mWISE
MANDIANT WORLDWIDE
INFORMATION SECURITY EXCHANGE
SEPTEMBER 18-20, 2023 | WASHINGTON, D.C.

Security Operations

Continuous Validation: Trusting Your Detections Even When They Don't Fire

AJ King

Director of Threat Research at SnapAttack

SNAPATTACK

About Me

AJ King

Director of Threat Research at SnapAttack

SNAPATTACK

SIMSPACE

PROFICIO™

POINT LOMA NAZARENE UNIVERSITY

ESET

Career Snapshot

- Malware Remediation
- Security Administration
- Security Analysis
- Detection Engineering
- Threat Research
- Training development
- Leadership

Agenda

What will we cover in this talk?

- Situation, Complication, Question
- Why Validation Matters
- The Pipeline in Stages
- Questions The Pipeline Answers
- Divining into The Pipeline
- Example Pipelines
- Putting it all Together
- Challenges / Solutions
- Key Takeaways
- Questions

SNAPATTACK

mWISE

Introduction

Situation	New Detection Engineering teams often manually create rules in every security product.
Complication	The library of detections quickly becomes unmanageable and unverifiable.
Question	How can you be sure your detection library is providing value?

SNAPATTACK

mWISE

Adopting a detection-as-code approach will increase confidence in your detection library by **reducing human error and validating your detections against attack data.**

SNAPATTACK

mWISE

Why Validation Matters

$4.45M

Average Cost of a Data Breach

what's the problem?

Threat Detection is unreliable, slow, and inaccurate

33% Detection Failure Rate

Security teams and tools fail to detect incidents one-third of the time.

Mean Time To Identify 182 Days

Security teams take too long to identify breaches.

24% Technique Coverage

Security tools miss 76% of all MITRE ATT&CK techniques used by adversaries.

IBM, Cost of a Data Breach Report, 2023
Cardinal Ops, 3rd Annual State of SIEM Detection Risk, 2023

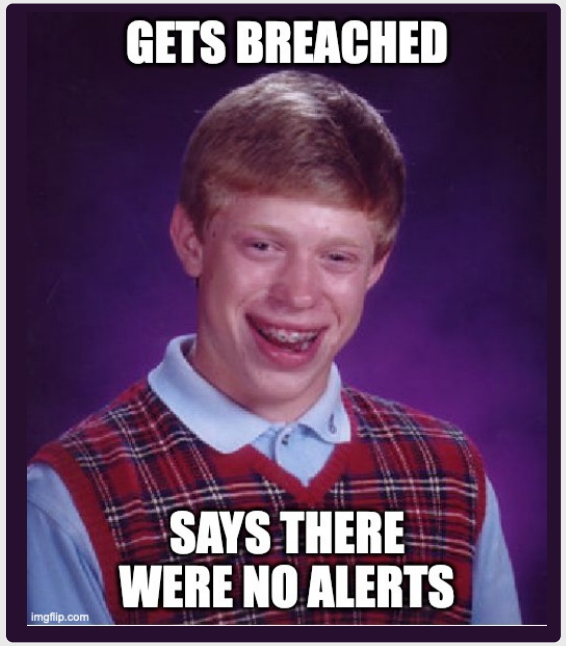
SNAPATTACK

mWISE

What's wrong with this Analytic? Zero Alerts == Zero Problems?

Alerts Last 90 Days - 0

```
index=main sourcetype=Microsoft-Windows-Sysmon\Operational
Command_Line='net localgroup  administrators'
Image="C:\WINDOWS\system32\net.exe"
```



SNAPATTACK

mWISE

What's wrong with this Analytic?

```
index=main sourcetype=Microsoft-Windows-Sysmon\Operational
Command_Line='net localgroup  administrators'
Image="C:\WINDOWS\system32\net.exe"
```

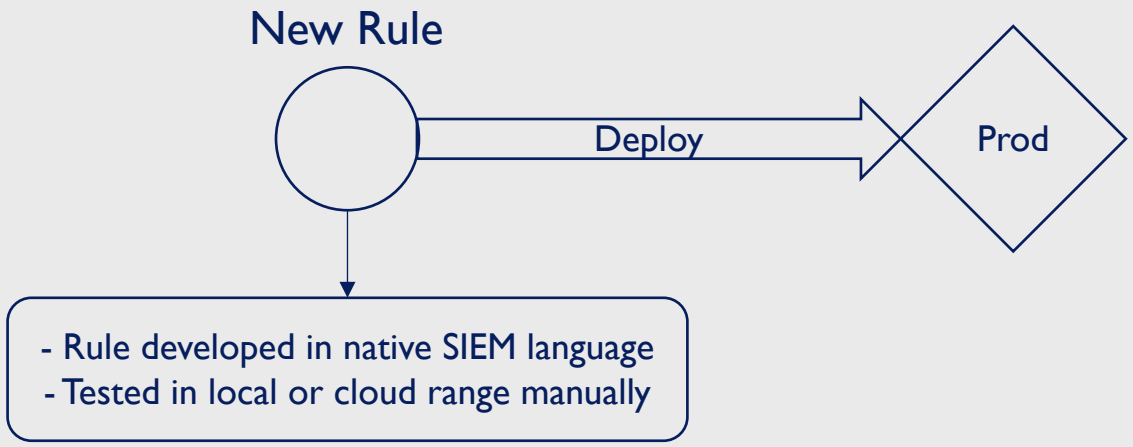
HERE ARE JUST SOME OF THE MOST COMMON ERRORS WE SEE …

- Incorrect sourcetype, backslash instead of forward slash → **Human error – user typo**
- Field name is CommandLine, not Command_Line → **Data model mismatch**
- Values need to be enclosed in double quotes, not single quotes → **Wrong quote / syntax error**
- Extra space in command; is matching the exact command including spaces → **Invisible space or lack of**
- Paths need to be escaped with a second backslash → **Escaping special chars**
- net.exe spawns net1.exe (when net was patched for Y2K) → **Windows quirks**

SNAPATTACK

mWISE

Stage One – No Automation



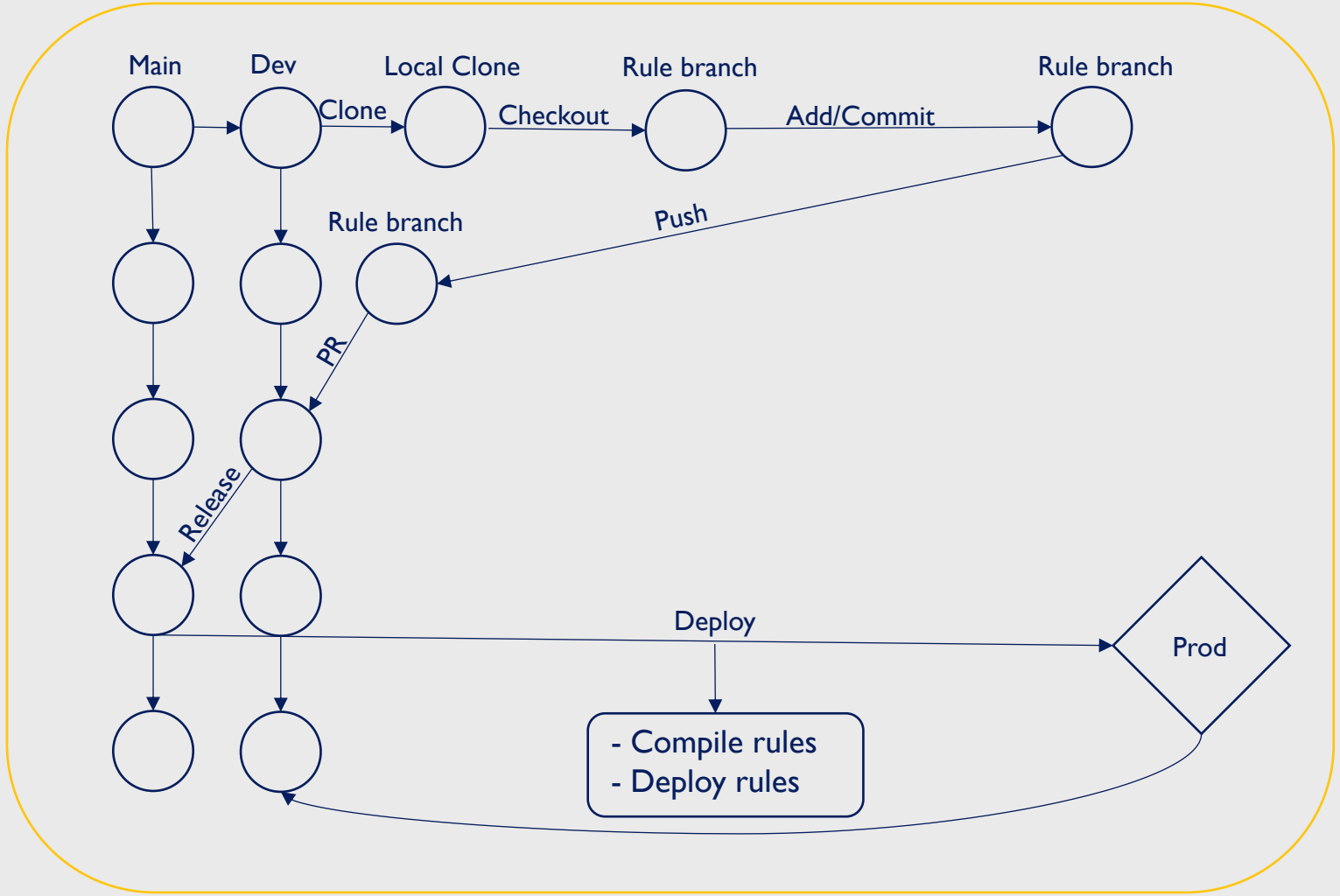
Issues

- Syntax issues found after deployment
- High false positive rates discovered after deployment
- Original rule developer leaves and no one knows what the rule was for or if it works since there is no unit test
- No versioning

SNAPATTACK

mWISE

Stage Two – Automation But No Testing

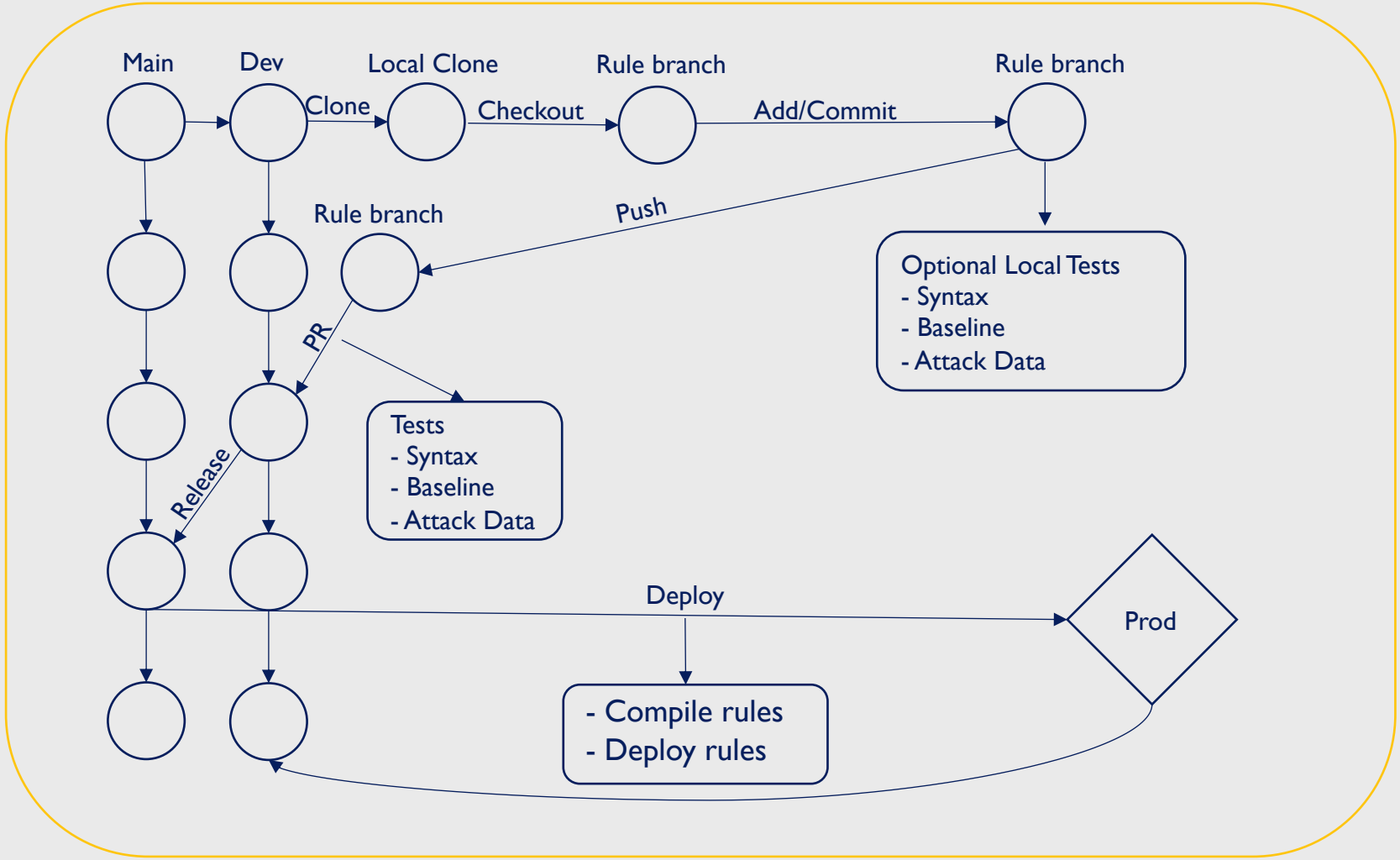


- Broken rules are sitting in prod undiscovered.
- Analysts are overwhelmed with false positives but there is no process to iterate on rules.
- Rules alert but don't detect the intended threat activity.

SNAPATTACK

mWISE

Stage Three – No Post-Deployment Process

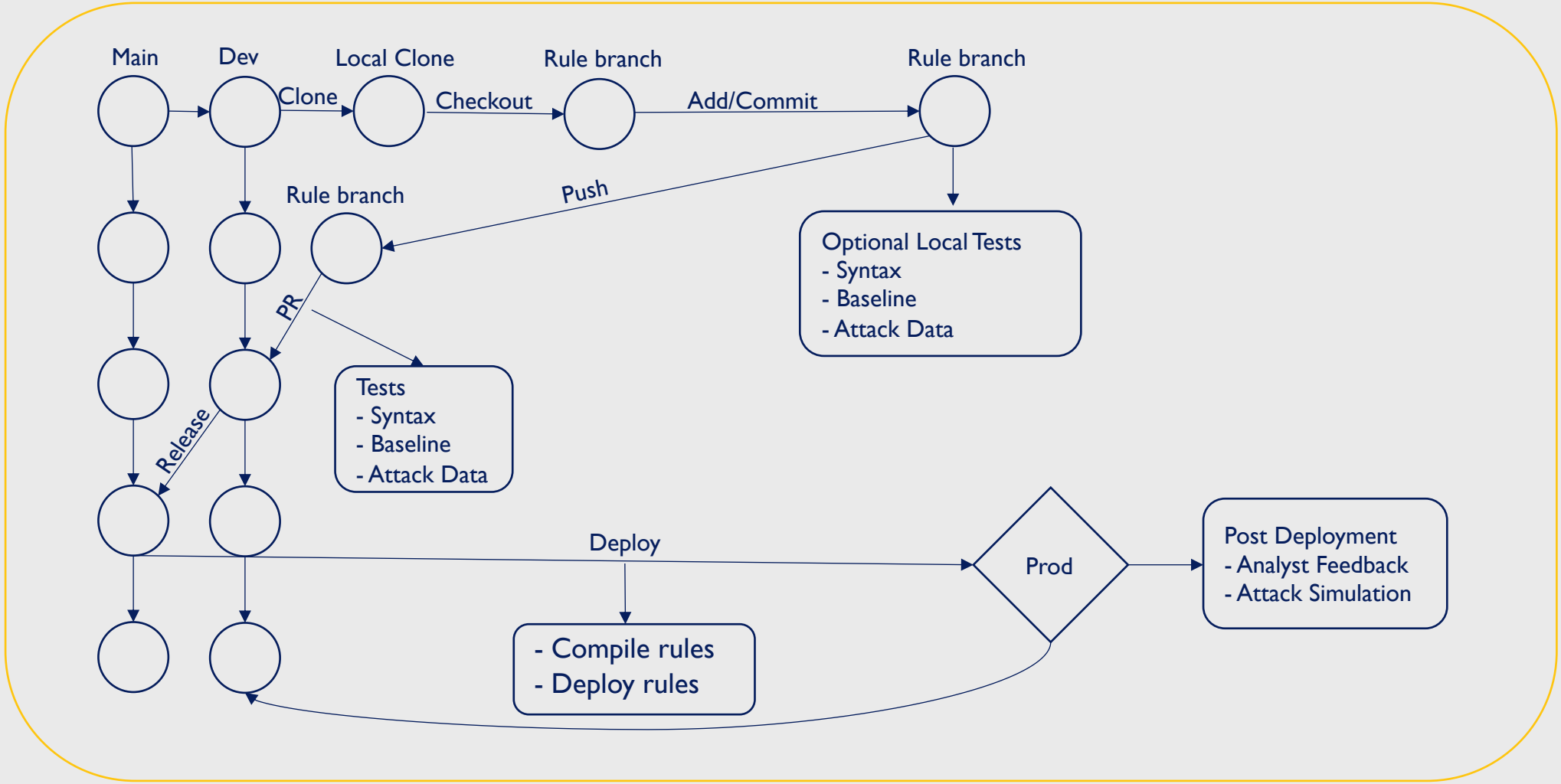


- Rules hit prod and don't work due to logging issues.
- Analysts aren't providing feedback and false positive rates climb.

SNAPATTACK

mWISE

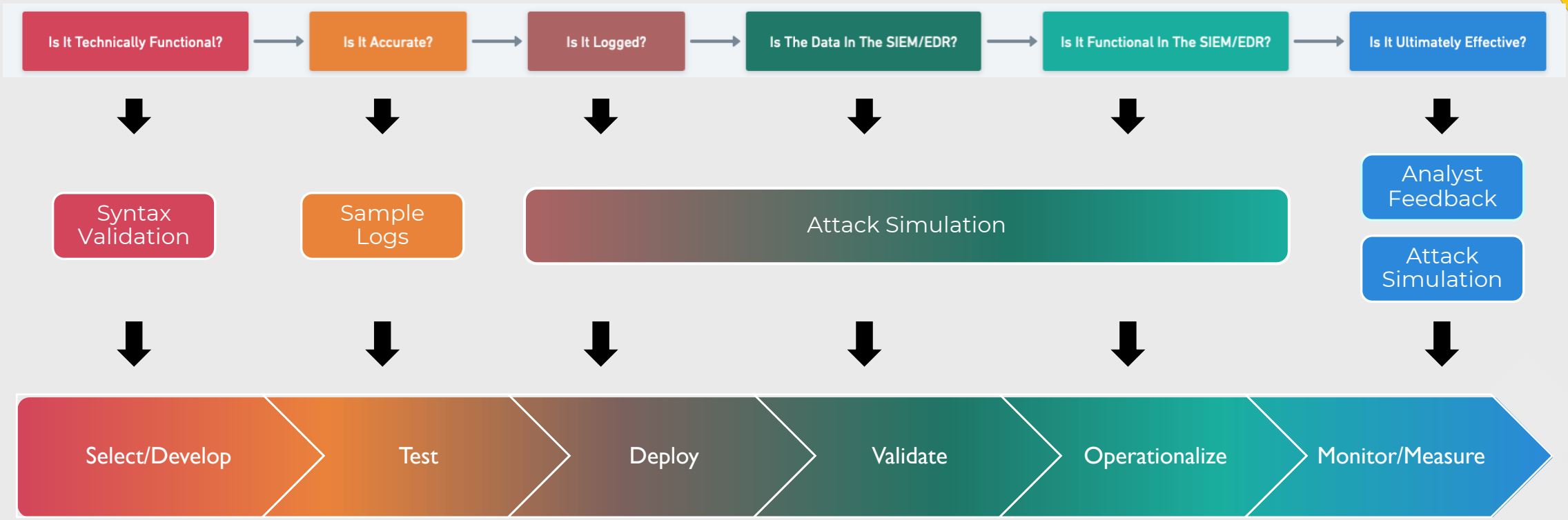
The Pipeline



SNAPATTACK

mWISE

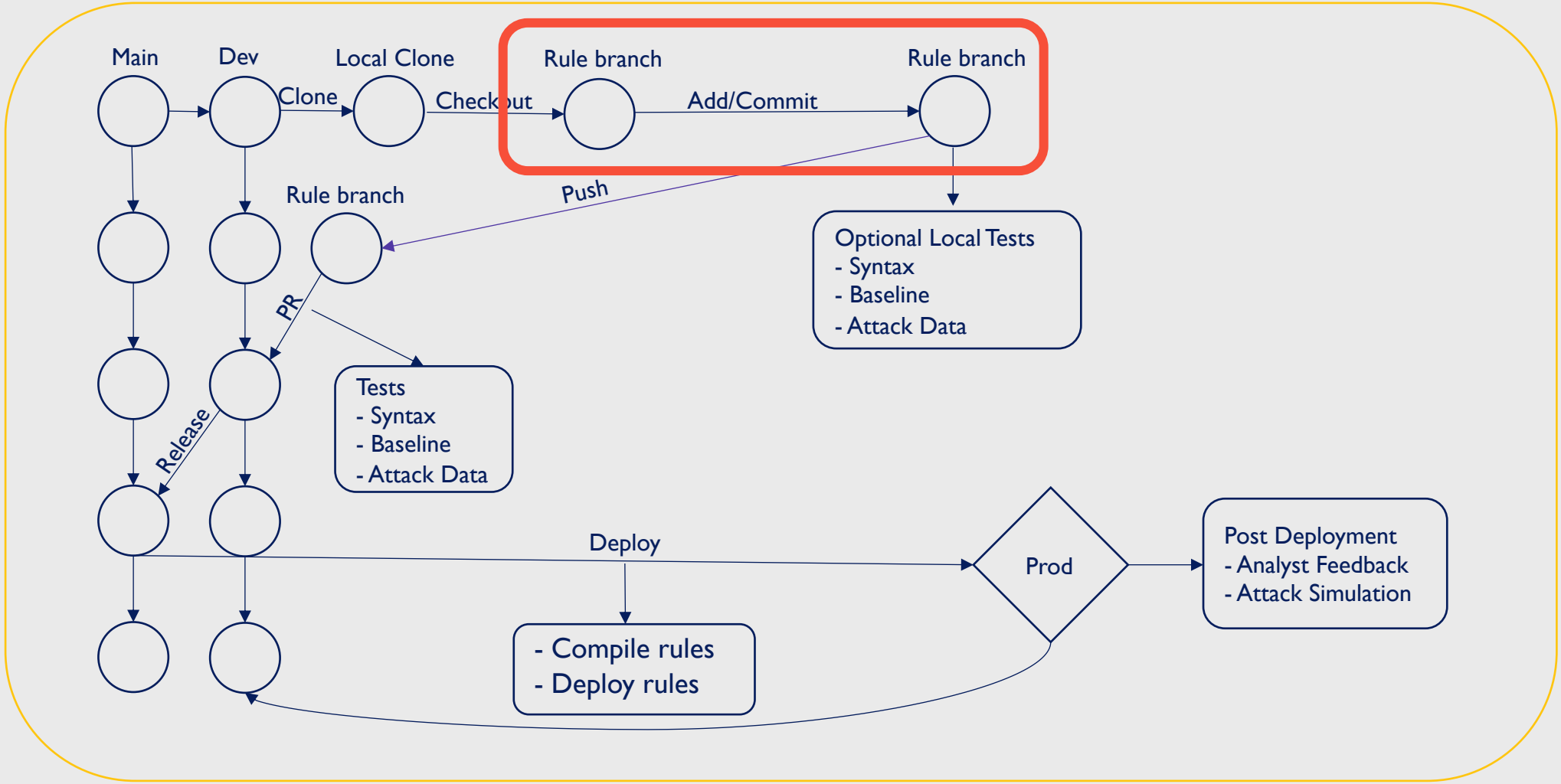
What, How, When



SNAPATTACK

mWISE

The Pipeline - Detection Engineer Workflow



SNAPATTACK

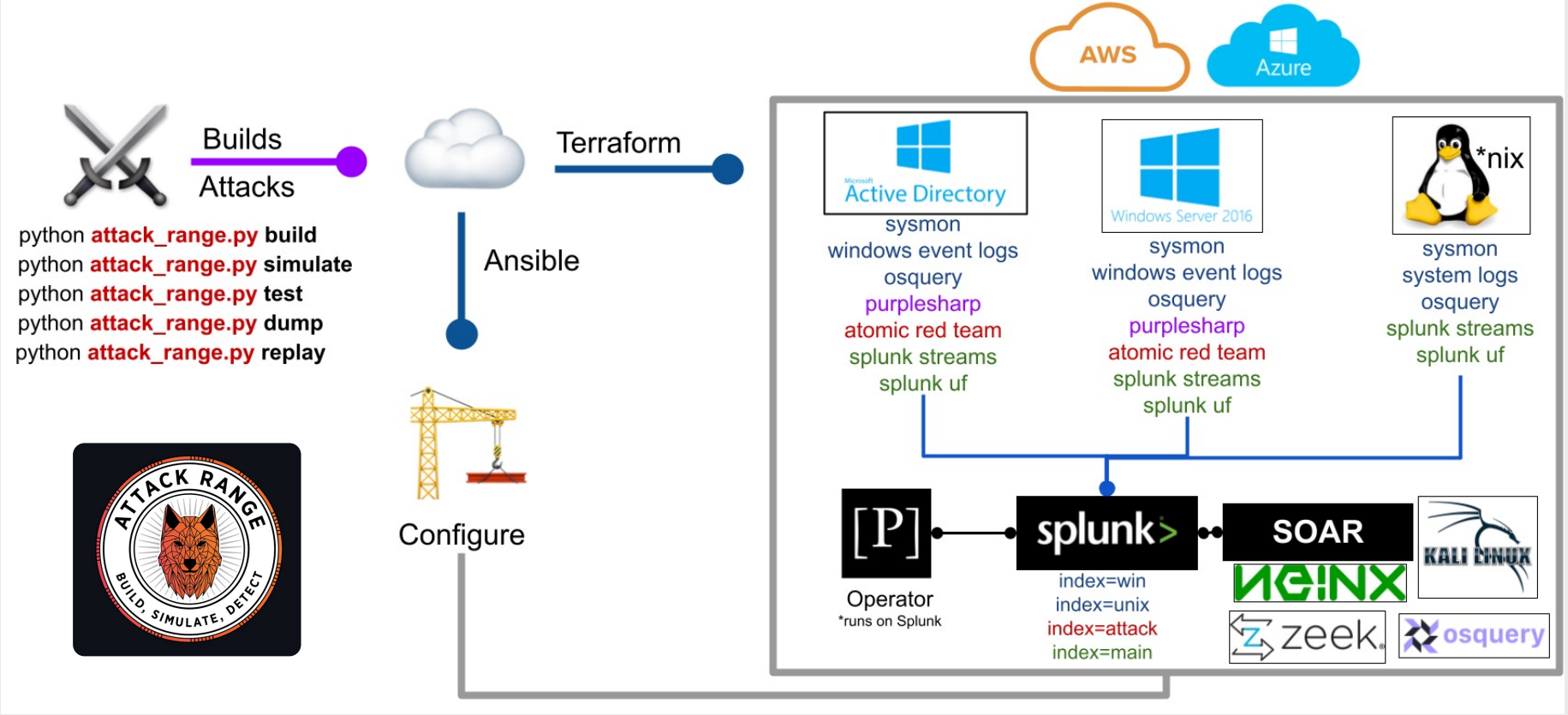
mWISE

Detection Engineer Workflow

SNAPATTACK

mWISE

Infrastructure-As-Code



SNAPATTACK

mWISE

Detection-As-Code

```yaml
title: Powershell WMI Persistence
id: 9e07f6e7-83aa-45c6-998e-0af26efd0a85
status: test
description: Adversaries may establish persistence and elevate privileges by executing malicious content triggered by a Windows Management Instrumentation (WMI) event subscription.
references:
    - https://github.com/redcanaryco/atomic-red-team/blob/f339e7da7d05f6057fdfcdd3742bfcf365fee2a9/atomics/T1546.003/T1546.003.md
    - https://github.com/EmpireProject/Empire/blob/08cbd274bef78243d7a8ed6443b8364acd1fc48b/data/module_source/persistence/Persistence.psm1#L545
author: frack113
date: 2021/08/19
modified: 2022/12/25
tags:
    - attack.privilege_escalation
    - attack.t1546.003
logsource:
    product: windows
    category: ps_script
    definition: 'Requirements: Script Block Logging must be enabled'
detection:
    selection_ioc:
        - ScriptBlockText|contains|all:
            - 'New-CimInstance '
            - '-Namespace root/subscription '
            - '-ClassName __EventFilter '
            - '-Property ' #is a variable name
        - ScriptBlockText|contains|all:
            - 'New-CimInstance '
            - '-Namespace root/subscription '
            - '-ClassName CommandLineEventConsumer '
            - '-Property ' #is a variable name
    condition: selection_ioc
falsepositives:
    - Unknown
level: medium
```

SNAPATTACK

mWISE

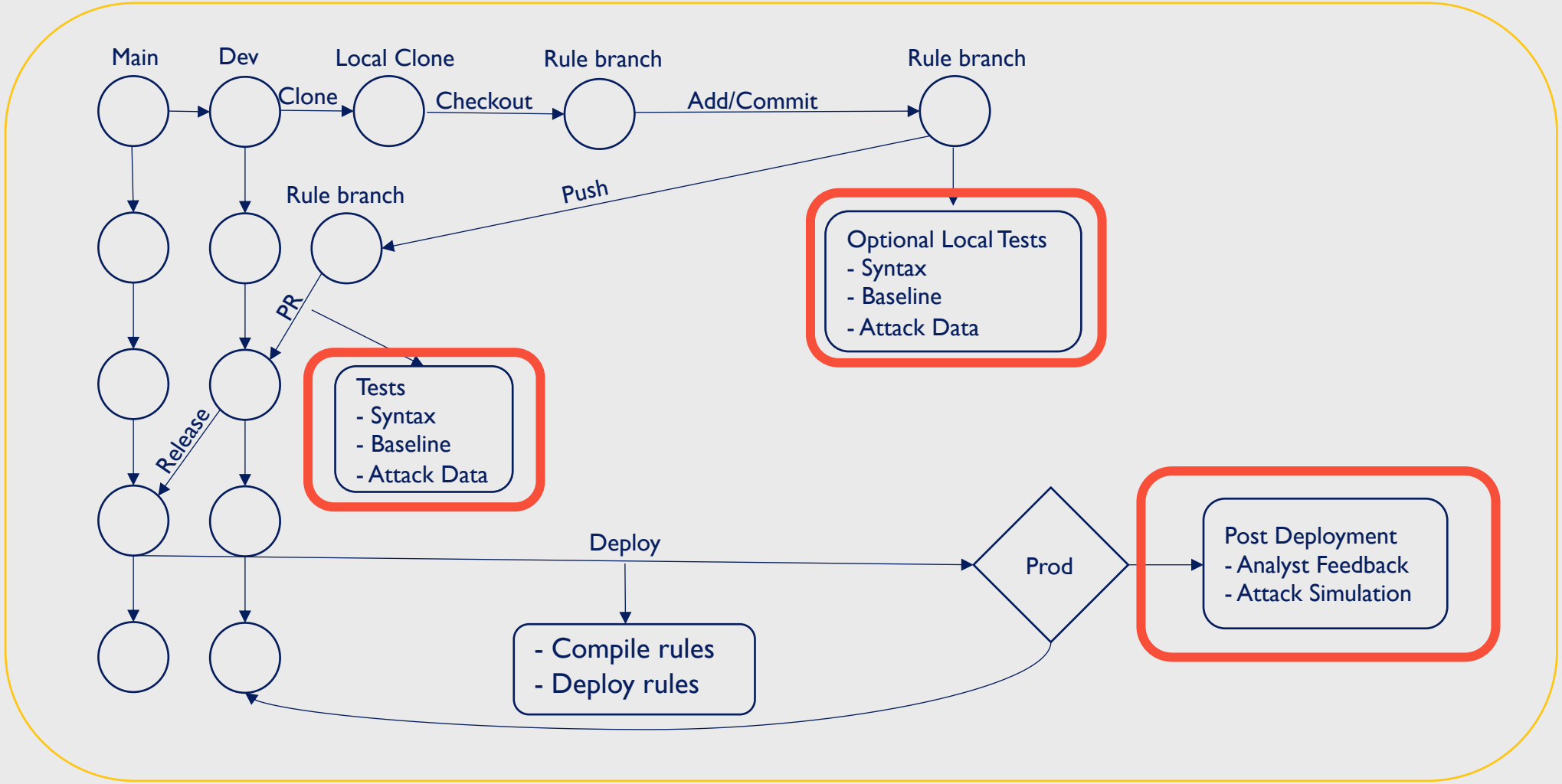
Attack-As-Code

```yaml
attack_technique: T1546.003
display_name: 'Event Triggered Execution: Windows Management Instrumentation Event Subscription'
atomic_tests:
- name: Persistence via WMI Event Subscription - CommandLineEventConsumer
  auto_generated_guid: 3c64f177-28e2-49eb-a799-d767b24dd1e0
  description: |
    Run from an administrator powershell window. After running, reboot the victim machine.
    After it has been online for 4 minutes you should see notepad.exe running as SYSTEM.

    Code references

    https://gist.github.com/mattifestation/7fe1df7ca2f08cbfa3d067def00c01af

    https://github.com/EmpireProject/Empire/blob/master/data/module_source/persistence/Persistence.psm1#L545
  supported_platforms:
  - windows
  executor:
    command: |
      $FilterArgs = @{name='AtomicRedTeam-WMIPersistence-CommandLineEventConsumer-Example';
                      EventNameSpace='root\CimV2';
                      QueryLanguage="WQL";
                      Query="SELECT * FROM __InstanceModificationEvent WITHIN 60 WHERE TargetInstance ISA 'Win32_PerfFormattedData_PerfOS_System' AND TargetInstance.SystemUpTime >= 240 AND TargetInstance.SystemUpTime < 325"};
      $Filter=New-CimInstance -Namespace root/subscription -ClassName __EventFilter -Property $FilterArgs

      $ConsumerArgs = @{name='AtomicRedTeam-WMIPersistence-CommandLineEventConsumer-Example';
                      CommandLineTemplate="$($Env:SystemRoot)\System32\notepad.exe";}
      $Consumer=New-CimInstance -Namespace root/subscription -ClassName CommandLineEventConsumer -Property $ConsumerArgs

      $FilterToConsumerArgs = @{
      Filter = [Ref] $Filter;
      Consumer = [Ref] $Consumer;
      }
      $FilterToConsumerBinding = New-CimInstance -Namespace root/subscription -ClassName __FilterToConsumerBinding -Property $FilterToConsumerArgs
    cleanup_command: |
      $EventConsumerToCleanup = Get-WmiObject -Namespace root/subscription -Class CommandLineEventConsumer -Filter "Name = 'AtomicRedTeam-WMIPersistence-CommandLineEventConsumer-Example'"
      $EventFilterToCleanup = Get-WmiObject -Namespace root/subscription -Class __EventFilter -Filter "Name = 'AtomicRedTeam-WMIPersistence-CommandLineEventConsumer-Example'"
      $FilterConsumerBindingToCleanup = Get-WmiObject -Namespace root/subscription -Query "REFERENCES OF {$($EventConsumerToCleanup.__RELPATH)} WHERE ResultClass = __FilterToConsumerBinding" -ErrorAction SilentlyContinue
      $FilterConsumerBindingToCleanup | Remove-WmiObject
      $EventConsumerToCleanup | Remove-WmiObject
      $EventFilterToCleanup | Remove-WmiObject
    name: powershell
    elevation_required: true
```

SNAPATTACK

mWISE

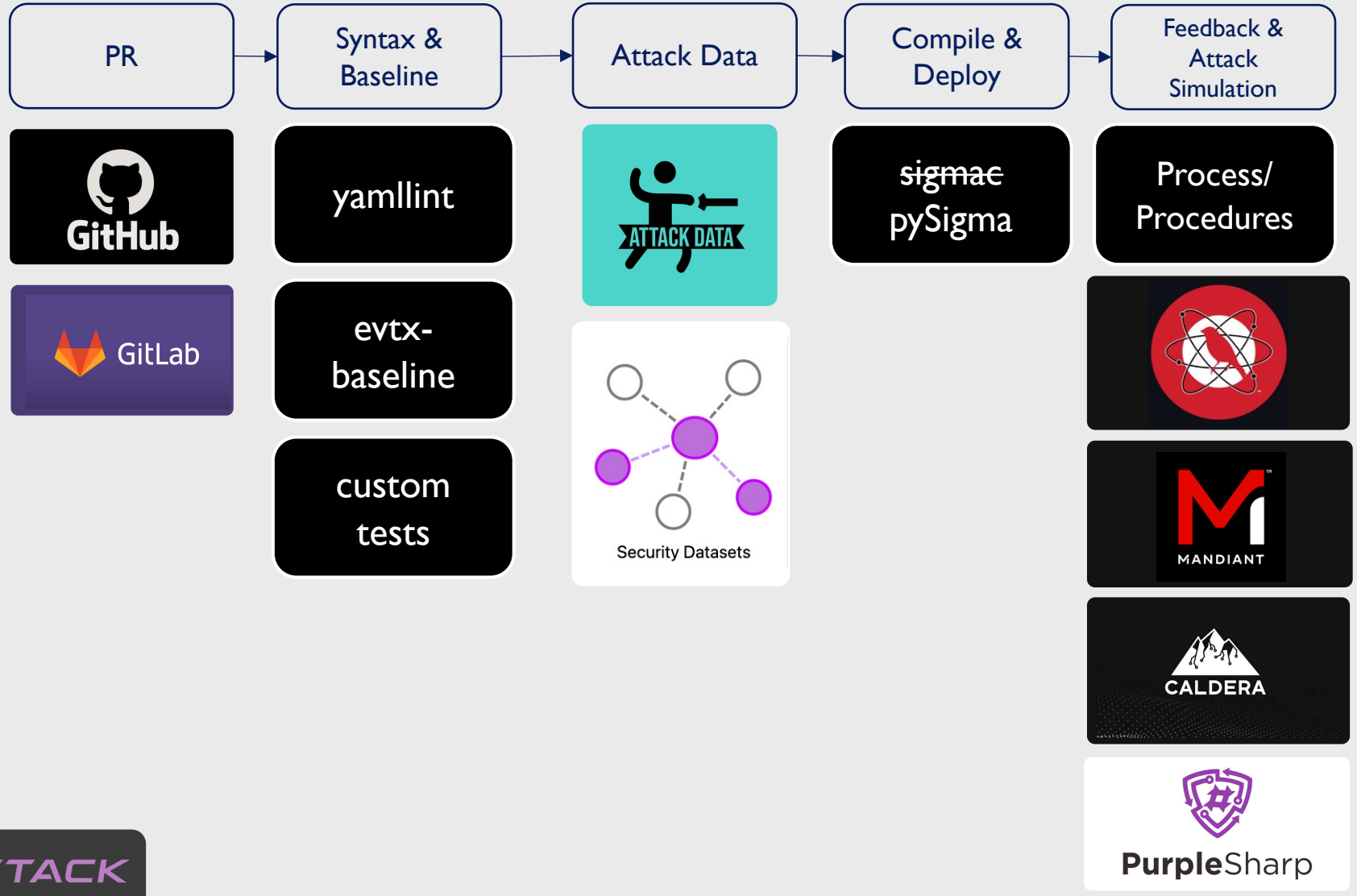
The Pipeline - Validate, Deploy, Feedback



SNAPATTACK

mWISE

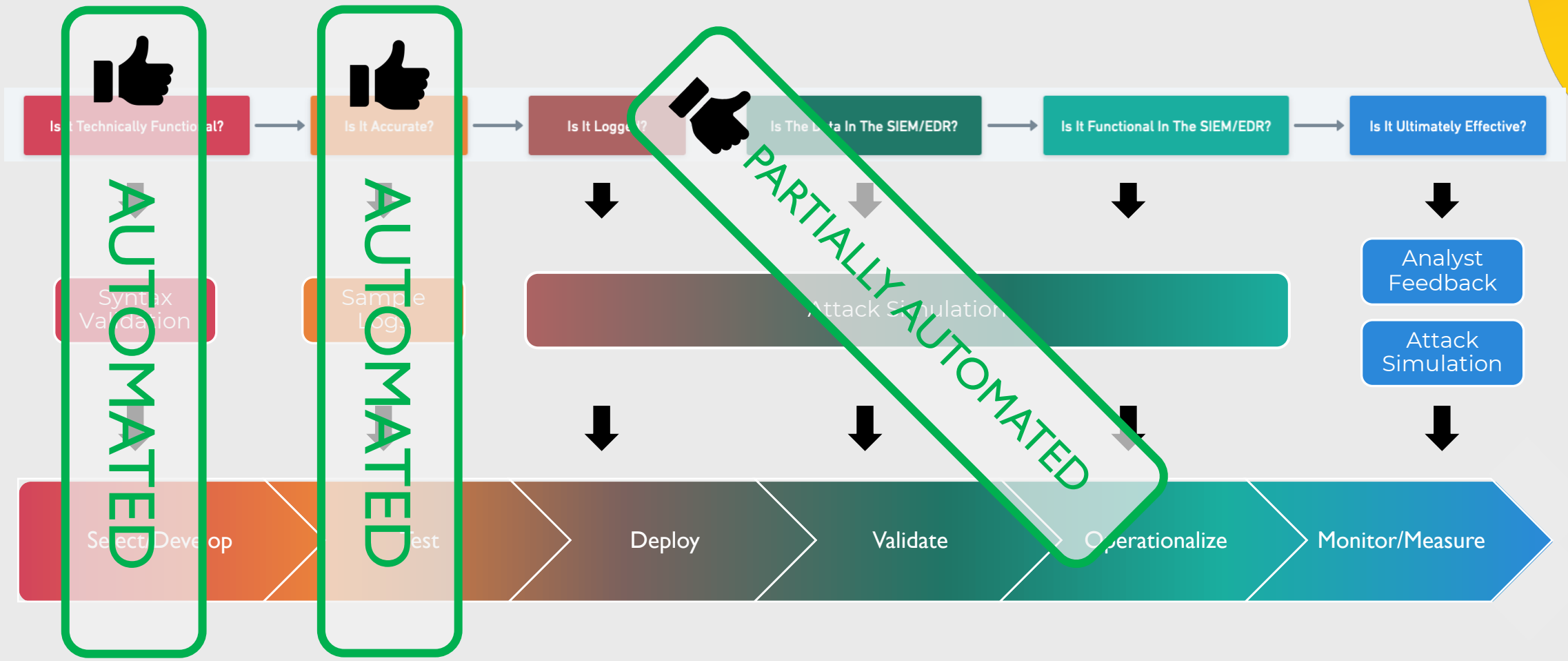
Validate, Deploy, Feedback



SNAPATTACK

mWISE

What, How, When – Automated



SNAPATTACK

mWISE

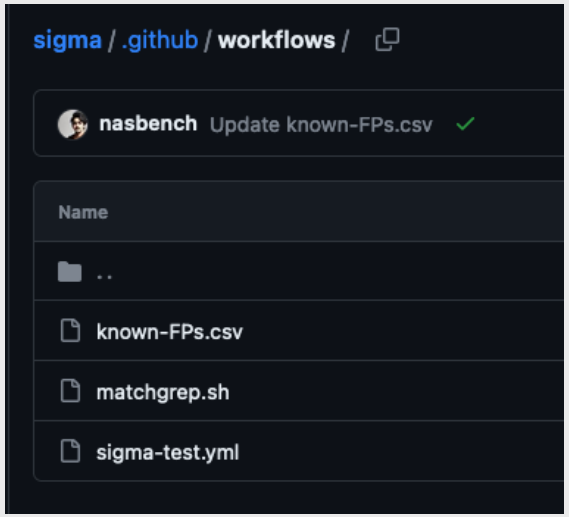
Syntax & Baseline Example (Sigma)

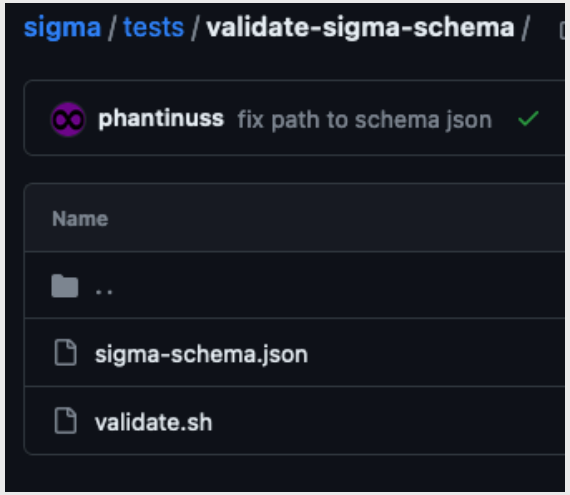
validate.sh: runs a check to ensure the rule aligns with the Sigma schema.

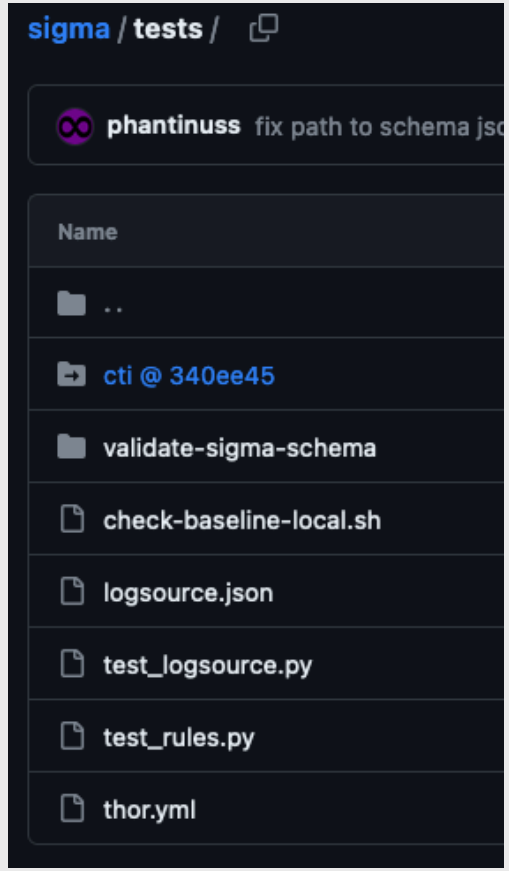
check-baseline-local.sh: runs evtx-sigma-checker against Sigma rules using evtx-baseline data.

test_logsource.py: checks for logsource or fieldname errors on all rules.

test_rules.py: checks for noncompliance or common errors on all rules.







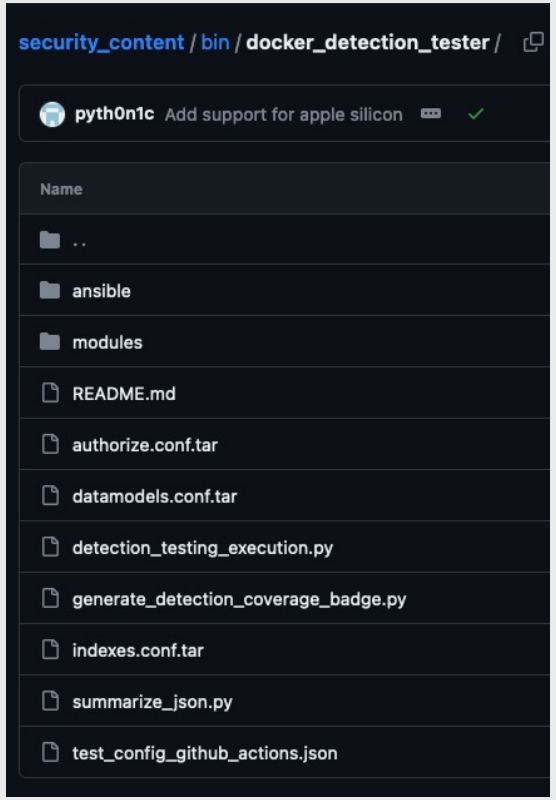
SNAPATTACK

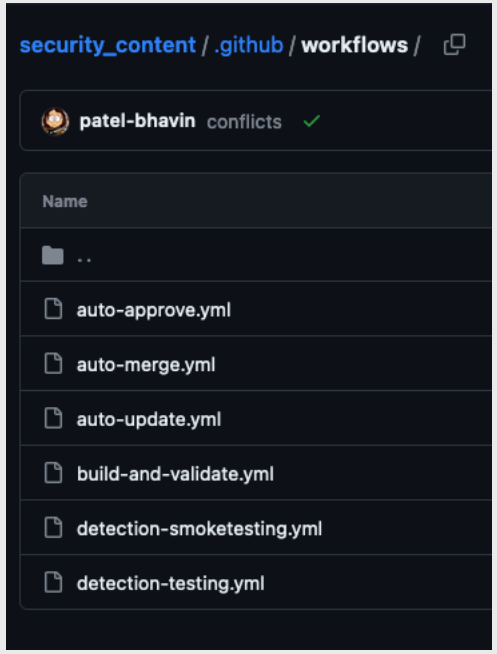
mWISE

Attack Data Example (Splunk)

Splunk Security Content: docker_detection_tester

1. Downloads the latest version of the security_content Repo
2. Lints/Sanity Checks all the Detections
3. Builds an ESCU Splunk App
4. Starts Docker containers (available from Docker Hub as splunk/splunk:latest), installing required Splunkbase Apps and ESCU
5. Distributes Detection Tests Across those Containers
6. Summarizes the Results of those Detections





SNAPATTACK

mWISE

Putting it all together (GitHub)

1. Store rules in GitHub repository in standard format (Sigma).
2. Create GitHub Actions and scripts to test rules on Pull Request.
3. Create GitHub Actions and scripts to compile and deploy rules.
4. Create feedback loop between Analysts and Detection Engineers.
5. Regularly run Attack Scripts to validate logging and deployed rules.

Challenges

- You need a team with coding and git knowledge.
- New detection may require new datasets or attack scripts for validation.
- Pipelines can be complicated to maintain especially with turnover.
- When the pipeline breaks you need to fix it.
- You still need to test rules once deployed in production.

Solutions

- Invest in continuous learning for your team and/or have them collaborate with DevOps.
- Lean on community projects to gain access to crowdsourced data.
- Create a culture of documentation -- encourage your team to write as much as possible.
- Create a culture of automation -- when things break encourage your team to write tests and fixes into the pipeline and document the process.
- Set aside dedicated times to run attack scripts and test detections. Once you iron this out for your environment you can automate it.
- If you don't have the ability to do build a pipeline in-house explore vendor solutions. Come talk to us at booth 206.

SNAPATTACK

mWISE

- Zero alerts does not equal zero problems.
- A validation pipeline will increase confidence in your detection library.
- A CI/CD workflow via GitHub is just one of many ways to create a pipeline -- use what works for your environment.
- There are strong community projects available – use them.
- Automation and documentation will make your pipeline sustainable.
- Learn from what has already been created.

Questions?

AJ King

Director of Threat Research at SnapAttack

X (Formerly Twitter): @ajkingio

Email: ajking@snapattack.com

Reading List: ajking.io/mwise

SNAPATTACK

mWISE