The future is coded: Hands-on advancing detection engineering

Sumit Patel,
Stefan Avgoustakis

Detection Engineering

Turning Ideas into actionable detection

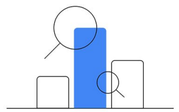
*“Detection engineering **transforms an idea** of how to detect a specific condition or activity **into a concrete description** of how to detect it.”*

– *Florian Roth*

Threat Detection Engineering Process

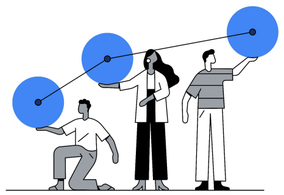
[Alex Teixeira]

Threat Hunting | CTI Reports | Red Team Exercise | Online Resources (blog, newsletter, socials, ..) | .....



Detection Value Analysis → Feasibility Analysis → Historical Analysis → Core Detection Artifacts → Engineering Handover

Eng. Backlog Management → Design & Prototype → Detection Models / MLOps → Tools Integration → Testing → Documentation Playbooks



DevOps Backlog Management → Deployment → Validation → Release Management → SecOps Workshop and Handover

Metrics Analysis → Triage Investigation & Playbook Updates → Quality Control → Feature Requests & Bug Fix → Engineering Feedback

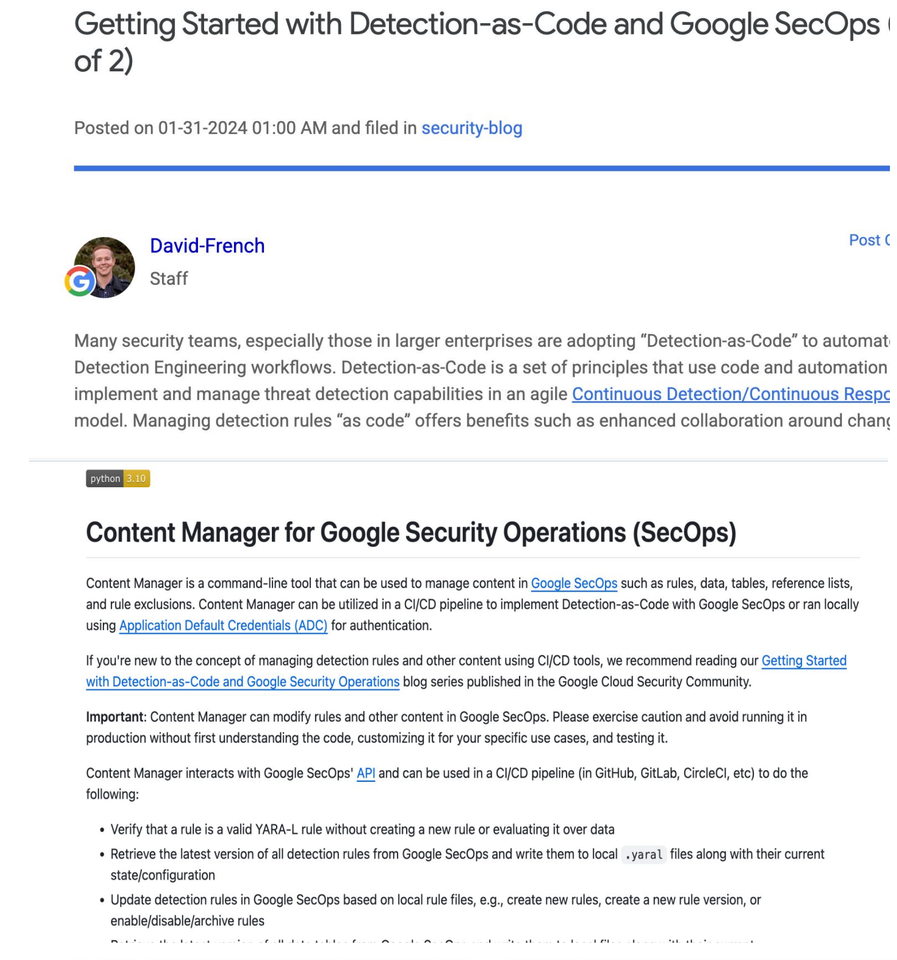
Detection Engineering

Challenges

- Difficulty Tracking Changes and Ensuring Consistency
- Lack of Robust Testing and Quality Assurance
- Difficulty Scaling Detection Efforts
- Lack of Collaboration and Knowledge Sharing
- Difficulty Managing Detection Content Lifecycle
- Difficulty in Applying Threat Intelligence to Detections
- Lack of a Repeatable and Consistent Detection Development Process
- Challenges in Maintaining Detection Rule Accuracy Over Time

What is Detection-as-Code (DaC)?

- A set of principles that use code & automation to manage detection content
- Leverages software development practices
- Treats detection content as code artifacts
- Some modern SecOps teams want everything “as-code”



Getting Started with Detection-as-Code and Google SecOps (of 2)

Posted on 01-31-2024 01:00 AM and filed in security-blog

David-French
Staff

Post (

Many security teams, especially those in larger enterprises are adopting “Detection-as-Code” to automat Detection Engineering workflows. Detection-as-Code is a set of principles that use code and automation implement and manage threat detection capabilities in an agile Continuous Detection/Continuous Respo model. Managing detection rules “as code” offers benefits such as enhanced collaboration around chan

python 3.10

Content Manager for Google Security Operations (SecOps)

Content Manager is a command-line tool that can be used to manage content in Google SecOps such as rules, data, tables, reference lists, and rule exclusions. Content Manager can be utilized in a CI/CD pipeline to implement Detection-as-Code with Google SecOps or ran locally using Application Default Credentials (ADC) for authentication.

If you're new to the concept of managing detection rules and other content using CI/CD tools, we recommend reading our Getting Started with Detection-as-Code and Google Security Operations blog series published in the Google Cloud Security Community.

Important: Content Manager can modify rules and other content in Google SecOps. Please exercise caution and avoid running it in production without first understanding the code, customizing it for your specific use cases, and testing it.

Content Manager interacts with Google SecOps' API and can be used in a CI/CD pipeline (in GitHub, GitLab, CircleCI, etc) to do the following:

- Verify that a rule is a valid YARA-L rule without creating a new rule or evaluating it over data
- Retrieve the latest version of all detection rules from Google SecOps and write them to local `.yaral` files along with their current state/configuration
- Update detection rules in Google SecOps based on local rule files, e.g., create new rules, create a new rule version, or enable/disable/archive rules

https://www.googlecloudcommunity.com/gc/Community-Blog/Getting-Started-with-Detection-as-Code-and-Google-SecOps-Part-1/ba-p/702154

https://github.com/chronicle/detection-rules/tree/main/tools/content_manager

Detection As Code applied

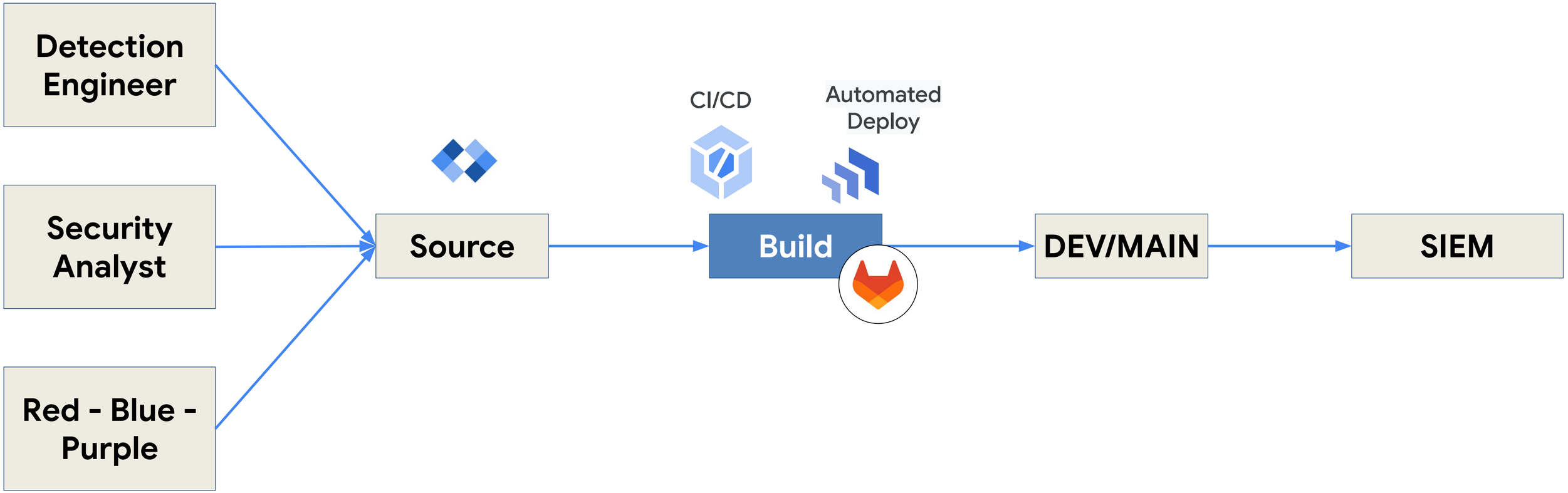
Difficulty Tracking Changes and Ensuring Consistency	Detection rules as code, version control systems & ensuring a single source of truth
Lack of Robust Testing and Quality Assurance	Automated testing procedures via CI/CD pipeline, detection rules are validated before deployment
Difficulty Scaling Detection Efforts	Automation of tasks - simplifying maintenance and scaling efforts
Lack of Collaboration and Knowledge Sharing	Repositories allow for better collaboration, peer review, and sharing of detection logic

Detection As Code applied

Difficulty Managing Detection Content Lifecycle	Versioning, tracking, and automated deployment - better management of the entire detection lifecycle
Difficulty in Applying Threat Intelligence to Detections	Automates the process of incorporating threat intelligence into your detection rules
Lack of a Repeatable and Consistent Detection Development Process	Promotes shift towards sharing CTI in machine-readable formats, simplifying integration into detection logic
Challenges in Maintaining Detection Rule Accuracy Over Time	Continuous testing and validation of detection rules within a CI/CD pipeline - identify and address any issues arising from changes in the environment or attacker behavior

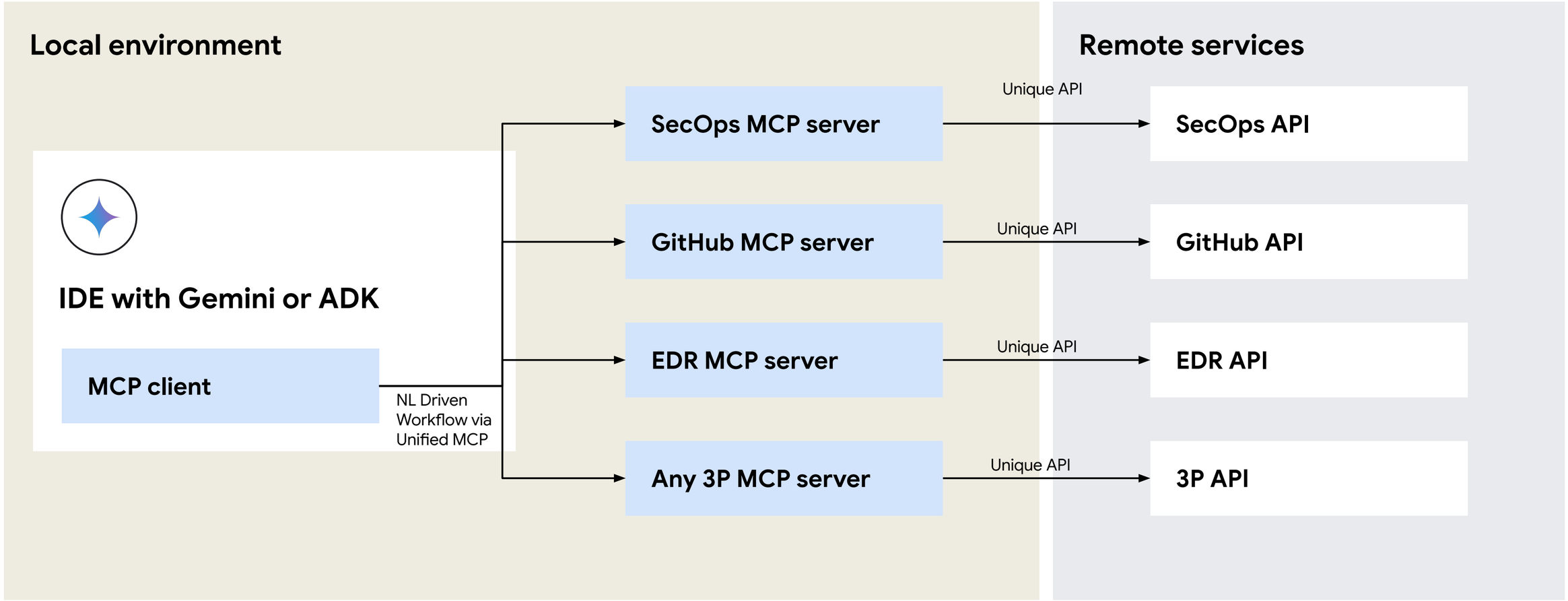
CI/CD

Continuous Integration Continuous Deployment



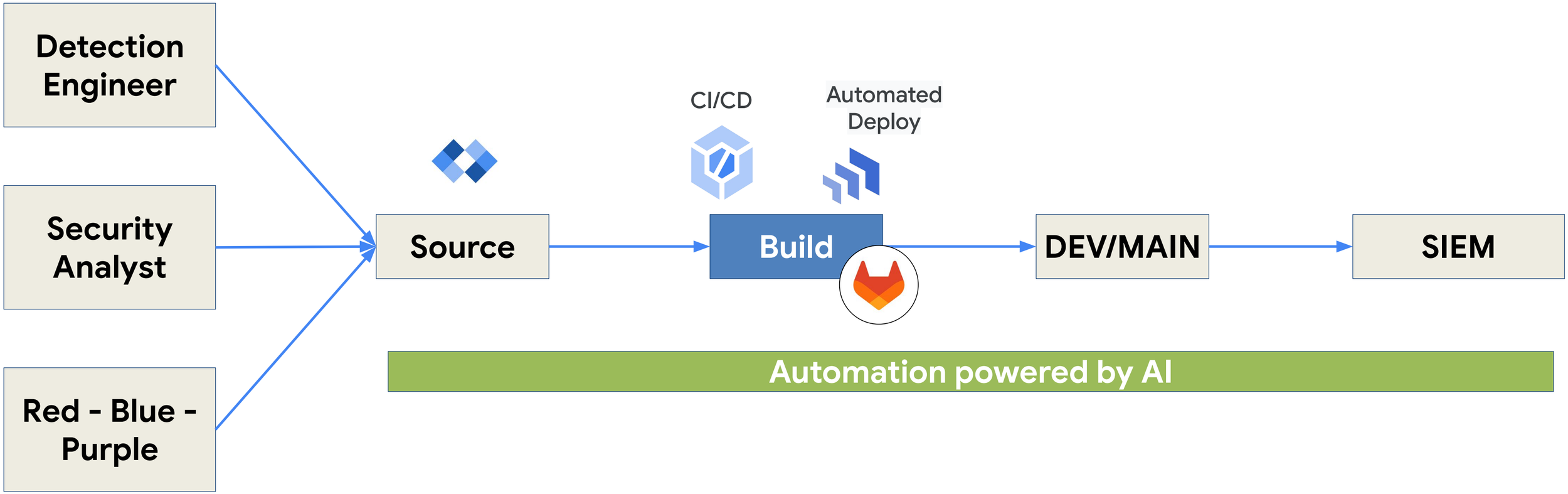
AI-Ready Security enabled by MCP

Build your own security workflows with open sourced MCP (Model Context Protocol) Server



CI/CD

Continuous Integration Continuous Deployment

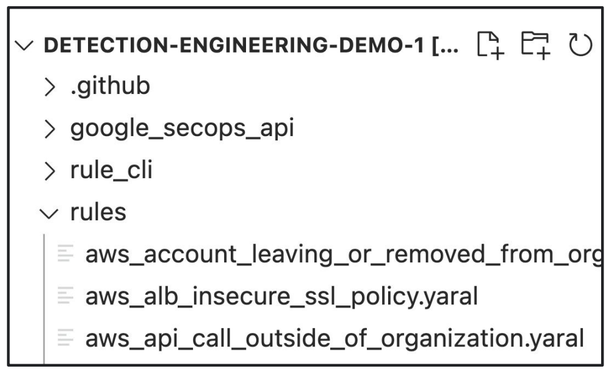


Workshop Goal: What we will build!

Visual Studio Code

GitHub

GitHub Actions



CI/CD Pipeline Jobs

Run Tests

Get rules

Update rules

Read, create, update, and verify rules via APIs

SecOps

Use of natural language

MCP Server

SecOps API

Workshop instruction guide

Follow guide here: **http://bit.ly/47b7Qym**

Contact

Sumit Patel, Stefan Avgoustakis
LinkedIn:
linkedin.com/in/sumit-p-4870b41b
linkedin.com/in/stefanavgoustakis