

From soup to nuts: Building a Detection-as-Code pipeline

David French

Staff Adoption Engineer, Google Cloud

[@threatpunter](#)

Amsterdam 2024 FIRST Technical Colloquium

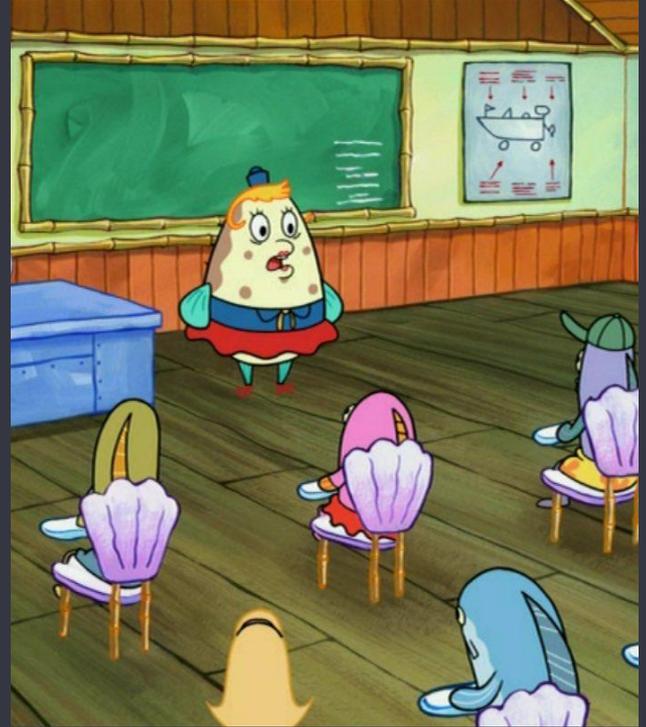
David French / About Me

- 18+ years in IT and cybersecurity
 - Blue team life: Detection Engineer, Threat Hunter, SOC Analyst
 - Vendor life: Threat Research, Detection Engineering, building SIEMs & EDRs
- Currently at Google Cloud (Chronicle Security Operations)
- Formerly Twilio, Elastic, Endgame, Capital Group
- Speaker at Black Hat and BSides
- Creator of [Dorothy](#) - Adversary simulation tool for Okta
- Likes to share knowledge & research: [Blog](#), [community contributions](#), MITRE ATT&CK
- Enjoys hiking, fishing, cycling, etc



Intended audience

- Anyone curious about how to manage detection content “as code” and how to get started
- Defensive security practitioners: Detection Engineers, SOC Analysts, etc
- Maybe you manage rules/signatures manually in your security tools and want to automate that
- If you’re already an expert in Detection-as-Code, you might not learn a ton 😊



Agenda

1. What is Detection-as-Code?
2. Example Detection Engineering workflow with Detection-as-Code
3. Benefits of managing detection rules “as code”
4. Designing the pipeline
5. Building a pipeline to manage detection content
6. Wrap up
 - a. Key takeaways
 - b. Links to useful resources
 - c. Q&A

What is Detection-as-Code (DaC)?

- A set of principles that use code and automation to implement and manage threat detection content
- Traditional approach: Security team manually configures rules & signatures in security tools
- Detection-as-Code: Leverages software development practices & tools and treats detection content as code artifacts
- Gaining in popularity; growing acceptance

[Can We Have “Detection as Code”?](#) — Anton Chuvakin

[Automating Detection-as-Code](#) — John Tuckner

[Detection-as-code: Why it works and where to start](#) — Kyle Bailey

[Detection as Code: Detection Development Using CI/CD](#) — Patrick Bareiß, Jose Hernandez

Can We Have “Detection as Code”?



Anton Chuvakin · Follow

Published in Anton on Security · 5 min read · Sep 21, 2020



268



4



Automating Detection-as-Code

Last updated on September 28, 2022



Written by John Tuckner

Head of Research at Tines Labs, Tines

Detection-as-Code

Why it works and where to start.

Kyle Bailey (@kylebailey22)

Security Engineer @ Panther Labs

Core technologies to automate detection content management

Version Control System (VCS)

Software that tracks changes to code over time

Facilitates structured development processes rollbacks

Examples: Git, Subversion, Mercurial

Software Development Platform

Provides a centralized workspace for managing Git repositories

Provides issue tracking, pull requests, code reviews, etc

Examples: GitHub, GitLab, Bitbucket

Continuous Integration / Continuous Delivery Tools

CI/CD tools automate the building, testing, and deployment of code changes

Examples: Jenkins, CircleCI, GitLab CI/CD, GitHub Actions

Example Detection-as-Code workflow

Propose Changes

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

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

Run Tests

GitLab 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 GitLab project

A CI/CD pipeline detects changes to the main branch and pushes any pending updates to the SIEM

The latest version of all rules is pulled from the SIEM and committed to the repo to include updated metadata



Benefits of managing detection rules as code

Benefits of DaC: Collaboration (1)

- Challenge with traditional method of managing detection rules: People make mistakes
- DaC makes it easy for the team to discuss and contribute to changes to detection content
- A group of practitioners with unique insights working together will result in more accurate and effective rules
- Peer review reduces risk
 - False negatives
 - False positive explosions

New Rule - DNS query to typosquatting domain

[Open](#) requested to merge [add-new-typosquatting-rule](#) into [main](#) just now

Overview 1 **Commits** - **Pipelines** 0 **Changes** 2

This rule detects DNS queries to domains that contain one of our company's domains names that is not registered with our expected domain registrar.

👍 0 🗑️ 0 😬

17 + rule whois_dns_query_to_typosquatting_domain {

David French @frenchee1 · just now Maintainer ✓ 😬 ✎ ⋮

Does this rule overlap with `whois_dns_query_to_namecheap_domain`?

Does it provide additional coverage that we don't have today?

Have we considered expanding the existing rule to cover this detection use case?

Reply... Resolve thread 🗑️

8 ✓ **Approved by** 🧑

✓ **Ready to merge!**

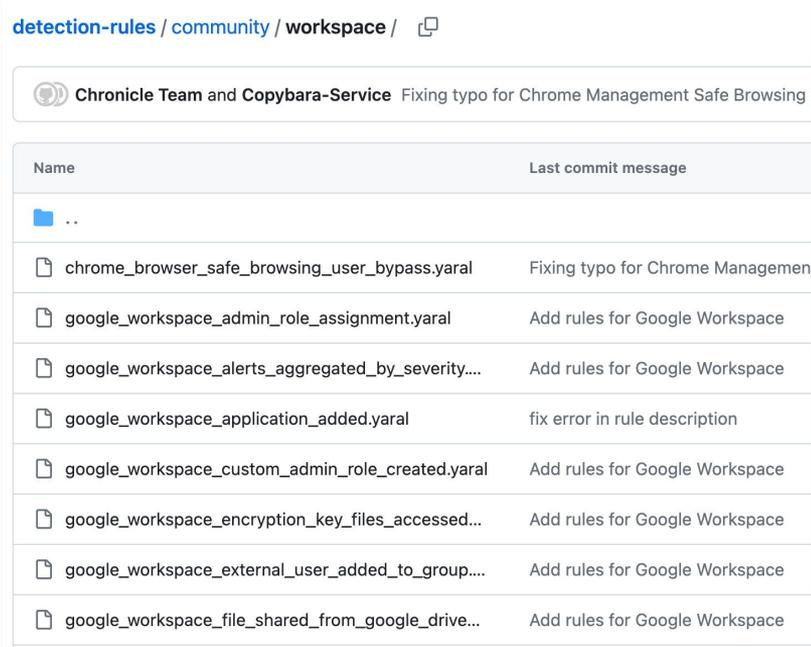
Delete source branch Squash commits [?](#) Edit commit message

1 commit and 1 merge commit will be added to main.

Merge

Benefits of DaC: Collaboration (2)

- Easier to share detection content with the security community; stronger defense against attacks
 - Google: <https://github.com/chronicle/detection-rules>
 - Elastic: <https://github.com/elastic/detection-rules>
 - Splunk: https://github.com/splunk/security_content
 - Microsoft: <https://github.com/Azure/Azure-Sentinel>
 - Sigma: <https://github.com/SigmaHQ/sigma>

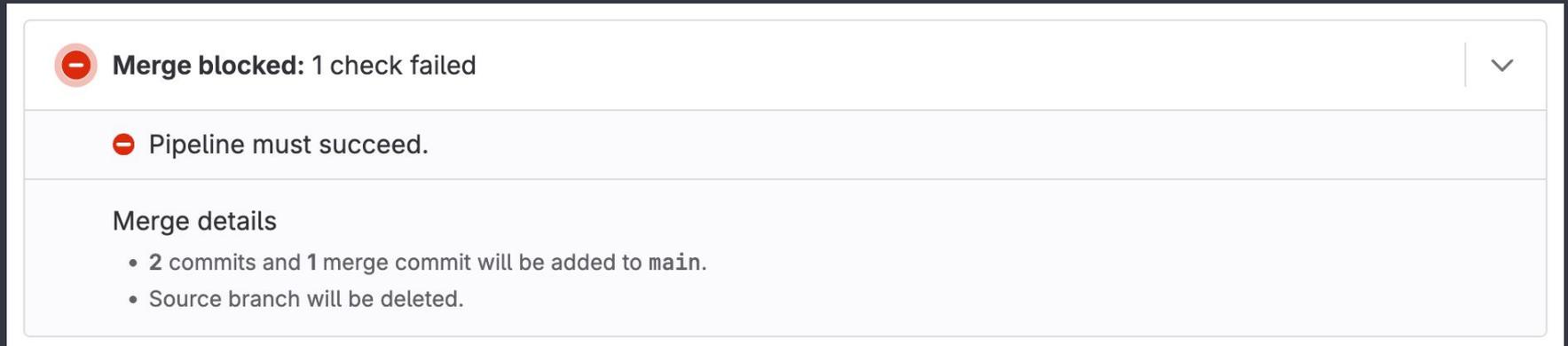


The screenshot shows a GitHub repository page for 'detection-rules / community / workspace'. The page displays a commit history table with the following columns: 'Name' and 'Last commit message'. The table lists several commits, including one from the 'Chronicle Team and Copybara-Service' and several from 'google_workspace_...'.

Name	Last commit message
..	
chrome_browser_safe_browsing_user_bypass.yaral	Fixing typo for Chrome Management
google_workspace_admin_role_assignment.yaral	Add rules for Google Workspace
google_workspace_alerts_aggregated_by_severity...	Add rules for Google Workspace
google_workspace_application_added.yaral	fix error in rule description
google_workspace_custom_admin_role_created.yaral	Add rules for Google Workspace
google_workspace_encryption_key_files_accessed...	Add rules for Google Workspace
google_workspace_external_user_added_to_group...	Add rules for Google Workspace
google_workspace_file_shared_from_google_drive...	Add rules for Google Workspace

Benefits of DaC: Change management

- DaC provides more control over changes made to detection content
- Detection content stored in a software development platform e.g. GitHub, GitLab
- Changes are tested, reviewed, and approved before getting deployed to prod
- Some organizations require robust change control for both preventive **and** detective security controls



The screenshot shows a GitHub notification for a blocked merge. At the top, a red circle with a minus sign is followed by the text "Merge blocked: 1 check failed". To the right of this text is a vertical line and a downward-pointing chevron icon. Below this, a red circle with a minus sign is followed by the text "Pipeline must succeed.". Underneath, the heading "Merge details" is followed by two bullet points: "2 commits and 1 merge commit will be added to main." and "Source branch will be deleted."

Merge blocked: 1 check failed

– Pipeline must succeed.

Merge details

- 2 commits and 1 merge commit will be added to main.
- Source branch will be deleted.

Benefits of DaC: Automation

- CI/CD tools used to ensure continuous process for building, testing, and deploying changes to detection content
- Tests reduce risk of introducing false positives/negatives
 - Reduce problem of alert fatigue
- Test in dev before deploying to prod

Triggered via pull request 3 days ago

Status	Total duration	Billable time	Artifacts
Success	11m 18s	14m	—

test_and_deploy_in_dev.yml
on: pull_request

```
graph LR; A[validate-terraform-configu... 9s] --> B[deploy-to-dev 12s]; B --> C[trigger-detections-in-dev 11s]; C --> D[check-for-alerts 10m 15s];
```

The screenshot displays a successful CI/CD pipeline run. The top section shows the run was triggered by a pull request, with a status of 'Success', a total duration of 11m 18s, and a billable time of 14m. Below this, the pipeline configuration for 'test_and_deploy_in_dev.yml' is shown, triggered on a pull request. The pipeline consists of four sequential steps, all of which are marked as successful with green checkmarks: 'validate-terraform-configu...' (9s), 'deploy-to-dev' (12s), 'trigger-detections-in-dev' (11s), and 'check-for-alerts' (10m 15s).

Designing & building the pipeline

Pipeline design



```
▼ DETECTION-ENGINEERING
  > chronicle_api
  > rule_cli
  ▼ rules
    ≡ google_workspace_mfa_disabled.yaral
    ≡ ioc_domain_internal_policy.yaral
    ≡ okta_new_api_token_created.yaral
    ≡ suspicious_asn_watchlist_1.yaral
    ≡ whois_dns_query_to_typosquatting_domai...
```

GitLab CI/CD Pipeline Jobs

Run Tests

Get rules

Update rules

SIEM

Write code to read, create, update,
and verify rules via the SIEM's API

Software development platform
and version control system (VCS)

Managing detection rules via an API (1)

- At this point, we're assuming:
 - We have some rules configured in our SIEM
 - Our SIEM has an API endpoint for managing rules
- SIEM vendors may provide example code or engineers may have to write it themselves
- Users expect parity between what they can do in the UI of a security tool versus the API

```
▼ DETECTION-ENGINEERING
  ▼ siem_api
    ▼ rules
      __init__.py
      create_rule.py
      get_rule_deployment.py
      get_rule.py
      list_rules.py
      test_create_rule.py
      test_get_rule_deployment.py
      test_get_rule.py
      test_list_rules.py
      test_update_rule_deployment.py
      test_update_rule.py
      test_verify_rule.py
      update_rule_deployment.py
      update_rule.py
      verify_rule.py
```

Managing detection rules via an API (2)

- Python modules are wrapped in a simple CLI to use in CI/CD pipeline jobs in GitHub, GitLab, etc
- Additional modules & logic written to handle logic for updating rules

```
$ python -m rule_cli --help
19-Jan-24 14:41:10 MST | INFO | <module> | Rule CLI started
usage: __main__.py [-h] [--pull-latest-rules] [--update-remote-rules] [--verify-rules]
                  {verify-rule} ...

rule_cli

options:
  -h, --help                show this help message and exit
  --pull-latest-rules       Retrieves the latest version of all rules from Chronicle and writes
                             them to local files.
  --update-remote-rules     Update rules in Chronicle based on local rule files.
  --verify-rules            Verify that all local rules are valid YARA-L 2.0 rules.

subcommands:
  {verify-rule}
    verify-rule             Verify that a rule is a valid YARA-L 2.0 rule.
$
```

Managing detection rules via an API (3)

- Some teams use Infrastructure-as-Code tools to manage SIEM rules & configuration
 - e.g. Terraform, Pulumi
- Code is stored in central repository and CI/CD jobs “apply” changes to “infrastructure” (security tools)
- These tools can overwrite changes made in the UI if that’s your desired behavior

```
rule_okta_administrator_role_assigned_to_non_admin_user_account.tf > ...
1 module "rule_okta_administrator_role_assigned_to_non_admin_user_account" {
2   source = "../modules/sumo_monitor"
3   standard_name = "Administrator Role Assigned to Non-Admin User Account"
4   standard_description = "Identifies when an administrator role is assigned to a
  non-admin Okta user account i.e. a standard user account that does not follow our
  company's admin account naming conventions. Investigate using playbook PB-100."
5   standard_query = <<EOF
6   _sourceCategory="okta" user.account.privilege.grant
7 ] where eventType="user.account.privilege.grant" AND !("%"target[0].alternateId" matches /
  ^admin\./)
8 EOF
9   standard_folder = sumologic_monitor_folder.detections.id
10  tines_webhook = sumologic_connection.tines_webhook.id
11  tines_webhook_override = <<EOF
12 {
13   "rule.name": "${Name}",
14   "rule.description": "${Description}",
15   "query.url": "${QueryURL}",
16   "query": "${Query}",
17   "trigger.range": "${TriggerTimeRange}",
18   "trigger.name": "${TriggerTime}",
19   "alert.payload": "${ResultsJson}"
20 }
21 EOF
22 }
```

```
threatpunter detection-as-code % terraform apply
sumologic_connection.tines_webhook: Refreshing state... [id=0000000000003948D]

Terraform used the selected providers to generate the following execution plan.
+ create

Terraform will perform the following actions:

# sumologic_monitor_folder.detections will be created
+ resource "sumologic_monitor_folder" "detections" {
+   content_type = "Folder"
  (...)
}
```

GitLab project layout

Python modules for managing rules via SIEM's API

SIEM rules stored as code artifacts

GitLab CI/CD pipeline configuration file

Name
chronicle_api
rule_cli
rules
.gitignore
.gitlab-ci.yml
CONTRIBUTING.md
LICENSE
README.md
requirements.txt
requirements_dev.txt
rule_config.yaml

CLI with commands to retrieve, update, and verify rules via SIEM's API

Rule configuration file

Used to configure rule state (e.g. enabled/disabled/archived)

Used to store rule metadata (e.g. rule ID, create time, etc)

Defining a rule schema: Benefits

- Provides a way to structure and standardize rules
- Ensures rule structure is consistent across authors
- Define which parts are required/optional
- Automation - Easier to validate, test, and deploy detection content if it's in a consistent format
- Easier to share rules within the community
- Example of a schema using [Pydantic](#) 

```
33 from pydantic import BaseModel
34
35
36 class Rule(BaseModel):
37     """Data class for a YARA-L rule."""
38
39     name: str
40     id: str | None
41     resource_name: str | None
42     create_time: str | None
43     revision_id: str | None
44     revision_create_time: str | None
45     enabled: bool
46     alerting: bool
47     archived: bool | None
48     archive_time: str | None
49     run_frequency: str | None
50     type: str | None
51     text: str
```

Defining a rule schema: Popular formats

- YAML - Used by [Splunk](#) and [Sigma](#)
- TOML - Used by [Elastic](#)
- Example of a YARA-L rule in TOML format →
- I decided to decouple the rule config & metadata from the rule logic
 - Granular control over deploying to multiple SIEM instances (e.g. if you're deploying to dev, prod, etc or an MSSP deploying to multiple customers)

```
rules/ag_ioc_sha256_hash_vt_basic.toml 0 → 100644
1 + ruleId = "ru_2b5db5f7-af6a-400b-81d7-7b458ec1f0f3"
2 + versionId = "ru_2b5db5f7-af6a-400b-81d7-7b458ec1f0f3@v_17
3 + versionCreateTime = "2023-11-18T05:00:23.398703Z"
4 + liveRuleEnabled = true
5 + alertingEnabled = true
6 + archived = false
7 + ruleType = "MULTI_EVENT"
8 + ruleText = ""rule ag_ioc_sha256_hash_vt_basic {
9 +
10 +   meta:
11 +     author = "Google Cloud Security"
12 +     description = "Used for the Alert Graph Workshop. Det
13 +     docx file types"
14 +     type = "alert"
15 +     tags = "threat indicators, vt enrichment"
16 +     assumption = "Assumes MISP data has been ingested int
17 +     data_source = "microsoft sysmon"
18 +     severity = "Medium"
19 +     priority = "Medium"
20 +
```

Validating rules against a schema

- Catch issues as early as possible; minimize risk of deploying broken rules
 - Missing/invalid values
 - Misconfigurations e.g. a rule that's enabled cannot be archived until it's disabled
 - Invalid rule/file names
- [Pydantic](#) and [Marshmallow](#) are great for this 

```
07-Feb-24 10:55:38 MST | ERROR | <module> | ValidationError occurred for rule
[
  {
    "type": "bool_type",
    "loc": [
      "enabled"
    ],
    "msg": "Input should be a valid boolean",
    "input": null,
    "url": "https://errors.pydantic.dev/2.6/v/bool_type"
  }
]
```

Verifying rule syntax

- Options to verify the syntax of a rule:
 - Via your SIEM's API if supported
 - Develop your own linter for rule parsing & validation (++ effort to create and maintain)
- Some SIEMs prevent a rule from being created/modified if syntax errors are found

```
422 19-Jan-24 23:42:19 UTC | INFO | verify_rules | Rule verification succeeded for 3 rules
423 19-Jan-24 23:42:19 UTC | ERROR | verify_rules | Rule verification failed for 1 rules
424 19-Jan-24 23:42:19 UTC | ERROR | verify_rules | Rule verification failed for rule (/builds/
response: {
425   "compilationDiagnostics": [
426     {
427       "message": ": accessing field \"udm.metadata.product_namee\": field \"product_namee\" does not exist, valid fields are: \"id\", \"p
ted_timestamp\", \"ingested_timestamp\", \"event_type\", \"vendor_name\", \"product_name\", \"product_version\", \"product_event_type\", \"prod
ck_to_product\", \"ingestion_labels\", \"tags\", \"enrichment_state\", \"log_type\", \"base_labels\", \"enrichment_labels\"\nline: 17 \ncolumn:
428       "position": {
429         "startLine": 17,
430         "startColumn": 6,
431         "endLine": 17,
432         "endColumn": 34
433       },
434       "severity": "ERROR"
```

Pulling the latest rules from the SIEM

- We need to keep the GitLab project up-to-date with the latest version of all rules in the SIEM
- CLI argument pulls latest rules from SIEM and writes rule files and rule config file

```
422 Attempting to pull the latest version of all rules from Chronicle
423 $ python -m rule_cli --pull-latest-rules
424 19-Jan-24 23:03:35 UTC | INFO | <module> | Rule CLI started
425 19-Jan-24 23:03:35 UTC | INFO | <module> | Attempting to pull latest version of all Chronicle rules and update local files
426 19-Jan-24 23:03:35 UTC | INFO | get_remote_rules | Attempting to retrieve all rules from Chronicle
427 19-Jan-24 23:03:36 UTC | INFO | get_remote_rules | Retrieved 3 rules
428 19-Jan-24 23:03:36 UTC | INFO | get_remote_rules | Retrieved a total of 3 rules
429 19-Jan-24 23:03:36 UTC | INFO | get_remote_rules | Attempting to retrieve rule deployment state for 3 rules
430 19-Jan-24 23:03:38 UTC | INFO | dump_rules | Writing 3 rule files to /builds/
431 19-Jan-24 23:03:38 UTC | INFO | dump_rule_config | Writing rule config to /builds/
g.yaml
```

Dumping the rule logic and rule configuration

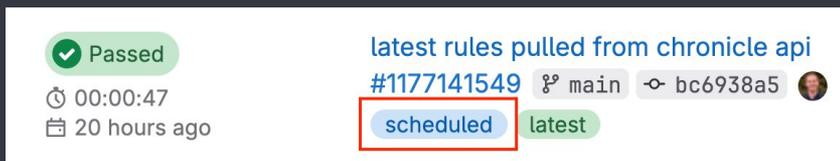
- Rule logic is written to the `rules` directory
- Rule configuration and metadata is written to a `rule_config.yaml` file

```
▼ rules
  google_workspace_mfa_disabled.yaral
  google_workspace_multiple_files_sent_as_email_attach.yaral
  google_workspace_new_trusted_domain_added.yaral
  google_workspace_password_policy_changed.yaral
  ioc_domain_internal_policy.yaral
  okta_new_api_token_created.yaral
```

```
rule_config.yaml ×
1  google_workspace_mfa_disabled:
2    alerting: true
3    archive_time: null
4    archived: false
5    create_time: '2024-01-19T23:01:45.962463Z'
6    enabled: true
7    id: ru_48a222d0-a3ea-4ed7-9b77-3794cd1d3844
8    resource_name: projects/1013673362905/locations/us/inst
9    revision_create_time: '2024-01-19T23:01:45.962463Z'
10   revision_id: v_1705705305_962463000
11   run_frequency: LIVE
12   type: SINGLE_EVENT
13   whois_dns_query_to_typosquatting_domain:
14     alerting: false
15     archive_time: null
16     archived: false
17     create_time: '2024-02-06T23:10:44.941199Z'
18     enabled: false
```

Syncing rules between the SIEM and GitLab

- CI/CD pipeline job runs on a schedule
- Pulls latest rules from SIEM
- Writes files containing rules and rule config
- Commits any changes to the main branch of the GitLab project



Passed

latest rules pulled from chronicle api

#1177141549 main bc6938a5

scheduled latest

00:00:47

20 hours ago

```
435 $ git status
436 On branch main
437 Your branch is up to date with 'origin/main'.
438 Changes to be committed:
439   (use "git restore --staged <file>..." to unstage)
440   new file:   rule_config.yaml
441   new file:   rules/google_workspace_mfa_disabled.yaral
442   new file:   rules/google_workspace_new_trusted_domain_added.yaral
443   new file:   rules/google_workspace_password_policy_changed.yaral
444 $ CHANGES=$(git status --porcelain | wc -L)
445 $ echo "There are $CHANGES changes to commit" && [ "$CHANGES" -gt "0" ] && git commit -m "latest rules pulled from chronicle api"
446 There are 4 changes to commit
447 [main 2ba0c7e] latest rules pulled from chronicle api
448 4 files changed, 200 insertions(+)
449 create mode 100644 rule_config.yaml
450 create mode 100644 rules/google_workspace_mfa_disabled.yaral
451 create mode 100644 rules/google_workspace_new_trusted_domain_added.yaral
452 create mode 100644 rules/google_workspace_password_policy_changed.yaral
453 To https://gitlab.com/... .git
454   f72ea3e..2ba0c7e  main -> main
455 $ echo "Current time is $(date)"
456 Current time is Fri Jan 19 23:03:40 UTC 2024
457 $ git log -1
458 commit 2ba0c7e101d563181062c685ace06a56cdc41837
459 Author: David French <...>
460 Date:   Fri Jan 19 23:03:38 2024 +0000
461     latest rules pulled from chronicle api
462 Cleaning up project directory and file based variables
463 Job succeeded
```

Example commit made by CI/CD job

Reviewing rule modifications that were made in the SIEM's UI

Showing 2 changed files with 4 additions and 4 deletions

Hide whitespace changes | Inline | Side-by-side

rule_config.yaml +2 -2 | View file @ edb32fbf

```
@@ -18,8 +18,8 @@ google_workspace_new_trusted_domain_added:
18 18     enabled: true
19 19     id: ru_00474bba-ed86-4f07-9595-2e1b2f756d14
20 20     resource_name: projects/[REDACTED]/locations/us/instances/[REDACTED]/rules/ru_00474bba-ed86-4f07-9595-2e1b2f756d14
21 -    revision_create_time: '2024-01-19T23:18:03.297999Z'
22 -    revision_id: v_1705706283_297999000
21 +    revision_create_time: '2024-01-19T23:20:09.328815Z'
22 +    revision_id: v_1705706409_328815000
23 23     run_frequency: LIVE
24 24     type: SINGLE_EVENT
25 25     google_workspace_password_policy_changed:
... ..
```

rules/google_workspace_new_trusted_domain_added.yaral +2 -2 | View file @ edb32fbf

```
@@ -25,8 +25,8 @@ rule google_workspace_new_trusted_domain_added {
25 25     mitre_attack_version = "v13.1"
26 26     type = "Alert"
27 27     data_source = "Workspace Activity"
28 -    severity = "Medium"
29 -    priority = "Medium"
28 +    severity = "High"
29 +    priority = "High"
30 30
```



latest rules pulled from chronicle api
David French authored 3 days ago



bc6938a5



Creating a new rule

Detection Engineer creates a GitLab pull request to create a new SIEM rule

New Rule - Okta Admin Role Assigned to Non-Admin User Account

David French requested to merge `new-rule-okta-admin-role-a` into `main` just now

Overview 0 Commits 0 Pipelines 0 **Changes 2** Mark as done

Compare `main` and latest version 2 files +43 -0

```
rules/okta_administrator_role_assigned_to_non_admin_user_account.yaral 0 → 100644 +40 -0 Viewed
```

```
1 + rule okta_administrator_role_assigned_to_non_admin_user_account {
2 +
3 + meta:
4 +   author = "Google Cloud Security"
5 +   description = "Detects when an administrator role is assigned to a non-admin Okta user account i.e. a standard
6 +   user account that does not follow our company's admin account naming conventions."
7 +   reference = "https://help.okta.com/en-us/content/topics/security/administrators-learn-about-admins.htm"
8 +   mitre_attack_tactic = "Persistence, Privilege Escalation"
9 +   mitre_attack_technique = "Account Manipulation"
10 +  mitre_attack_url = "https://attack.mitre.org/techniques/T1098/"
11 +  mitre_attack_version = "v14"
12 +  type = "Alert"
13 +  data_source = "Okta"
14 +  severity = "High"
15 +  priority = "High"
16 +
17 +  events:
18 +    $user.metadata.product_name = "Okta"
19 +    $user.metadata.vendor_name = "Okta"
20 +    $user.metadata.product_event_type = "user.account.privilege.grant"
21 +    $user.target.user.email_addresses != /*admin\./
22 +    $user.principal.user.userid = $userid
23 +
24 +  match:
```

```
rule_config.yaml
```

```
1 + okta_administrator_role_assigned_to_non_admin_user_account:
2 +   alerting: true
3 +   enabled: true
4 google_workspace_mfa_disabled:
5   alerting: true
6   archive_time: null
```

New Rule - Okta Admin Role Assigned to Non-Admin User Account

Merge request actions ▾

David French requested to merge `add-new-okta-rule-1` into `main` 6 days ago

Protecting the main branch

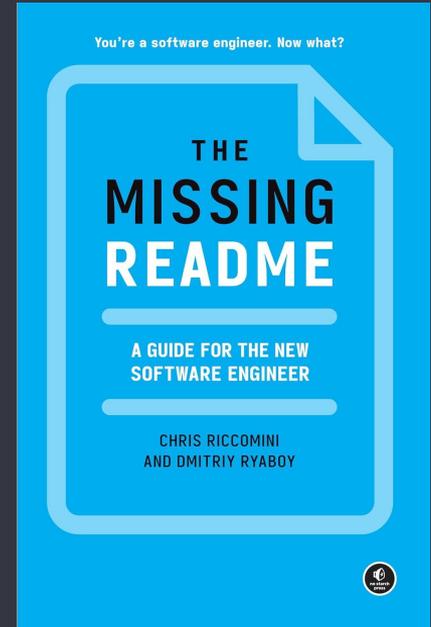
- Protect the main branch of your GitLab/GitHub project
- Prevent code from being merged until tests pass and approval is obtained



```
444 08-Feb-24 22:35:14 UTC | INFO | verify_rules | Rule verification succeeded for 9 rules
445 08-Feb-24 22:35:14 UTC | ERROR | verify_rules | Rule verification failed for 1 rules
446 08-Feb-24 22:35:14 UTC | ERROR | verify_rules | Rule verification failed for rule (/builds/
-engineering/rules/okta_administrator_role_assigned_to_non_admin_user_account.yaral). Response: {
447   "compilationDiagnostics": [
448     {
449       "message": ": accessing field \"udm.metadata.product_nameee\": field \"product_nameee\" does not exist, valid fields are:
\"id\", \"product_log_id\", \"event_timestamp\", \"collected_timestamp\", \"ingested_timestamp\", \"event_type\", \"vendor_name\", \"p
roduct_name\", \"product_version\", \"product_event_type\", \"product_deployment_id\", \"description\", \"url_back_to_product\", \"ing
estion_labels\", \"tags\", \"enrichment_state\", \"log_type\", \"base_labels\", \"enrichment_labels\", \"structured_fields\"\\nline: 17
```

Lessons learned: Code reviews

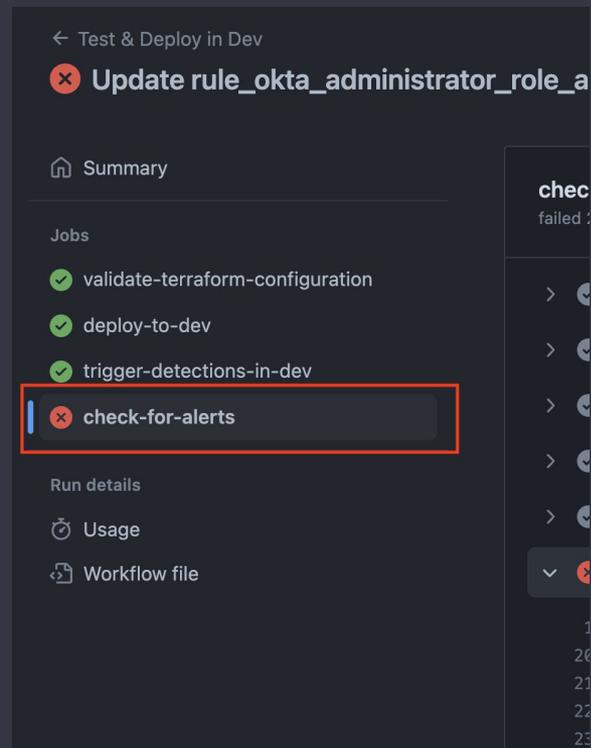
- Your rule may be criticized (its logic or the basis for the rule)
- Common for conflict to occur at this stage
- Authors: Assume positive intent - try to avoid getting defensive
- Reviewers:
 - Provide constructive feedback, explain your thought process, and make suggestions
 - Review in a timely manner
- Build a culture of trust and knowledge sharing
- Develop a rule style guide



“Don’t be the reason improvements
wither on the vine”

Testing rules: Don't skip this step!

- If you're not testing your detection rules on a regular basis, you're on shaky ground
- Can you say with confidence that your logging, detection, and alerting is working properly?
- Broken detections result in false negatives 😞
- Challenges & considerations
 - Time: It can take longer to develop a test than the rule itself!
 - Build vs. buy: Do we have the expertise to develop & automate tests?
 - Tech debt: What if you have hundreds of rules without tests? 😞



The problem with untested rules

- Environments drift
- Infrastructure and technologies come and go, software is updated
- Logging interruptions occur
- Vendors change their logging schemas
- Attack techniques no longer work (relevancy)
- Active detection rules that will never fire waste detection engine resources



Options for testing rules

- Run the rule against sample data
 - Better than having no tests at all
- Trigger the rule and validate alerts were generated
 - More comprehensive
 - Validates logging, detection, and alerting pipeline is working
 - Get started with free projects like [Atomic Red Team](#) and [Red Team Automation](#)
 - You can't test everything (and that's okay) e.g. anomaly detections



Triggering the rule

Run code in CI/CD pipeline job to carry out actions via Okta API and trigger detection rule

```
(venv) $ python -m detections_cli --run-all-tests
09-Feb-24 15:50:11 MST | INFO | <module> | detections_cli started
09-Feb-24 15:50:11 MST | INFO | <module> | Running all rule tests
09-Feb-24 15:50:11 MST | INFO | main | Executing test 'Assign Admin Role to Okta User' (Test ID: a17971ab-3980-4936-92e0-d65d9f448204)
09-Feb-24 15:50:11 MST | INFO | create_user | Attempting to create new Okta user [REDACTED] com
09-Feb-24 15:50:12 MST | INFO | create_user | Created new Okta user [REDACTED].com (ID: 00uf22f61vYv4xpgk5d7)
09-Feb-24 15:50:12 MST | INFO | assign_admin_role | Attempting to assign admin role 'READ ONLY ADMIN' to Okta user ID 00uf22f61vYv4xpgk5d7
09-Feb-24 15:50:12 MST | INFO | assign_admin_role | Assigned admin role 'READ_ONLY_ADMIN' to Okta user ID 00uf22f61vYv4xpgk5d7
09-Feb-24 15:50:12 MST | INFO | deactivate_user | Attempting to deactivate Okta user ID 00uf22f61vYv4xpgk5d7
09-Feb-24 15:50:12 MST | INFO | deactivate_user | Deactivated Okta user ID 00uf22f61vYv4xpgk5d7
09-Feb-24 15:50:12 MST | INFO | delete_user | Attempting to delete Okta user ID 00uf22f61vYv4xpgk5d7
09-Feb-24 15:50:13 MST | INFO | delete_user | Deleted Okta user ID 00uf22f61vYv4xpgk5d7
09-Feb-24 15:50:13 MST | INFO | main | Ending Test 'Assign Admin Role to Okta User' (Test ID: a17971ab-3980-4936-92e0-d65d9f448204)
```

Validating alerts

- Validate that alert was generated by detection rule
- Check for your test indicators in alerts
- Close alerts and any tickets/cases that were created
- CI/CD pipeline job success/failure

```
(venv) $ python -m detections_cli --validate-alerts
09-Feb-24 15:52:46 MST | INFO | <module> | detections_cli started
09-Feb-24 15:52:46 MST | INFO | <module> | Validating alerts created by tests
09-Feb-24 15:52:46 MST | INFO | validate_alerts | Checking for alerts created by test 'Assign Admin Role to Okta User'
09-Feb-24 15:52:46 MST | INFO | validate_alerts | Checking for alerts created for rule 'okta_administrator_role_assigned_to_non_admin_user_account' (Rule ID: ru_340b3f6d-916a-4365-86c0-b63cd5deb265)
09-Feb-24 15:52:46 MST | INFO | validate_alerts | Found 1 matching alerts for test 'Assign Admin Role to Okta User' and indicators for rule 'okta_administrator_role_assigned_to_non_admin_user_account' (Rule ID: ru_340b3f6d-916a-4365-86c0-b63cd5deb265)
```



Pipeline #1178536142 passed

Pipeline passed for `c6663939` on `new-rule-okta-admin...` just now

Deploying changes to the SIEM

Merged by  David French just now

Merge details

• Changes merged into main with [0f7a1152](#).

Changes are pushed to the SIEM after code is merged into the main branch

```
update_remote_rules | Checking if any rule updates are required
update_remote_rules | Local rule name okta_administrator_role_assigned_to_non_admin_user_account not found in remote rules
update_remote_rules | Local rule okta_administrator_role_assigned_to_non_admin_user_account has no rule id value. Creating a new rule
update_remote_rules | Created new rule okta_administrator_role_assigned_to_non_admin_user_account (ru_340b3f6d-916a-4365-86c0-b63cd5deb265)
update_remote_rule_state | Rule okta_administrator_role_assigned_to_non_admin_user_account (None) - Enabling rule
update_remote_rule_state | Rule okta_administrator_role_assigned_to_non_admin_user_account (None) - Enabling alerting for rule
update_remote_rules | Logging summary of rule changes...
update_remote_rules | Rules created: 1
update_remote_rules | created okta_administrator_role_assigned_to_non_admin_user_account (ru_340b3f6d-916a-4365-86c0-b63cd5deb265)
update_remote_rules | Rules new_version_created: 0
update_remote_rules | Rules enabled: 1
update_remote_rules | enabled okta_administrator_role_assigned_to_non_admin_user_account (ru_340b3f6d-916a-4365-86c0-b63cd5deb265)
update_remote_rules | Rules disabled: 0
update_remote_rules | Rules alerting_enabled: 1
update_remote_rules | alerting_enabled okta_administrator_role_assigned_to_non_admin_user_account (ru_340b3f6d-916a-4365-86c0-b63cd5deb265)
update_remote_rules | Rules alerting_disabled: 0
update_remote_rules | Rules archived: 0
update_remote_rules | Rules unarchived: 0
```

Syncing rule metadata to GitLab

- After changes are deployed to SIEM
- Pipeline job pulls latest rules from SIEM and commits updated metadata to rule config file in GitLab project

```
63 60 type: MULTI_EVENT
61 + okta_administrator_role_assigned_to_non_admin_user_account:
62 + alerting: true
63 + archive_time: null
64 + archived: false
65 + create_time: '2024-02-08T23:39:19.682863Z'
66 + enabled: true
67 + id: ru_340b3f6d-916a-4365-86c0-b63cd5deb265
68 + resource_name: projects/[REDACTED]/locations/us/instances/[REDACTED]
69 + revision_create_time: '2024-02-08T23:39:19.682863Z'
70 + revision_id: v_1707435559_682863000
71 + run_frequency: HOURLY
72 + type: SINGLE_EVENT
64 73 okta_new_api_token_created:
65 74 alerting: false
66 75 archive_time: null
... ..
```



latest rules pulled from chronicle api
David French authored 7 minutes ago

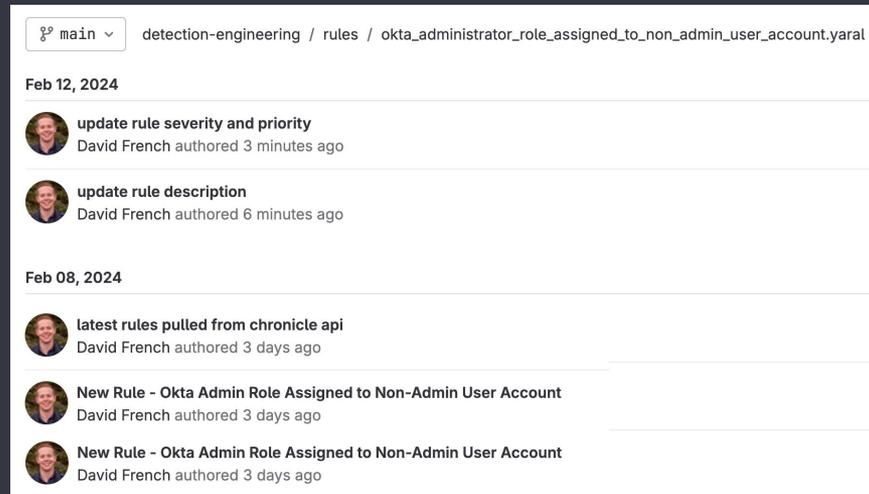
Modifying rules

- Detection Engineer creates a branch and pull request with proposed changes
- Tests succeed
- Peer review & approval obtained
- Changes are merged to the main branch
- Rule changes are deployed to SIEM



Auditing for rule changes

- Commit history in VCS makes it easy to review prior versions of a rule
- Context around changes is preserved in pull requests
- Can revert to a previous version if needed

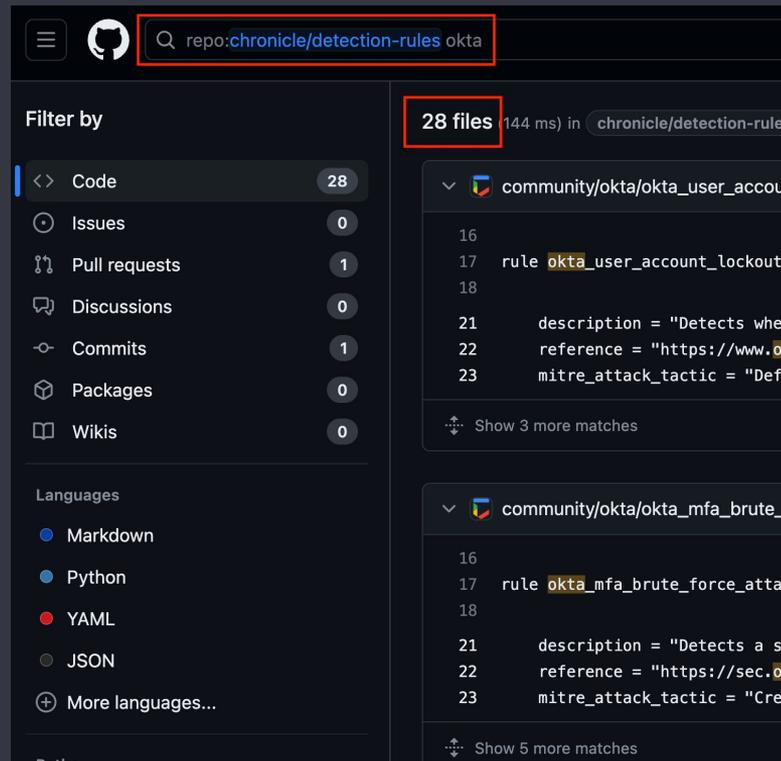


The screenshot shows a commit history for a file named `okta_administrator_role_assigned_to_non_admin_user_account.yaral` in a repository under the path `detection-engineering / rules /`. The history is organized by date:

- Feb 12, 2024**
 - update rule severity and priority** by David French, authored 3 minutes ago.
 - update rule description** by David French, authored 6 minutes ago.
- Feb 08, 2024**
 - latest rules pulled from chronicle api** by David French, authored 3 days ago.
 - New Rule - Okta Admin Role Assigned to Non-Admin User Account** by David French, authored 3 days ago.
 - New Rule - Okta Admin Role Assigned to Non-Admin User Account** by David French, authored 3 days ago.

Benefits of centralized detection management

- Auditors might ask for proof that you have a detection implemented (and that its tested)
 - For example, detections related to data loss prevention or SWIFT compliance
- Purple Teaming – Offensive team can analyze detections and look for ways to evade them
- Code repository is searchable
 - Can quickly check if you have a rule for an attack technique



Key takeaways

Which organizations can benefit from adopting DaC?

Yes

Large orgs with complex, dynamic IT environment and lots of normalized security data available

Auditing & change management needed for detective security controls

Security budget for Detection Engineers and required engineering expertise

Modern security tools (manage content via API)

No

Small orgs with simple, static IT environment

Limited security budget

Not much security data available for analysis

Small (or no dedicated) security team

Security tools with no support for integration

Partnering with an MSSP may be a good fit (they're likely using DaC to manage rules across multiple customers)

Advantages of adopting Detection-as-Code

- Increased collaboration around rule development and sharing in the community
 - A group of practitioners with unique insights working together will result in more accurate and effective rules
- More control over changes to detections
- Automated testing
 - Reduced risk of introducing false positives/negatives
 - Provides confidence that your logging, detection, and alerting is working
 - Helps identify issues quickly before misses occur



Useful resources

- Detection-as-Code
 - My blog series and example code for getting started: [1](#) and [2](#)
 - [Can We Have “Detection as Code”?](#) — Anton Chuvakin
 - [Automating Detection-as-Code](#) — John Tuckner
 - [Detection-as-Code: Why it works and where to start](#) — Kyle Bailey
 - [Detection as Code: Detection Development Using CI/CD](#) — Patrick Bareiß, Jose Hernandez
 - [Detection-as-Code panel](#) — Julie Agnes Sparks, Jackie Bow, Jessica Rozhin, Louis Barrett
- Detection Engineering
 - [Detection Engineering Weekly](#) — Zack Allen
 - [Practical Threat Detection Engineering](#) — Megan Roddie, Jason Deyalsingh, Gary J. Katz
- Free rules: [Google](#), [Elastic](#), [Splunk](#)

Thank you

David French

Staff Adoption Engineer, Google Cloud

[@threatpunter](#)

Q&A

Acknowledgement

Thanks to the following people for their contributions to the security community and/or providing me with valuable feedback:

Kyle Bailey, Patrick Bareiss, Adam Cole, James Black, Anton Chuvakin, Dan Dye, Serhat Gülbetekin, Jose Hernandez, Dave Herral, Justin Ibarra, Chris Long, Dan Lussier, Vishwanath Mantha, Christopher Martin, Anton Ovrutsky, Julie Agnes Sparks, John Stoner, John Tuckner, Wade Wells, Ross Wolf