

YARA Rule Writing for DFIR

Write YARA rules you can defend: what each modifier costs, the denominator behind every false-positive figure, patterns chosen by what it costs an author to change them, and the two-sided control that separates a clean estate from a broken rule.

Modules	16
Lessons	78
Phases	4
Free preview	Course Orientation (7 lessons)

The method

A YARA rule is a claim about what a file contains. This course runs the same loop for every rule it writes, because a rule that matches your sample and nothing else is a hash with extra steps.

1. Decide what the rule is claiming. This family, this packer, this technique, or this specific sample. The four claims need different rules and have different lifespans.
2. Choose strings that survive a recompile. Anything the author would change between builds is a bad string. What the code has to do to work is a good one.
3. Bound it with structure. File size, format, section characteristics. A condition that reaches the strings only for plausible candidates is faster and matches less by accident.
4. Test against a corpus, not a sample. Run it over goodware as well as malware. A rule nobody ran against a clean corpus is a rule whose false positive rate is unknown rather than low.
5. Measure what it costs to run. Scanning speed is a deployment constraint. A rule too slow to run at scale does not protect anything.

What this course assumes

No minimum experience and no prerequisite course. File formats, hex, regular expressions and the PE structure are explained where they are first needed.

What makes it go faster: comfort with a hex editor and some exposure to file formats. Neither is required. Every rule in the course is built against samples the course provides.

What this course does not cover: reverse engineering, malware analysis as a discipline, and unpacking. Those decide what your rule should look for; this course is about writing the rule once you know.

Full syllabus

Course Orientation

Course Orientation [FREE PREVIEW](#)

What SEC409 teaches: writing YARA rules that still fire next month, on files you have not seen, without burying the estate in false positives.

1. 0.1 Your First Scan, and What It Does Not Tell You
2. 0.2 The Specimens You Will Build
3. 0.3 The Corpus Your Rules Must Not Match
4. 0.4 Where Your Rules Will Run
5. 0.5 What a Match Is Worth
6. 0.6 What the Seven Modules Build
7. Module Summary

Phase 1: Writing and Running Rules

The Rule as a Claim

The YARA language: text strings and their modifiers, hex patterns with wildcards and jumps, regular expressions and their cost, the condition logic that turns candidate patterns into an assertion, and metadata that survives being read by somebody else.

1. 1.1 Anatomy of a Rule
2. 1.2 Text Strings and Modifiers
3. 1.3 Hex Patterns and Wildcards
4. 1.4 Regular Expressions
5. 1.5 The Condition Is the Rule
6. 1.6 Metadata and the Writing Loop
7. Module Summary
8. Check My Knowledge

Running Scans

Operating yr scan properly: what a target is and why directory scanning is not recursive by default, reading output at scale, threads and timeouts and size bounds, the controls that prove a negative, rule sets and namespaces, and the hunt loop.

1. 2.1 Targets and Recursion
2. 2.2 Reading What Comes Back
3. 2.3 Scanning at Scale
4. 2.4 Controls, and Proving a Negative
5. 2.5 Rule Sets and Namespaces
6. 2.6 The Hunt Loop
7. Module Summary
8. Check My Knowledge

Phase 2: Choosing What to Match

Choosing What to Match

Choosing patterns by what it would cost the author to change them: the durability question, both measurement columns, extraction as a workflow, the false friends your eye picks first, and the build-environment trap.

1. 3.1 The Durability Question
2. 3.2 The Durability Ladder
3. 3.3 Extracting Candidates
4. 3.4 Patterns That Look Good and Are Not
5. 3.5 The Build-Environment Trap
6. 3.6 Choosing the Set
7. Module Summary
8. Check My Knowledge

Modules and Structure

Reading a file with `yr dump` before writing any condition: `pe` fields and the bitmask trap, imports and `imphash`, entropy baselines, signatures and the Rich header, and what structural rules actually cost.

1. 4.1 Reading a File With `yr dump`
2. 4.2 Writing Conditions on `pe` Fields
3. 4.3 Imports and `Imphash`
4. 4.4 Sections and Entropy
5. 4.5 Signatures and Resources
6. 4.6 Structural Rules in Practice
7. Module Summary
8. Check My Knowledge

Non-Binary Formats

Formats where the binary model does not apply: why a correct rule returns nothing on a document or an encoded script, what each container keeps readable, webshell precision, ELF differences, and the three questions that transfer.

1. 5.1 The File Is Not a Binary
2. 5.2 Office Documents Are Archives
3. 5.3 Scripts and Layered Encoding
4. 5.4 Webshells
5. 5.5 ELF and the Linux Estate
6. 5.6 Choosing the Approach
7. Module Summary
8. Check My Knowledge

Phase 3: Maintaining and Deploying

Testing, Tuning and Performance

What happens after a rule is written: finding which rule in a set dominates a scan, the tuning loop, baselines and regression, diagnosing a rule that has gone quiet, reviewing rules you did not write, and the maintenance cadence.

1. 6.1 Finding the Expensive Rule
2. 6.2 The Tuning Loop
3. 6.3 Regression and Baselines
4. 6.4 When a Rule Stops Firing
5. 6.5 Reviewing Rules You Did Not Write
6. 6.6 The Maintenance Cadence
7. Module Summary
8. Check My Knowledge

Deploying and Hunting

What changes when a rule leaves your machine: the two engines and where they disagree, surfaces and thresholds, deploying a set, running a hunt, sharing rules honestly, and where the course leaves you.

1. 7.1 The Two Engines
2. 7.2 Surfaces and Thresholds
3. 7.3 Deploying a Set
4. 7.4 Hunting With Rules
5. 7.5 Sharing and Attribution
6. 7.6 Where This Leaves You
7. Module Summary
8. Check My Knowledge

Phase 0: Course Resources

Cheatsheets

The lookup layer: five sheets covering patterns and modifiers, scanning and output, choosing patterns, structure and modules, and non-binary formats, each entry stating what to read and what it does not prove.

1. Patterns and Modifiers
2. Scanning, Output and Controls
3. Choosing and Measuring Patterns
4. Structure and Modules
5. Non-Binary Formats and Deployment

Cookbooks

Three runbooks: writing a rule from a sample end to end, running a hunt whose negative result means something, and reviewing a rule set you did not write.

1. Writing a Rule From a Sample
2. Running a Hunt
3. Reviewing a Set You Did Not Write

Lab Setup

Building the environment every measurement in this course depends on: the yr binary, two specimen families, a clean corpus, a PE corpus, the non-binary specimens, and a controls directory, with a verification script that fails loudly.

1. Building the Lab

Walkthroughs

Two cases worked end to end including the wrong turns: a packed PE where the obvious signal points the wrong way, and a document-and-script chain where a correct rule matches nothing.

1. A Packed PE
2. A Document and a Script

Playbooks

Three procedures for the moments something fires: a rule hit and you have to decide what it establishes, a rule has gone quiet and you cannot tell why, and a set stopped producing hits after an update.

1. A Rule Hit
2. A Rule Has Gone Quiet
3. A Set Stopped Producing Hits

Operational Reference

The whole course condensed to one operating page: the command for each stage, the figure it produced, and the one judgment that stage turns on.

References and Further Reading

The tools, the documentation, the frameworks, and what to check first when something in the course behaves differently from the page.

Course Completion

Course Exam

YARA Rule Writing end-of-course exam: a hunt that returns zero hits including on the host holding the sample, testing whether you can separate did not match from was never evaluated and diagnose three independent failures under incident pressure.

1. Course Completion. YARA Rule Writing

Generated 2026-09-01 from the published course. The current syllabus is always at ridgelinecyber.com/training/courses/yara-rule-writing/