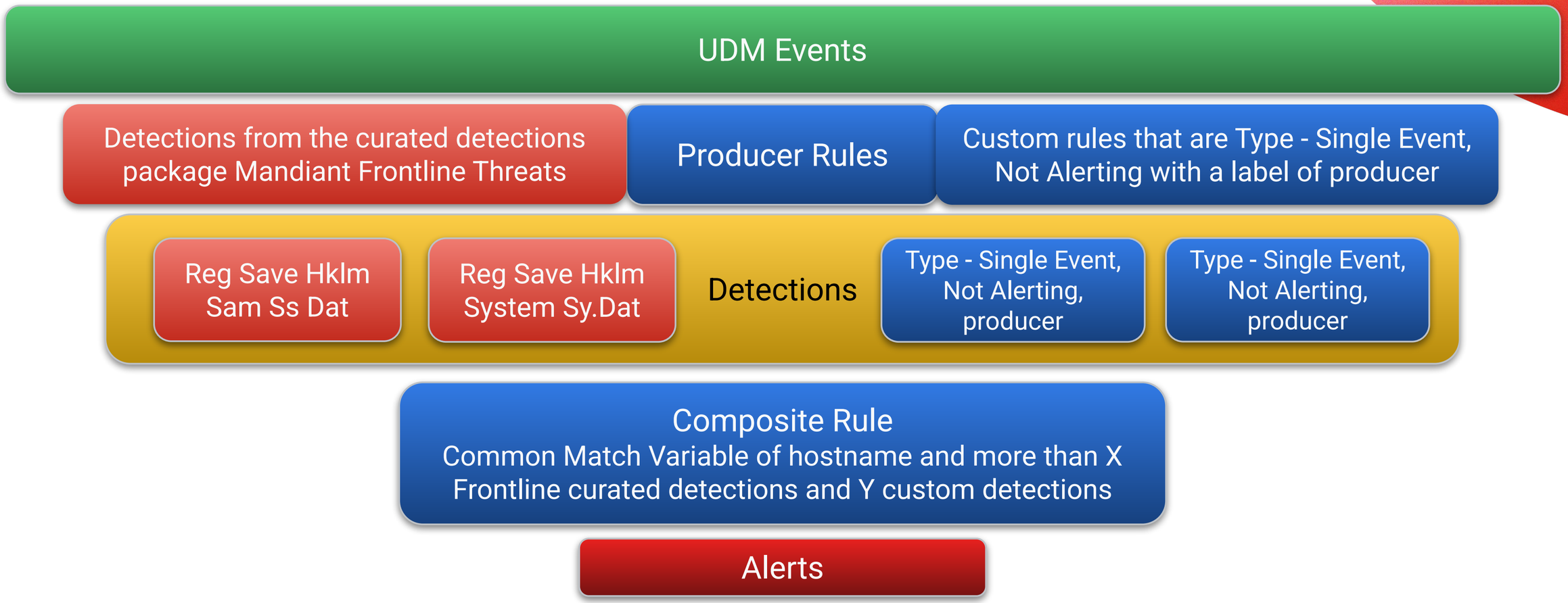
Provides detection engineers and threat hunters the ability to build more flexible and advanced detections

- Break large, complex detections into smaller, atomic detections
- Re-use smaller detections over and over
- Leverage common values in the smaller detections to build composite detections

Composite Detections



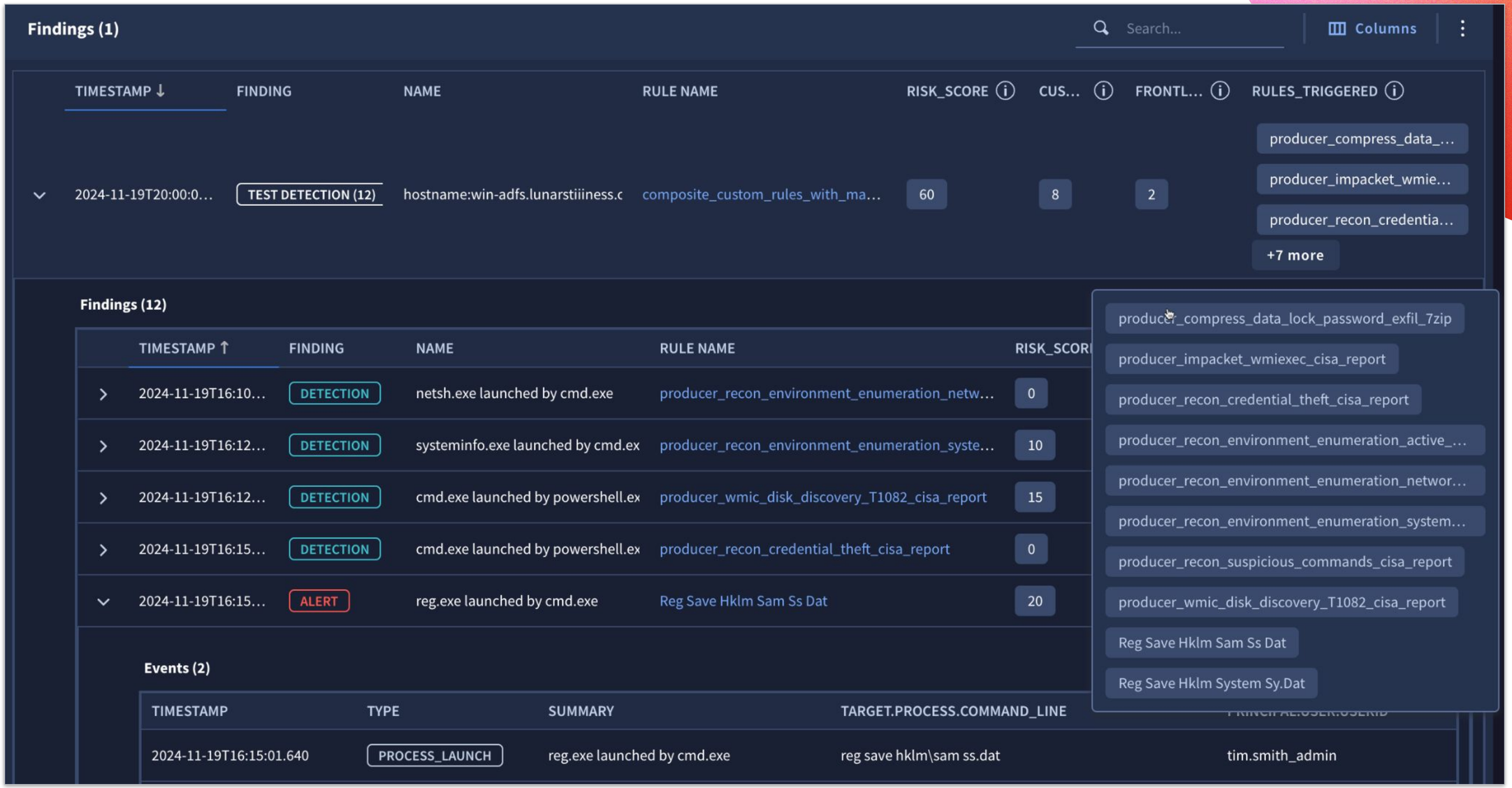
Composite Detections Curated & Custom Rules



Building a Composite Detection

```
events:
  $detect_prod.detection.detection.rule_id = /^ru_/
  $detect_prod.detection.detection.alert_state = "NOT_ALERTING"
  $detect_prod.detection.detection.rule_type = "SINGLE_EVENT"
  $detect_prod.detection.detection.rule_labels["type"] = "producer"
  $detect_prod.detection.detection.detection_fields["hostname"] = $hostname

  $detect_frontline.detection.detection.rule_set_display_name = "Mandiant Frontline Threats"
  $detect_frontline.detection.detection.detection_fields["hostname"] = $hostname

match:
  $hostname over 4h

outcome:
  $risk_score = 60
  $custom_rule_distinct_count = count_distinct($detect_prod.detection.detection.rule_id)
  $frontline_rule_distinct_count = count_distinct($detect_frontline.detection.detection.rule_id)
  $rules_triggered = arrays.concat(array_distinct($detect_prod.detection.detection.rule_name),
      array_distinct($detect_frontline.detection.detection.rule_name))

condition:
  $detect_prod and $detect_frontline and $custom_rule_distinct_count > 3 and
      $frontline_rule_distinct_count > 1
```

Composite Detections



Composite Detection - User Risk

```
events:
  $detect_prod.detection.detection.outcomes["principal_user_userid"] = $user
match:
  $user over 24h
outcome:
  $risk_score = 60
  $uniq_detection_count = count_distinct($detect_prod.detection.detection.rule_id)
  $total_detection_count = count($detect_prod.detection.detection.rule_id)
  $rules_triggered = array_distinct($detect_prod.detection.detection.rule_name)
  // sum of the risk score to measure against the threshold
  $cumulative_risk_score = sum($detect_prod.detection.detection.risk_score)
condition:
  $detect_prod and $cumulative_risk_score >= 90
```

Composite Detection - User Risk

TIMESTAMP	FINDING	NAME	RULE NAME	CUMUL...	UNIQ_DETE...	RULES_TRIGGERED
2024-11-20T00:00:00.000	TEST DETECTION (10)	user:tim.smith_admin	composite_cumulative_risk_scor...	175	7	producer_compress_data_lock_password... producer_recon_credential_theft_cisa_re... producer_recon_environment_enumerati... +4 more

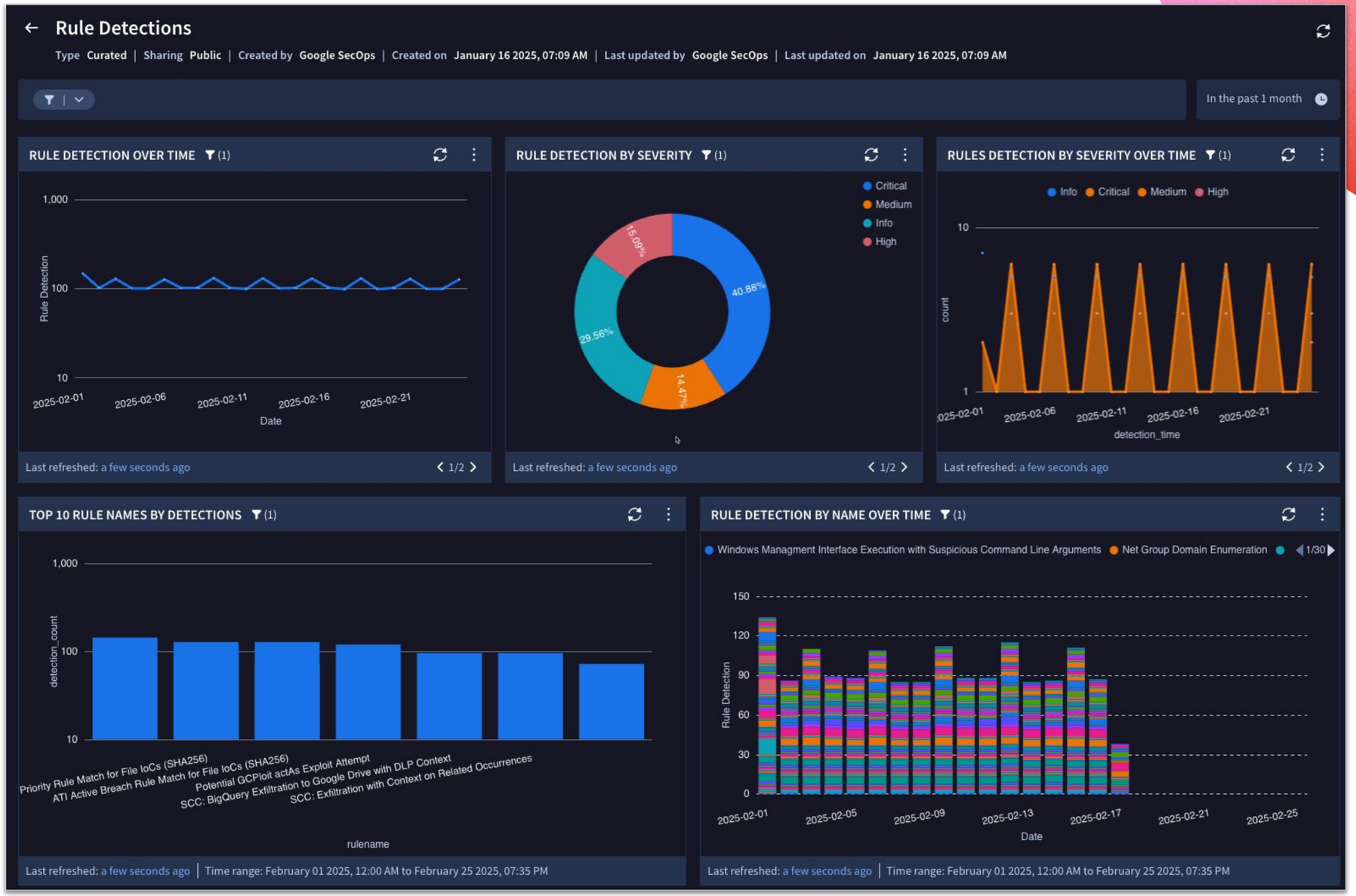
Findings (10)

TIMESTAMP	FINDING	NAME	RULE NAME	DETECTION.RISK_SCORE
2024-11-19T16:30:00.000	DETECTION	cmd.exe launched by powershell.exe	producer_recon_credential_theft_cisa_report	10
2024-11-19T16:28:30.000	DETECTION	cmd.exe launched by powershell.exe	producer_recon_environment_enumeration_active_direct...	15
2024-11-19T16:27:00.000	DETECTION	cmd.exe launched by powershell.exe	producer_recon_environment_enumeration_system_cisa...	10

Events (10)

TIMESTAMP	TYPE	SUMMARY	PRINCIPAL.USER.USERID	TARGET.PROCESS.COMMAND_LINE
2024-11-19T16:12:05.116	PROCESS_LAUNCH	cmd.exe launched by powershell.exe	tim.smith_admin	"cmd.exe" /c wmic product list brief
2024-11-19T16:12:13.933	PROCESS_LAUNCH	cmd.exe launched by powershell.exe	tim.smith_admin	"cmd.exe" /c wmic baseboard list full
2024-11-19T16:12:48.933	PROCESS_LAUNCH	cmd.exe launched by powershell.exe	tim.smith_admin	"cmd.exe" /c wevtutil qe security /rd:true /f:text /q:
2024-11-19T16:14:09.859	PROCESS_LAUNCH	cmd.exe launched by powershell.exe	tim.smith_admin	"cmd.exe" /c reg query hklm\software\realvnc

Visualizing Detections



04.
Detection-as-Code

Detection-as-Code is a methodology that applies software engineering principles to the creation and management of detection content

Motivations for Adopting DaC

Collaboration

A group of practitioners with unique insights working together results in more effective rules

Change Management

Every modification is tracked, reviewed, and approved leaving behind a clear audit trail

Automation

CI/CD pipeline automates the testing and deployment of detection rules

Version Control

Changes to detection content are tracked in git, which makes it easy to roll back changes if needed

Scalability

Managing a library of rules across multiple tools, environments, and customers is easier

Knowledge Management

Comprehensive documentation ensures that the detection’s strategy, triage & response steps, assumptions, blind spots, etc are available

Typical DaC Workflow

Propose Changes

Detection Engineer creates a new pull request in GitHub with their proposed rule changes

Example changes include creating a new rule or updating an existing rule

Run Tests

CI/CD pipeline job runs tests

Check for invalid rule configuration, duplicate rule names, verify rule syntax, etc

Execute tests to trigger rules and validate alert generation

Review & Approve

Security team discusses and collaborates on proposed changes in pull request

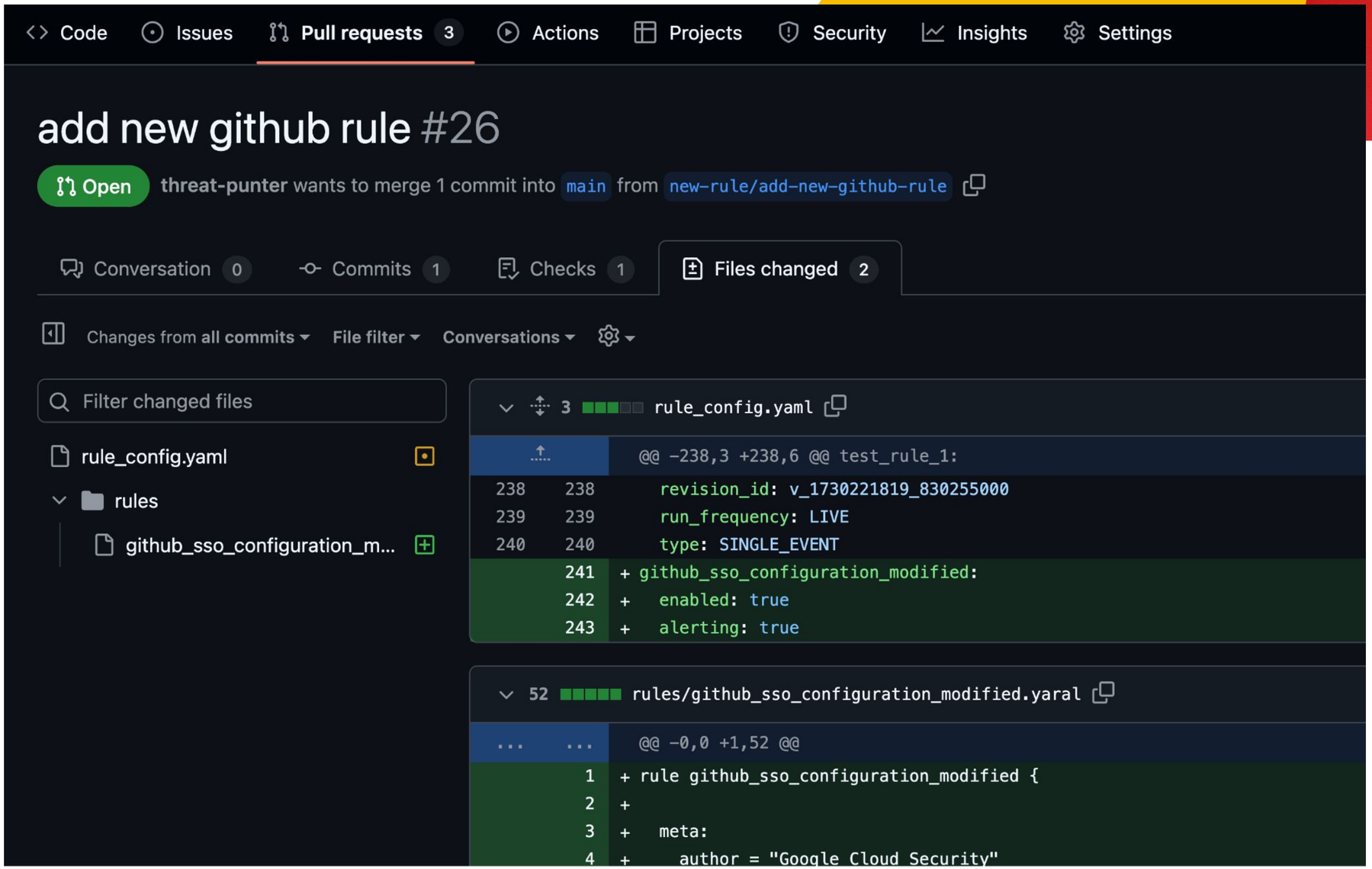
Changes are approved by one or more members of the security team

Deploy Changes

Changes are merged into the main branch of the GitHub repo

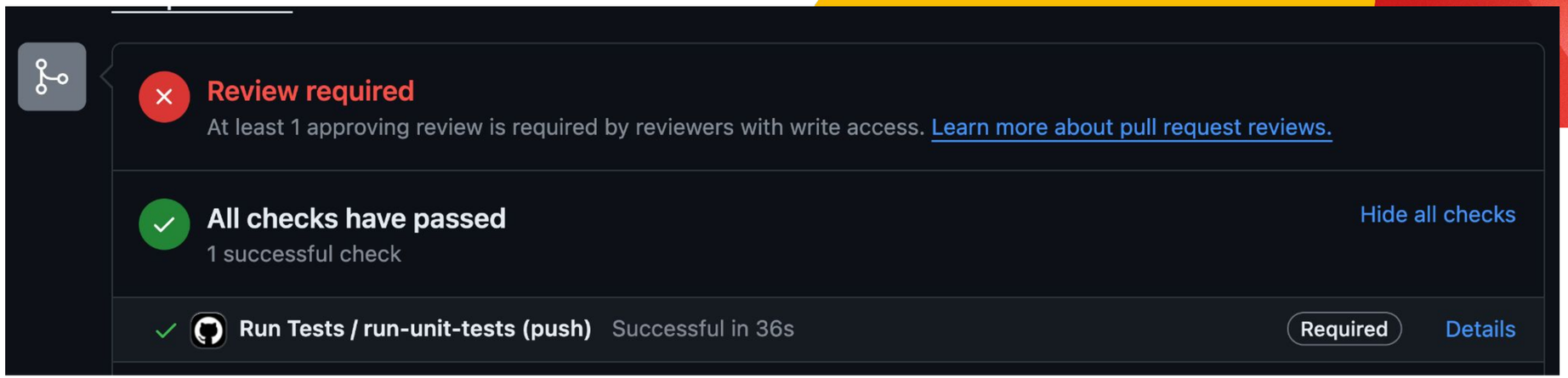
CI/CD pipeline detects changes to the main branch and pushes any pending updates to Google SecOps

Creating a New Rule (1)

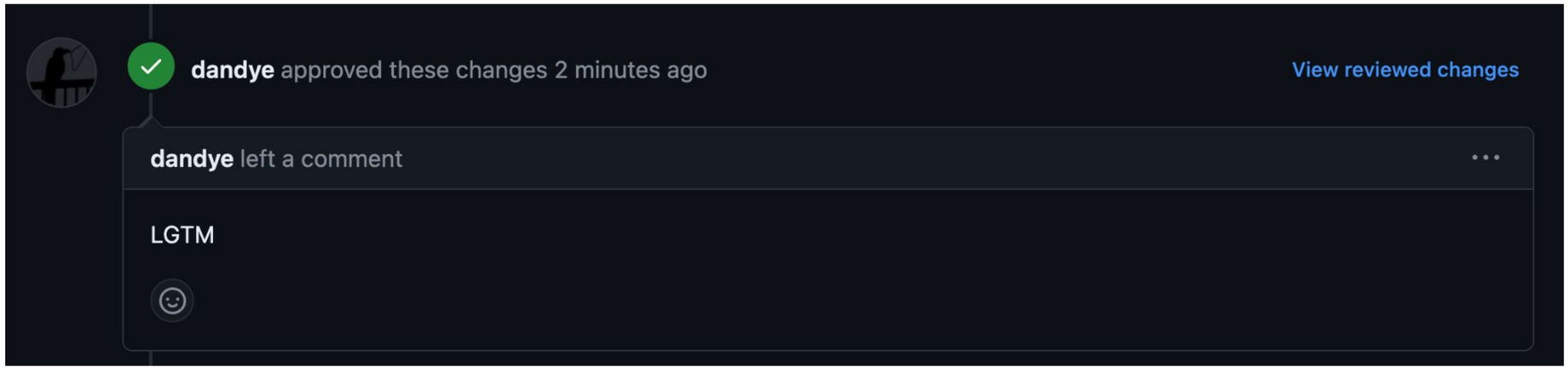


Creating a New Rule (2)

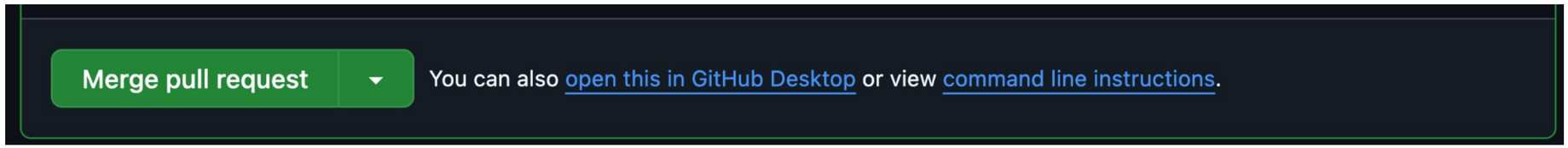
1



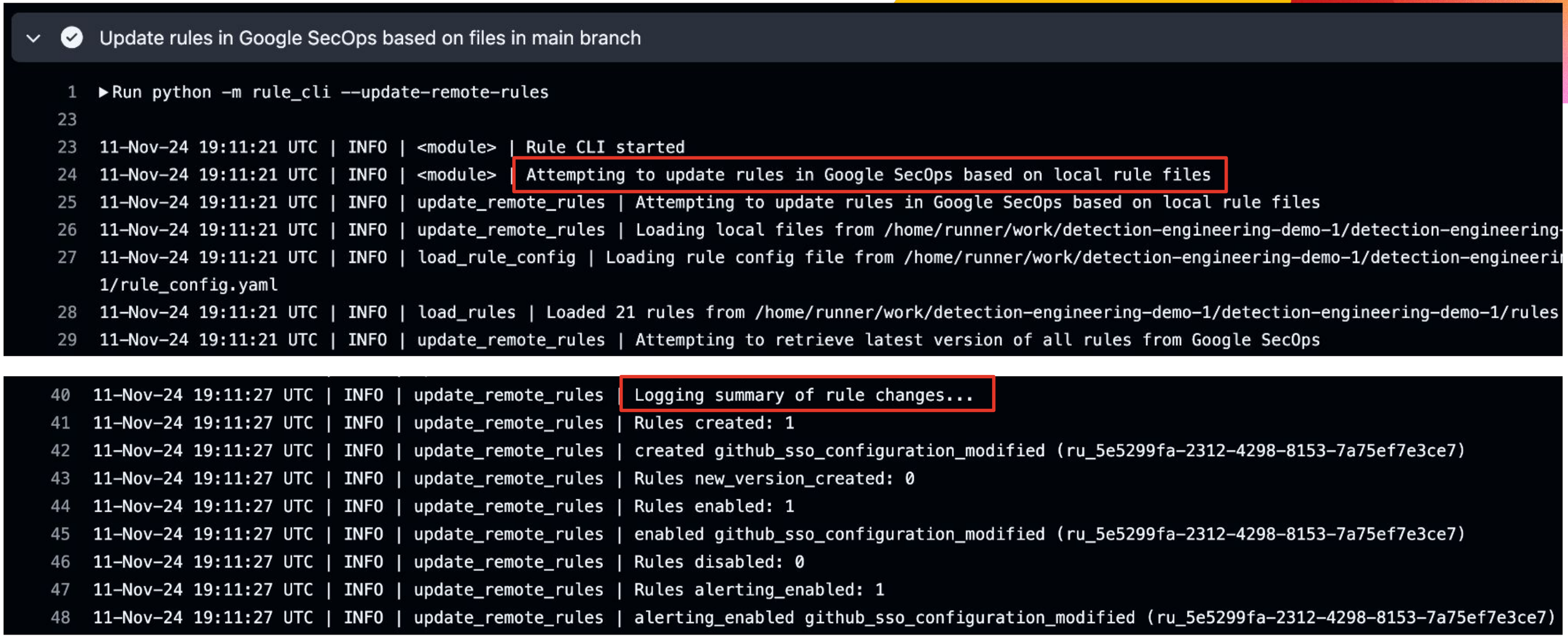
2



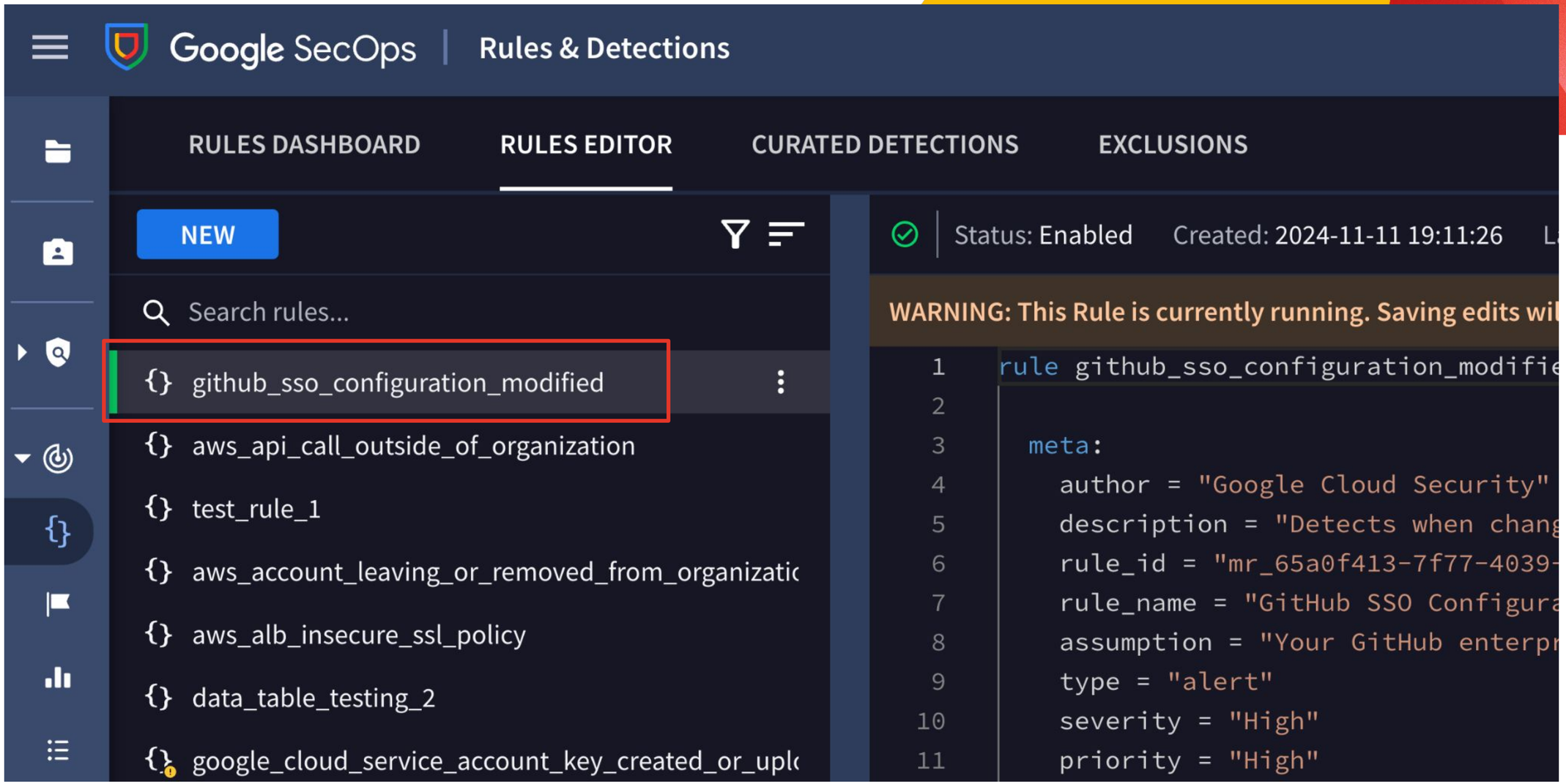
3



Creating a New Rule (3)



Creating a New Rule (4)



Implementation Considerations

Source of truth for detection content

For example, GitHub or Google SecOps

Should your pipeline overwrite any changes made outside your DAC pipeline?

Change control

If all changes should go through your DAC pipeline, should the permission to modify files in the UI be restricted?

Monitoring for rule changes

Is the behavior of your content being modified outside of your DAC pipeline suspicious?

You can detect this behavior using Google SecOps

Resources

SecOps Community

Umbrella site with loads of resources to engage with others on all things SecOps!

New to Google SecOps Blog

Blog series with over 50 blogs on building content using many of the tools we discussed today!

How To Videos

Over 40 videos that demonstrate how to build a rule, use functions, apply metrics to rules, integrate entity graph data and more!

Community-Driven Detection Content

Repository of YARA-L rules that can be used to accelerate adoption as well as share & collaborate on detection content with community

Forums

Not clear on how to write that last line of YARA-L? Not sure what the best function to use is? Come and engage the community and ask!

Community Blog

Everything from threat research to integrating with GCP audit logs, Detection-as-Code how tos, and technical tutorials and more!

Final Thoughts...

- Leverage threat intelligence to focus detection engineering
- Curated detections provide rules to get your started - Cloud, Windows, UEBA, Active Breaches and more!
- Generative AI can help get you started with detection logic
- Functions and Entity Graph provide additional flexibility to craft your own rules with more on the way!
- It’s easy to implement Detection-as-Code with Google SecOps

Your feedback is greatly appreciated!

Complete the session survey in the mobile app

Thank you