Detection As Code

SEC407

Write detections once in Sigma, test them against committed fixtures, run them through CI that blocks a bad rule from merging, and deploy them to Sentinel, Splunk, or Elastic without touching a console. You finish the course with a working detection-as-code pipeline in your own GitHub: a structured rule repository, fixture-based tests, GitHub Actions CI, automated SIEM deployment, and an ATT&CK coverage layer generated from the repo.

Modules	20
Lessons	166
Phases	5
Free preview	Course Orientation (5 lessons)

The method

Detection as code is version control, testing and deployment applied to detection content. This course runs the same loop for every rule it ships.

1. Author once, in a source format. Sigma is the source of truth and the backend query is a build artifact. Editing the deployed query is how a repository and a SIEM drift apart.

2. Put it under review like code. A branch, a diff, a second pair of eyes. A rule change nobody reviewed is a production change nobody reviewed.

3. Test in CI, not after deployment. Syntax, conversion and logic checks that run on every commit. The pipeline is what makes the review meaningful rather than ceremonial.

4. Deploy from the pipeline. If a rule can reach production by hand, the repository is a record of intentions rather than of what is running.

5. Measure coverage from the repository. Coverage computed from what is deployed, with a stated denominator, rather than from a spreadsheet somebody maintains separately.

The course closes by shipping a detection end to end through the pipeline you built.

What this course assumes

No minimum experience and no prerequisite course. Git, YAML, CI concepts and the conversion model are introduced from nothing, and the course assumes you are a detection engineer rather than a developer.

What makes it go faster: a Git host you can push to and a SIEM to deploy into. Neither is required; the pipeline is built and run locally throughout.

What this course does not cover: writing detection logic in depth, which is its own course, and SIEM administration. This is the engineering practice around the rules.

Full syllabus

Phase 1: Foundations

Course Orientation FREE PREVIEW

What Detection As Code teaches: write detections in Sigma, test them against committed fixtures, gate them on CI, and deploy them to any SIEM through a pipeline you own.

1. DC0.1 What Detection As Code Is
2. DC0.2 Sigma, and a Rule That Converts
3. DC0.3 The Pipeline, and What You Build
4. DC0.4 Lab Environment and Tooling
5. DC0.5 Course Structure and How to Study

Sigma as the Source Format

Author Sigma detection rules from scratch: the rule anatomy, the logsource taxonomy that targets the right events, the field maps and lists that express matching, the modifier system, the condition language, and correlation rules.

1. DC1.1 Anatomy of a Rule
2. DC1.2 Logsource: Targeting the Right Events
3. DC1.3 The Detection Block
4. DC1.4 Modifiers
5. DC1.5 The Condition Expression
6. DC1.6 Correlation Rules
7. DC1.7 Regex and Special-Character Handling
8. DC1.8 Placeholders and Value Lists
9. DC1.9 Rule Collections
10. DC1.10 Write and Validate a Rule End to End
11. Module Summary
12. Check My Knowledge

Phase 2: The Detection Repository

The Detection Repository

Turn a pile of rule files into a detection library: a repository layout that encodes tactic, the rule-test-documentation contract, a metadata standard enforced repo-wide, and naming and identity that hold at scale.

1. DC2.1 The Repository as a System
2. DC2.2 The Rule-Test-Documentation Contract
3. DC2.3 The Metadata Standard
4. DC2.4 Naming and Identity at Scale
5. DC2.5 Templates and the Cost of Compliance
6. DC2.6 Documenting a Detection
7. DC2.7 Rule Relationships and the related Field
8. DC2.5 Organizing at Scale and the Build-Time Payoff
9. DC2.9 The Detection Repository as a Sensitive Asset
10. DC2.6 Lay Out the NE Detection Repository
11. Module Summary
12. Check My Knowledge

Git for Detection Content

Manage detection content as code: a branching model that keeps main always deployable, commit hygiene that makes the history an audit trail, the pull request as the review surface for a detection, why detection review is its own discipline, and CODEOWNERS routing approvals by tactic.

1. DC3.1 Git as the Detection Control Plane
2. DC3.2 A Branching Model for Detections
3. DC3.3 Commit Hygiene and History as Audit Trail
4. DC3.4 Reading the History: log, blame, and bisect
5. DC3.5 Merge Conflicts as Detection Decisions
6. DC3.6 The Pull Request as Review Surface
7. DC3.7 Why Detection Review Is Its Own Discipline
8. DC3.8 CODEOWNERS and Routing
9. DC3.9 Branch Protection
10. DC3.10 Build: Take a Change Through the Whole Workflow
11. Module Summary
12. Check My Knowledge

Backend Conversion

Turn portable Sigma source into the query your SIEM actually runs.

1. DC4.1 Why Portable Source Needs Conversion
2. DC4.2 The Conversion Toolchain: sigma-cli and pySigma
3. DC4.3 One Rule, Three Backends
4. DC4.4 The Processing Pipeline: Field and Table Mapping
5. DC4.5 Pipelines as Code
6. DC4.6 Where Conversion Breaks
7. DC4.7 Backend Differences That Change Detection Meaning
8. DC4.8 Native Rule Formats Versus Plain Queries
9. DC4.9 Converting the Whole Repository
10. DC4.10 Build: Convert the NE Rule Set to the Team's Backends
11. DC4.11 Performance and Cost-Aware Conversion
12. Module Summary
13. Check My Knowledge

Phase 3: The Pipeline

Testing Detections

A rule that converts cleanly can still be dead on arrival.

1. DC5.1 Why Conversion Isn't Enough
2. DC5.2 Fixture Anatomy
3. DC5.3 Writing True-Positive Fixtures
4. DC5.4 Writing Benign-Baseline Fixtures
5. DC5.5 Building the Test Harness
6. DC5.6 Atomic Red Team as a Fixture Source
7. DC5.7 Running the Suite, Reading the Output
8. DC5.8 Fixture Quality
9. DC5.9 Fixtures Committed with the Rule
10. DC5.10 Build a Tested Rule End-to-End
11. Module Summary
12. Check My Knowledge

CI for Detection Content

A suite that only runs when someone remembers to run it protects nothing.

1. DC6.1 Why CI Rather Than a Local Script
2. DC6.2 GitHub Actions Fundamentals
3. DC6.3 The Lint Stage
4. DC6.4 The Convert Stage
5. DC6.5 The Test Stage
6. DC6.6 Block on Failure
7. DC6.7 Caching and Performance
8. DC6.8 Reading a Failed CI Run
9. DC6.9 What a Reviewer Sees
10. DC6.10 Build the Workflow End to End
11. Module Summary
12. Check My Knowledge

Deployment

A merged rule that a person still pastes into a portal is not deployed.

1. DC7.1 The Last Manual Step
2. DC7.2 What a Deployed Rule Actually Is
3. DC7.3 The Sentinel Deployment Artifact
4. DC7.4 The Splunk Deployment Artifact
5. DC7.5 The Secret You Must Not Commit
6. DC7.6 OIDC: The Credential That Does Not Exist
7. DC7.7 Least-Privilege Scoping
8. DC7.8 The Deploy Job
9. DC7.9 Environment Promotion
10. DC7.10 Rollback, Drift, and the Complete Workflow
11. Module Summary
12. Check My Knowledge

Phase 4: Operating the Program

Coverage as Code

A coverage map maintained by hand is out of date the moment a rule merges.

1. DC8.1 Coverage as a Computed Property
2. DC8.2 The Technique Tag Is the Datum
3. DC8.3 Reading the Rule Set Into an Inventory
4. DC8.4 Generating a Navigator Layer
5. DC8.5 Scoring Coverage Honestly
6. DC8.6 Gap Analysis From the Inventory
7. DC8.7 From a Gap to a Rule Request
8. DC8.8 Build: The Coverage Layer in CI
9. Module Summary
10. Check My Knowledge

Tuning and the Feedback Loop

A false positive is a signal, not a failure.

1. DC9.1 A False Positive Is a Signal
2. DC9.2 Tuning as a Reviewed Change
3. DC9.3 Refine or Suppress
4. DC9.4 Suppression and Allowlists as Code
5. DC9.5 Versioning a Tuning Change
6. DC9.6 Where Sigma Stops
7. DC9.7 The Native-Detection Escape Hatch
8. DC9.8 Build: Tune a Rule and Take One Native
9. Module Summary
10. Check My Knowledge

Rule Health and Operating Model

A rule set nobody measures rots silently.

1. DC10.1 A Rule Is Not Done When It Ships
2. DC10.2 Rule Health from the Repo
3. DC10.3 Rule Health from the SIEM
4. DC10.4 The Rule-Health Report
5. DC10.5 Coverage Drift Over Time
6. DC10.6 The Maturity Gradient
7. DC10.7 RACI and the Operating Model
8. DC10.8 Cross-Team Flow
9. Module Summary
10. Check Your Knowledge

Phase 5: Capstone

Capstone: Ship a Detection End to End

The whole lifecycle in one assessment. A new threat lands with no coverage, and you carry it through every stage you built: Sigma rule, fixtures, tests, pull request, CI, deployment, coverage layer, and rule-health entry. One detection, shipped end to end.

1. DC11.1 The Threat and the Rule
2. DC11.2 Fixtures and Tests
3. DC11.3 The Pull Request and CI
4. DC11.4 Deployment
5. DC11.5 Coverage and Health
6. DC11.6 What You Built
7. Capstone Summary
8. Check Your Knowledge

Phase 0: Course Resources

Cheatsheets

The Sigma syntax, the Git operations, the pipeline stages and the failure modes, one sheet per part of the system.

1. Sigma Rule Syntax
2. The Repository and Its Metadata
3. Backend Conversion
4. Testing Detections
5. CI and Deployment
6. Coverage and Tuning
7. Git for Detections
8. Rule Health and the Operating Model
9. The Toolchain
10. The Silent Failures
11. Starting From Nothing
12. One Rule, End to End

Cookbooks

Ordered procedures for building the pipeline, each stopping where the value stops rather than where the method ends.

1. Standing Up the Repository
2. Making the Rules Testable
3. Automating Deployment
4. Measuring Coverage Honestly
5. Tuning Without Losing the Detection
6. Migrating to a New SIEM
7. Answering an Auditor

Lab Setup

A working repository, pipeline and deployment path built from nothing, with every silent failure produced deliberately.

1. Building the Repository
2. Building the Pipeline
3. Producing the Silent Failures
4. Deploying and Drifting
5. Verify, and What the Lab Cannot Teach

Walkthroughs

Detection engineering reasoned end to end, including the cases where the pipeline was green and the answer was still no.

1. The Green Pipeline Over Dead Rules
2. The Rule That Could Not Be Written
3. The Coverage Figure That Was Wrong
4. The Rule Nobody Should Have Tuned
5. The Detection That Was Not There
6. The Migration Estimate

Playbooks

What to do when the pipeline fails, a rule turns out dead, drift appears, or somebody asks for a rule you cannot write.

1. The Pipeline Is Failing
2. A Rule Has Never Fired
3. Production Does Not Match the Repository
4. A Rule Is Producing Too Much
5. Somebody Wants a Rule You Cannot Write
6. The Deployment Broke Something
7. Somebody Asks How You Know It Works

Playground
Every practice surface available for this course, what each one gives you, and where the gaps are.

References & Further Reading
Sigma, pySigma, ATT&CK, Atomic Red Team, DeTT&CT, and the backend and CI documentation used throughout the Detection As Code course, organized by category.

Course Completion

Course Exam
Detection As Code end-of-course exam: a scenario-based assessment testing whether you can carry a new detection through the full pipeline, from a threat you have not seen to a deployed, measured rule, using the method built across all twelve modules.

1. Course Completion. Detection As Code

Generated 2026-09-01 from the published course. The current syllabus is always at ridgelinecyber.com/training/courses/detection-as-code/