

# Forward-Looking Statements



This presentation may contain forward-looking statements regarding future events, plans or the expected financial performance of our company, including our expectations regarding our products, technology, strategy, customers, markets, acquisitions and investments. These statements reflect management's current expectations, estimates and assumptions based on the information currently available to us. These forward-looking statements are not guarantees of future performance and involve significant risks, uncertainties and other factors that may cause our actual results, performance or achievements to be materially different from results, performance or achievements expressed or implied by the forward-looking statements contained in this presentation.

For additional information about factors that could cause actual results to differ materially from those described in the forward-looking statements made in this presentation, please refer to our periodic reports and other filings with the SEC, including the risk factors identified in our most recent quarterly reports on Form 10-Q and annual reports on Form 10-K, copies of which may be obtained by visiting the Splunk Investor Relations website at [www.investors.splunk.com](http://www.investors.splunk.com) or the SEC's website at [www.sec.gov](http://www.sec.gov). The forward-looking statements made in this presentation are made as of the time and date of this presentation. If reviewed after the initial presentation, even if made available by us, on our website or otherwise, it may not contain current or accurate information. We disclaim any obligation to update or revise any forward-looking statement based on new information, future events or otherwise, except as required by applicable law.

In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. We undertake no obligation either to develop the features or functionalities described, in beta or in preview (used interchangeably), or to include any such feature or functionality in a future release.

Splunk, Splunk> and Turn Data Into Doing are trademarks and registered trademarks of Splunk Inc. in the United States and other countries. All other brand names, product names or trademarks belong to their respective owners. © 2023 Splunk Inc. All rights reserved.

# Deploying Detection as Code at Scale

SEC1847A

**Nate Zastrow**

Senior Detection Engineer | Northwestern Mutual





# Nate Zastrow

Senior Detection Engineer  
Northwestern Mutual

# What is Detection as Code?



# What is Detection as Code?

Apply the tools and best practices from software development to detection content development.





# What is Detection as Code?

Apply the tools and best practices from software development to detection content development.

**Four Principles:**



# What is Detection as Code?

Apply the tools and best practices from software development to detection content development.

## Four Principles:

- Detection Language



# What is Detection as Code?

Apply the tools and best practices from software development to detection content development.

## Four Principles:

- Detection Language
- Version Control System





# What is Detection as Code?

Apply the tools and best practices from software development to detection content development.

## Four Principles:

- Detection Language
- Version Control System
- Automated Workflows



# What is Detection as Code?

Apply the tools and best practices from software development to detection content development.

## Four Principles:

- Detection Language
- Version Control System
- Automated Workflows
- Test-driven Development



# Why implement DaC?

# Why implement DaC?

- Detections are already code

# Why implement DaC?

- Detections are already code
- Ensure thorough detection validation
- Built-in change control
- Increase visibility of changes
- Improve documentation
- Increase development velocity
  - Easier discoverability
  - More confidence with unit tests and required approvals
  - Less time fixing broken alerts
  - Enable broad find/replace-type changes

# The Challenges We Faced





# The Challenges We Faced

- **700+** correlation searches
- **600+** properties of a correlation search configurable via API
- **20+** content creators across different teams



# The Challenges We Faced

- **700+** correlation searches
- **600+** properties of a correlation search configurable via API
- **20+** content creators across different teams
- Upskilling team members in new tools for DaC



# The Challenges We Faced

- **700+** correlation searches
- **600+** properties of a correlation search configurable via API
- **20+** content creators across different teams
- Upskilling team members in new tools for DaC
- Already have an extensive in-house detection validation process to integrate with



# The Challenges We Faced

- **700+** correlation searches
- **600+** properties of a correlation search configurable via API
- **20+** content creators across different teams
- Upskilling team members in new tools for DaC
- Already have an extensive in-house detection validation process to integrate with
- Mid-project Splunk® Cloud migration



# Tool Selection

# Tool Selection

## YAML - Detection Language

- Human and machine readable
- Syntax is relatively easy to learn
- Supports comments
- Ubiquitous

## Git - Version Control / Automation

- Lots of CI/CD features baked into enterprise versions
- Ubiquitous

## Docker - Automation

- Already integrated into our git environment
- Very easy to build and iterate on as we mature our use case
- Ubiquitous

## Python - Automation

- Splunk maintains a Python SDK so we don't need to write our own middleware
- Ubiquitous



# How It Started



# How It Started

- First iteration was a simple “pull” sync from Splunk® into git:
  - API call to get current state of all correlation searches
  - Convert results from JSON to YAML
  - Compare with YAML files already in repo
  - Write any changes out
  - Submit merge request for that day’s changes



# How It Started

- First iteration was a simple “pull” sync from Splunk® into git:
  - API call to get current state of all correlation searches
  - Convert results from JSON to YAML
  - Compare with YAML files already in repo
  - Write any changes out
  - Submit merge request for that day’s changes
- This allowed us to:



# How It Started

- First iteration was a simple “pull” sync from Splunk® into git:
  - API call to get current state of all correlation searches
  - Convert results from JSON to YAML
  - Compare with YAML files already in repo
  - Write any changes out
  - Submit merge request for that day’s changes
- This allowed us to:
  - Build and test the scaffolding we needed for the “push” end state



# How It Started

- First iteration was a simple “pull” sync from Splunk® into git:
  - API call to get current state of all correlation searches
  - Convert results from JSON to YAML
  - Compare with YAML files already in repo
  - Write any changes out
  - Submit merge request for that day’s changes
- This allowed us to:
  - Build and test the scaffolding we needed for the “push” end state
  - Narrow down which of the hundreds of properties we actually care about



# How It Started

- First iteration was a simple “pull” sync from Splunk® into git:
  - API call to get current state of all correlation searches
  - Convert results from JSON to YAML
  - Compare with YAML files already in repo
  - Write any changes out
  - Submit merge request for that day’s changes
- This allowed us to:
  - Build and test the scaffolding we needed for the “push” end state
  - Narrow down which of the hundreds of properties we actually care about
  - Provide basic change tracking through daily merge requests





# How It Started

- First iteration was a simple “pull” sync from Splunk® into git:
  - API call to get current state of all correlation searches
  - Convert results from JSON to YAML
  - Compare with YAML files already in repo
  - Write any changes out
  - Submit merge request for that day’s changes
- This allowed us to:
  - Build and test the scaffolding we needed for the “push” end state
  - Narrow down which of the hundreds of properties we actually care about
  - Provide basic change tracking through daily merge requests
  - Prove the concept to leadership



# How It's Going



# How It's Going

- All production detections managed via GitOps
- Mandatory review required for all changes to prod
- MR template checklist ensures consistent and thorough review of changes



# How It's Going

- All production detections managed via GitOps
- Mandatory review required for all changes to prod
- MR template checklist ensures consistent and thorough review of changes
- Full CI/CD pipeline with:
  - Linting
  - SPL safety checks
  - Unit tests
  - Deploy to Splunk® ES via Splunk® SDK
  - Automated documentation via git pages



# How It's Going

- All production detections managed via GitOps
- Mandatory review required for all changes to prod
- MR template checklist ensures consistent and thorough review of changes
- Full CI/CD pipeline with:
  - Linting
  - SPL safety checks
  - Unit tests
  - Deploy to Splunk® ES via Splunk® SDK
  - Automated documentation via git pages

**100+ merges per month / 1,000+ total**



**“I love peer review.”**  
**- Anonymous Engineer**





# Demo

# Where We're Going From Here

# Where We're Going From Here

- Fully integrate CI/CD with our existing detection validation process
- Automate more of the MR checklist (SPL linting, improved dynamic analysis)
- Bring in content creators outside of the security team
- Expand the scope to other Splunk® content (macros, lookups, dashboards)
- Bring other tools under the detection as code umbrella (endpoint sensor logic, suricata rules, etc.)

# What We Learned Along the Way

**Thorns**

---

**Roses**

---

# What We Learned Along the Way

## Thorns

---

- Upskilling is a team-wide lift
- It's a challenge to keep tooling simple and onboarding easy
- Schema updates can be a chore
- Running “push” and “pull” syncs simultaneously can get interesting

## Roses

---

# What We Learned Along the Way

## Thorns

---

- Upskilling is a team-wide lift
- It's a challenge to keep tooling simple and onboarding easy
- Schema updates can be a chore
- Running “push” and “pull” syncs simultaneously can get interesting

## Roses

---

- Enforced and automated documentation
- Enforced alert output standards
- Custom tagging
- Automated testing
- Catching all sorts of issues before they hit prod
- Team-wide visibility of changes
- Large scale updates

# Parting Advice

# Parting Advice

- Start small



# Parting Advice

- Start small
- Get your content creators on board up front

# Parting Advice

- Start small
- Get your content creators on board up front
- Bring in outside stakeholders early:
  - Incident Response
  - Red Team
  - Threat Intel
  - Upper Management

# Parting Advice

- Start small
- Get your content creators on board up front
- Bring in outside stakeholders early:
  - Incident Response
  - Red Team
  - Threat Intel
  - Upper Management
- Put a lot of thought into your definition language schema

# Parting Advice

- Start small
- Get your content creators on board up front
- Bring in outside stakeholders early:
  - Incident Response
  - Red Team
  - Threat Intel
  - Upper Management
- Put a lot of thought into your definition language schema
- Make it *(as) easy (as possible)*

# Additional Resources

- [SEC1197C - Build Detection as Code Like the Splunk Threat Research Team](#) ([slides](#))
- [Detection-as-code: Why it works and where to start](#)
- [Detectionengineering.io](#)
- [Sigma](#)
- [Splunk Security Content Repo](#)
- Me! ([email](#) | [LinkedIn](#) | [splunk-usergroups](#): @Nate Zastrow)

# Questions?

# Thank You

