FalconForce

Infrastructure as Code, Automation, and Testing: The Key to Unlocking the Power of Detection Engineering

NORTHSEC, MONTREAL 2023

Olaf Hartong

Detection Engineer and Security Researcher

- Purple teaming, Threat hunting
- IR and Compromise assessments

Former documentary photographer
Father of 2 boys
“I like warm hugs”

@olafhartong
github.com/olafhartong
olaf@falconforce.nl
olafhartong.nl / falconforce.nl

What you can expect from this talk

- Why we started doing this
- What does detection as code mean?
- How we document and store our detections
- The benefits of automatic deployment and testing
- Automatic validation of your detections

FalconForce Sentry

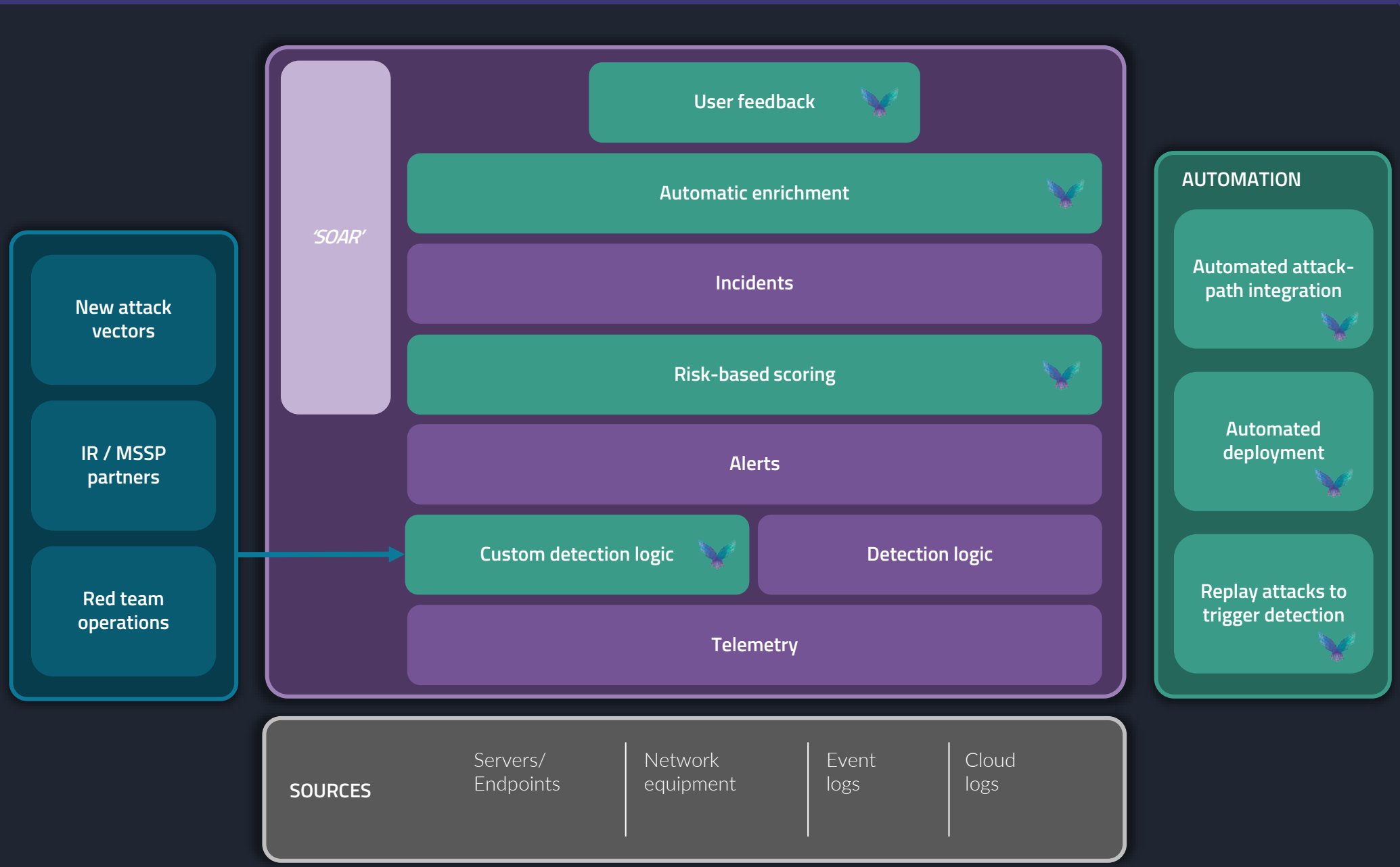
Our vision on threat detection evolution

Most basic threat detection model

Reshaping the model

- Stock content can be bypassed.
- Time wasted on repetitive tasks by an analyst.
- Alert fatigue due to large number of FPs.
- Underutilization of telemetry.
- No insight in operating effectiveness.

OVERVIEW – FalconForce Sentry



Detection as code

Detection as code - Removes most of the challenges

- Who changed rule x and what changed ?
- Will we break the detection with this modification ?
- When was rule x implemented and when was it last changed ?
- Can we assure the quality of a detection and its documentation ?
- Is my detection logic still working as expected ?

What is detection as code to us

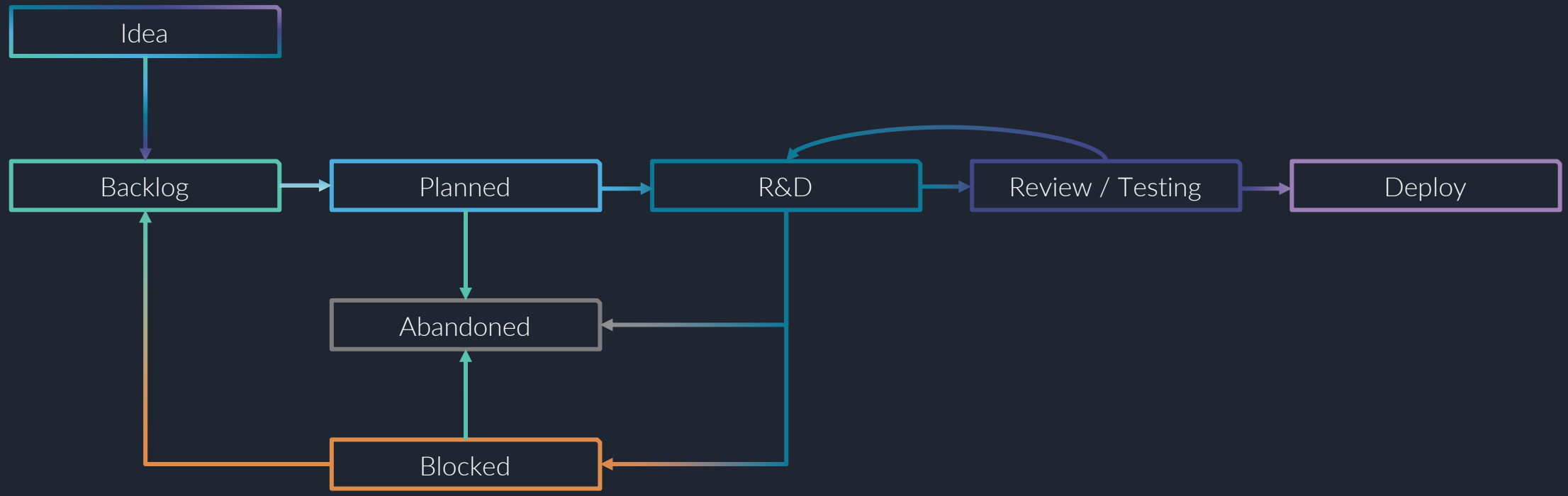
- Follow an Agile process and workflow
- Use a simple language and plan reusability
- Version controlled with peer review options
- Driven by built-in testing of the logic and docs
- Unit testing based on realistic attacks

Agile process flow

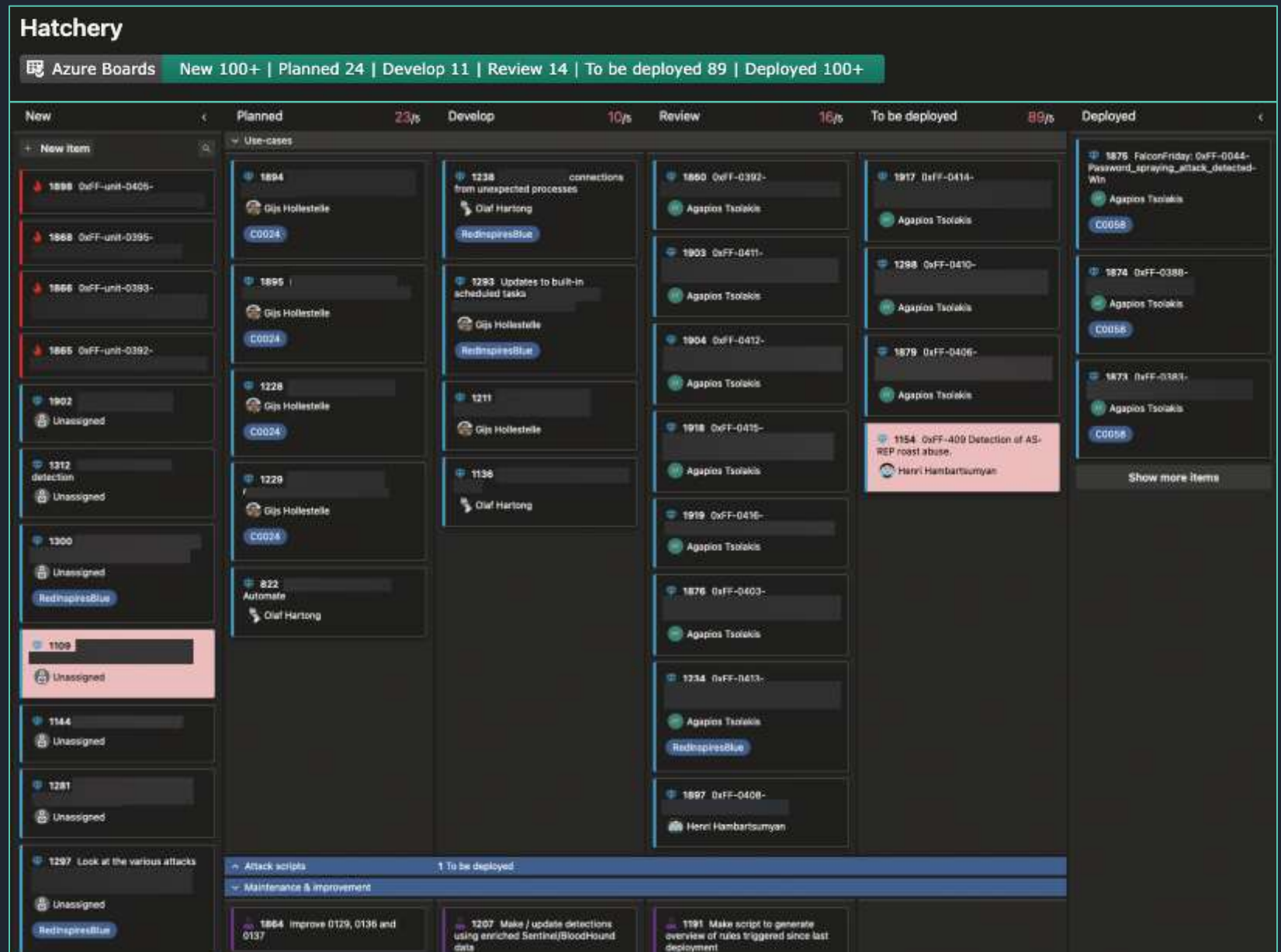
Follow an Agile process and workflow

- Have a backlog and prioritize this
- Write dedicated documentation per detection
- Test and review all changes
- Track progress and have standups
- Plan and organize your maintanance

Agile process flow – simplified example



The detection content phases in DevOps



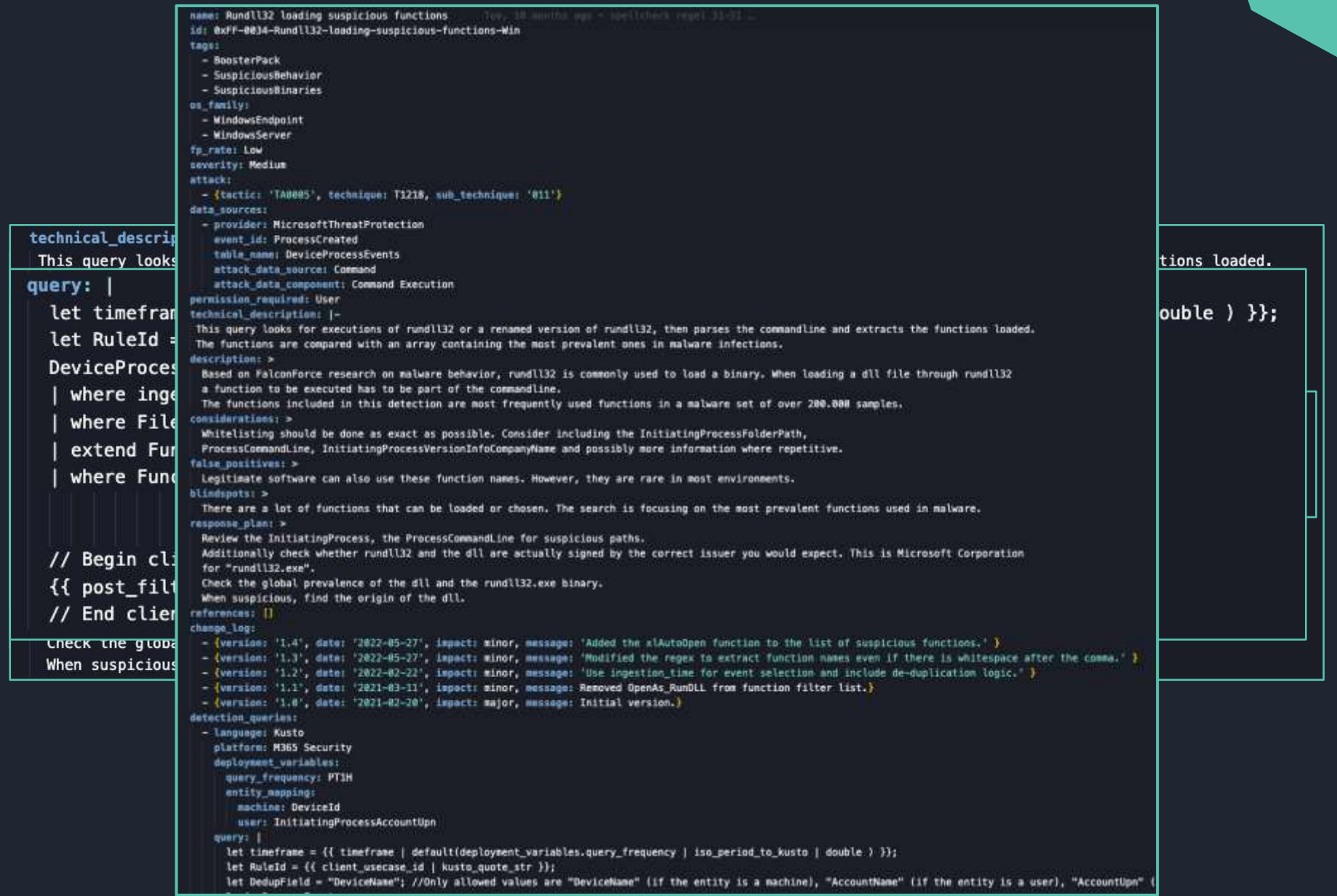
File format

Use a simple language and plan reusability

- Choose an easy to maintain machine-readable format, like YAML
- Make sure to be as expressive as possible and plan for code reuse
- Reuse lists, query components and lookup tables.
- Create a schema for validation and maintenance comfort
- Design for the ability to deploy to multiple environments

YAML structure

- Identifiers
- Tagging
- Used data sources
- Documentation
- Changelog
- Deployment info
- Rule logic



```yaml
name: Rundll32 loading suspicious functions
id: 0xFF-0034-Rundll32-loading-suspicious-functions-Win
tags:
  - BoosterPack
  - SuspiciousBehavior
  - SuspiciousBinaries
os_family:
  - WindowsEndpoint
  - WindowsServer
fp_rate: Low
severity: Medium
attack:
  - {tactic: 'TA0005', technique: T1218, sub_technique: '011'}
data_sources:
  - provider: MicrosoftThreatProtection
    event_id: ProcessCreated
    table_name: DeviceProcessEvents
    attack_data_source: Command
    attack_data_component: Command Execution
permission_required: User
technical_description: |-
 This query looks for executions of rundll32 or a renamed version of rundll32, then parses the commandline and extracts the functions loaded.
 The functions are compared with an array containing the most prevalent ones in malware infections.
description: >
  Based on FalconForce research on malware behavior, rundll32 is commonly used to load a binary. When loading a dll file through rundll32
  a function to be executed has to be part of the commandline.
  The functions included in this detection are most frequently used functions in a malware set of over 200.000 samples.
considerations: >
  Whitelisting should be done as exact as possible. Consider including the InitiatingProcessFolderPath,
  ProcessCommandLine, InitiatingProcessVersionInfoCompanyName and possibly more information where repetitive.
false_positives: >
  Legitimate software can also use these function names. However, they are rare in most environments.
blindspots: >
  There are a lot of functions that can be loaded or chosen. The search is focusing on the most prevalent functions used in malware.
response_plan: >
  Review the InitiatingProcess, the ProcessCommandLine for suspicious paths.
  Additionally check whether rundll32 and the dll are actually signed by the correct issuer you would expect. This is Microsoft Corporation
  for "rundll32.exe".
  Check the global prevalence of the dll and the rundll32.exe binary.
  When suspicious, find the origin of the dll.
references: []
change_log:
  - {version: '1.4', date: '2022-05-27', impact: minor, message: 'Added the xlAutoOpen function to the list of suspicious functions.' }
  - {version: '1.3', date: '2022-05-27', impact: minor, message: 'Modified the regex to extract function names even if there is whitespace after the comma.' }
  - {version: '1.2', date: '2022-02-22', impact: minor, message: 'Use ingestion_time for event selection and include de-duplication logic.' }
  - {version: '1.1', date: '2021-03-11', impact: minor, message: Removed OpenAs_RunDLL from function filter list.}
  - {version: '1.0', date: '2021-02-20', impact: major, message: Initial version.}
detection_queries:
  - language: Kusto
    platform: M365 Security
    deployment_variables:
      query_frequency: PT1H
      entity_mapping:
        machine: DeviceId
        user: InitiatingProcessAccountUpn
    query: |
      let timeframe = {{ timeframe | default(deployment_variables.query_frequency | iso_period_to_kusto | double ) }};
      let RuleId = {{ client_usecase_id | kusto_quote_str }};
      let DedupField = "DeviceName"; //Only allowed values are "DeviceName" (if the entity is a machine), "AccountName" (if the entity is a user), "AccountUpn" (
```

Schema support

```
name: AWS User Accessing Excessive Secrets
id: 0xFF-0235-AWS_User_Accessing_Excessive_Secrets-AWS
tags:
  - AWS
  - SuspiciousBehavior
os_family:
  - N/A
fp_rate: Low
severity: Medium
attack:
  - {tactic: 'TA0009', technique: T1530}
data_sources:
  - provider: AWS
    event_id: GetPasswordData
    table_name: AWSCloudTrail
    attack_data_source: Application Log
    attack_data_component: Application Log Content
permission_required:      You, 3 minutes ago • Uncommitted changes     Incorrect type. Expected "string".
```

- Administrator
- Domain Admin
- Root
- SYSTEM
- User

Made with flexibility in mind

Example query:

```
let timeframe = {{ timeframe|default('1h') }};
let initiators = {{ yaml2kusto(initiators | default(['wsmprovhost.exe’])) }};
let PowershellRemotingEvents = DeviceProcessEvents
| where Timestamp >= ago(timeframe)
| where InitiatingProcessFileName in~ (initiators)
// Customer-specific filter.
{{ post_filter_1 }};
// End customer-specific filter.
{% if (exclude_servers | default(True)) %}
// Filter out servers since Powershell remoting is common on servers.
DeviceInfo
| where DeviceName in (( PowershellRemotingEvents | project DeviceName ))
| where not(isempty(OSPlatform))
| summarize arg_max(Timestamp, OSPlatform) by DeviceName
| join kind=rightouter PowershellRemotingEvents on DeviceName
| where not(OSPlatform contains "server")
{% else %}
PowershellRemotingEvents
{% endif %}
```

Customer 1:

```
exclude_servers: true
post_filter_1: |
      | where not(DeviceName startswith 'br')
      | where not(FileName contains "test")
```

Customer 2:

```
timeframe: 4h
exclude_servers: false
initiators:
      - wsmprovhost.exe
      - powershell.exe
```

Query Customer 1:

```
let timeframe = 1h;
let initiators = “wsmprovhost.exe”;
let PowershellRemotingEvents = DeviceProcessEvents
| where Timestamp >= ago(timeframe)
| where InitiatingProcessFileName in~ (initiators)
// Customer-specific filter.
| where not(DeviceName startswith 'br')
| where not(FileName contains "test")
// End customer-specific filter.
// Filter out servers since Powershell remoting is common on servers.
DeviceInfo
| where DeviceName in (( PowershellRemotingEvents | project DeviceName ))
| where not(isempty(OSPlatform))
| summarize arg_max(Timestamp, OSPlatform) by DeviceName
| join kind=rightouter PowershellRemotingEvents on DeviceName
| where not(OSPlatform contains "server")
```

Query Customer 2:

```
let timeframe = 4h;
let initiators = dynamic(['wsmprovhost.exe’,’powershell.exe’]);
let PowershellRemotingEvents = DeviceProcessEvents
| where Timestamp >= ago(timeframe)
| where InitiatingProcessFileName in~ (initiators)
// Customer-specific filter.
// End customer-specific filter.
PowershellRemotingEvents
```

Version control and peer review

Version controlled with peer review options

- Track all changes via commits
- Allow for (enforced) peer reviews on merge
- Simple to roll back to a previous version
- Have the single source of truth
- Allows pipeline actions for automatic checks and deployments

Version control and peer review

```yaml
name: Rundll32 loading suspicious func
id: 0xFF-0034-Rundll32-loading-suspici
tags:
  - BoosterPack
  - SuspiciousBehavior
  - SuspiciousBinaries
os_family:
  - WindowsEndpoint
  - WindowsServer
fp_rate: Low
severity: Medium
attack:
  - {tactic: 'TA0005', technique: T121
data_sources:
  - provider: MicrosoftThreatProtectio
    event_id: ProcessCreated
    table_name: DeviceProcessEvents
    attack_data_source: Command
    attack_data_component: Command Execution
permission_required: User
```

Olaf Hartong, 13 months ago • added data source and data components - initial …

Initial Commit of 0xFF-0403

Approve | Complete

Active !937 Agapios Tsolakis 0xFF-0403- into master 1/4 comments resolved

Overview Files Updates Commits

No required checks
Optional check succeeded

Syntax validate & deploy ballpit Syntax validate & deploy ballpit succeeded Re-queue

1 reviewer approved

No merge conflicts
Last checked 3m ago

Description

Checklist for PR

The checklist below is a guidance for the submitter as well as for the reviewer.

- [x] The query uses case-insensitive string comparisons (unless performance is explicitly an issue due to expected table size).
- [x] Joins on keys are performed case-insensitive. This requires both column to be converted to lowercase.
- [x] Query generally makes sense and does what the title suggests.
- [x] When possible client_usecase.yml files have been created for Ballpit & FalconForce to deploy in own environment.
- [ ] When possible a testcase.yml has been created containing a test case.

Initial Commit of 0xFF-0403

Reviewers Add

Required

No required reviewers

Optional

Tom Schuurmans
Approved with suggestions

Gijs Hollestelle
No review yet

Olaf Hartong
No review yet

Tags

No tags

Work items

Use Case 1876: 0xFF-0403-Suspi...
Updated 1 May, Review

Automatic validation and deployment

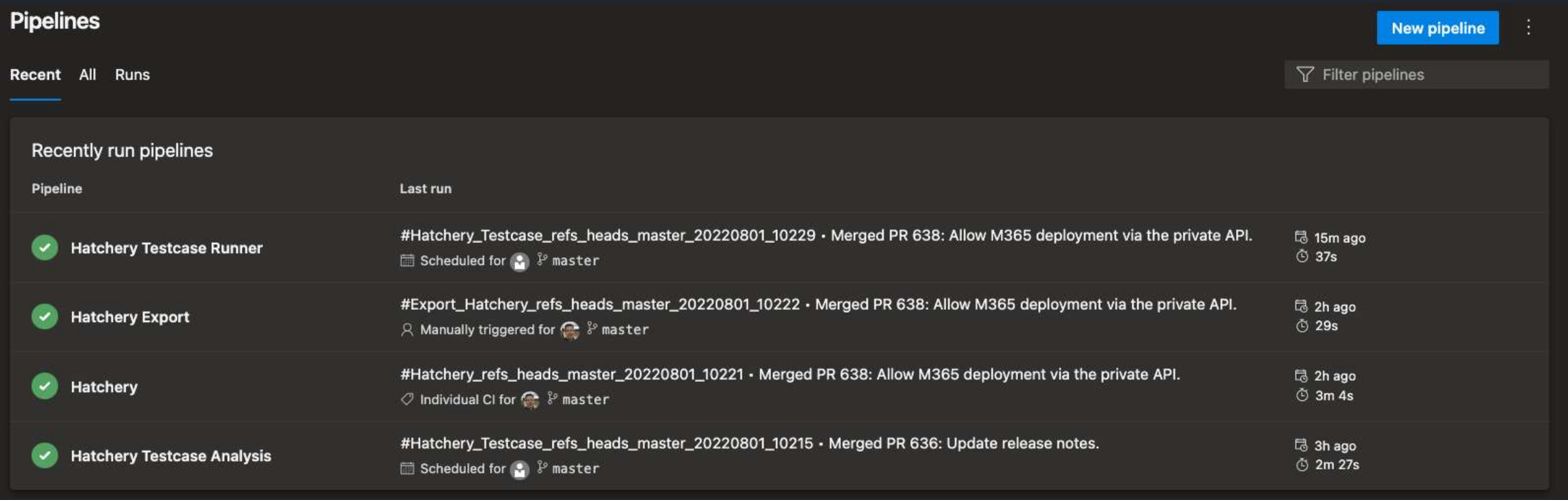
Pipelines

Driven by built-in testing of the logic and docs

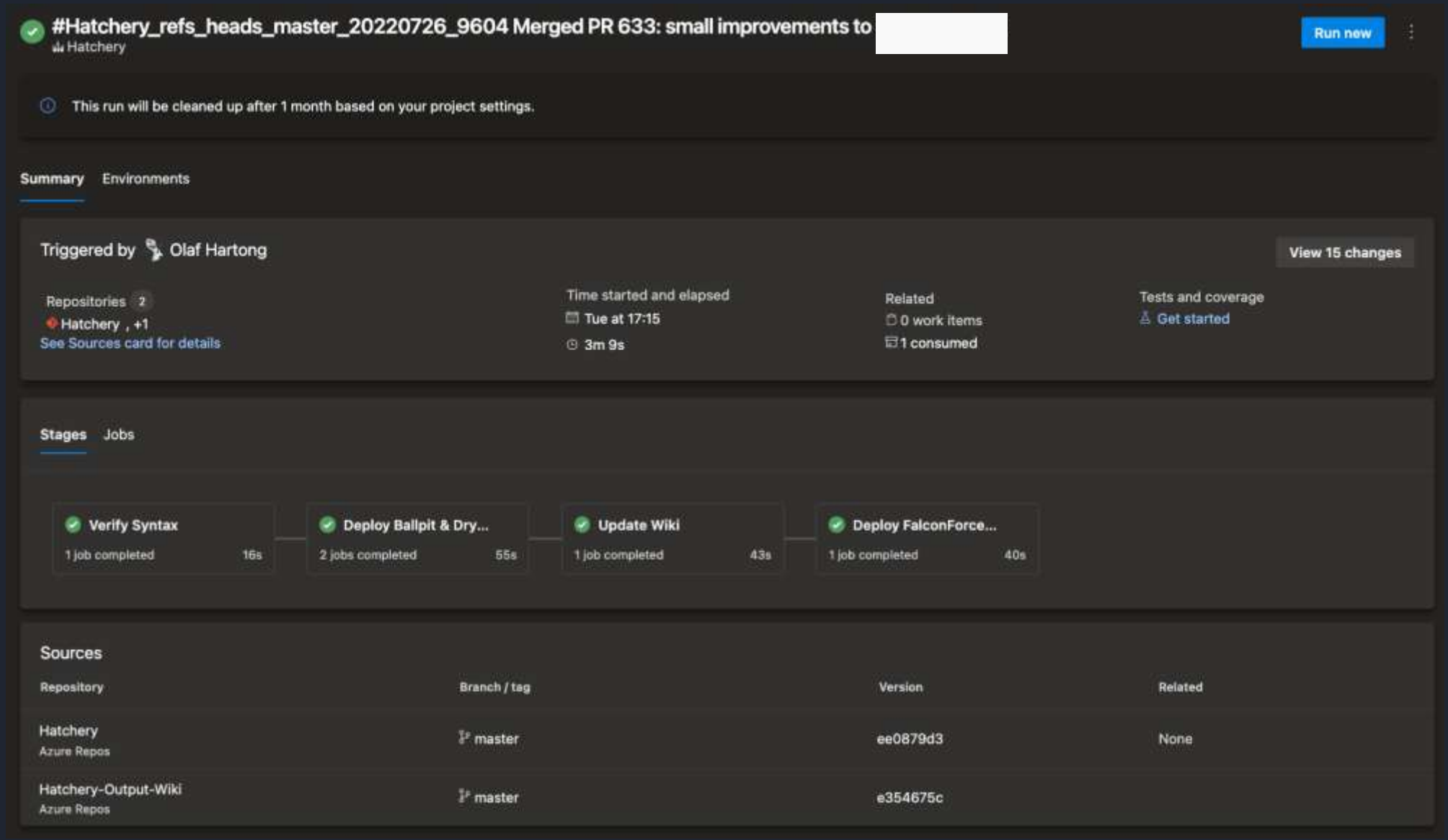
- Enforces static and dynamic testing of each rule
- Linting, language server and best practice checks
- All your environments are always running the latest version
- Generate your documentation and playbooks from the code
- This principle also applies for detection validation (in most cases)

https://medium.com/falconforce/deploying-detections-at-scale-part-0x01-use-case-format-and-automated-validation-7bc76bea0f43

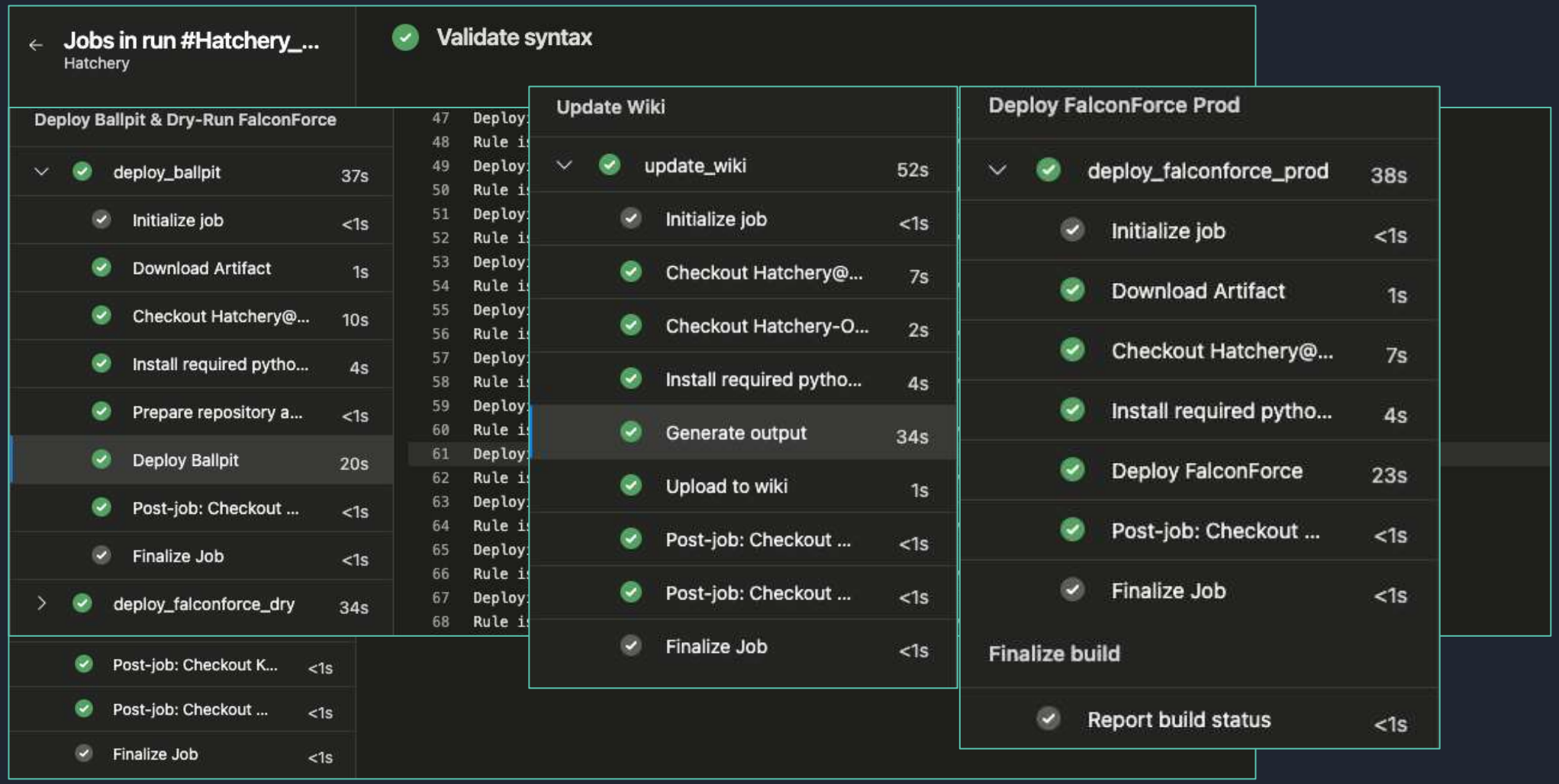
Our pipelines per environment



Our pipelines



Our pipelines



Pipeline error examples

Errors 4

- Warnings while verifying query syntax for 0xFF-0161-Overly_permissive_security_group-AWS, client C001-Demo:
  Verify Syntax • verify_syntax • Validate syntax
- Query contains Syntax errors: [{'code': 'KS142', 'category': 'General', 'severity': 'Error', 'description': "The name 'UserAccount' doe...
  Verify Syntax • verify_syntax • Validate syntax
- Exiting with error since there are unsilenced warnings.
  Verify Syntax • verify_syntax • Validate syntax
- Bash exited with code '1'.
  Verify Syntax • verify_syntax • Validate syntax

KQL language server

This enables us to do offline schema and language-based query validation.

Additionally, match fields, tables and entities in the output to what is mapped in the documentation. For instance, to ensure proper entity mapping.

The language server:

- Parses all Microsoft documentation and updates the schema
- Allows for custom parsers and watchlists
- Emulates the use of functions in Defender, like FileProfile

Available for free >> https://kql.falconforce.blue/api/kqlanalyzer

https://github.com/FalconForceTeam/KQLAnalyzer

KQL language server – simplified example

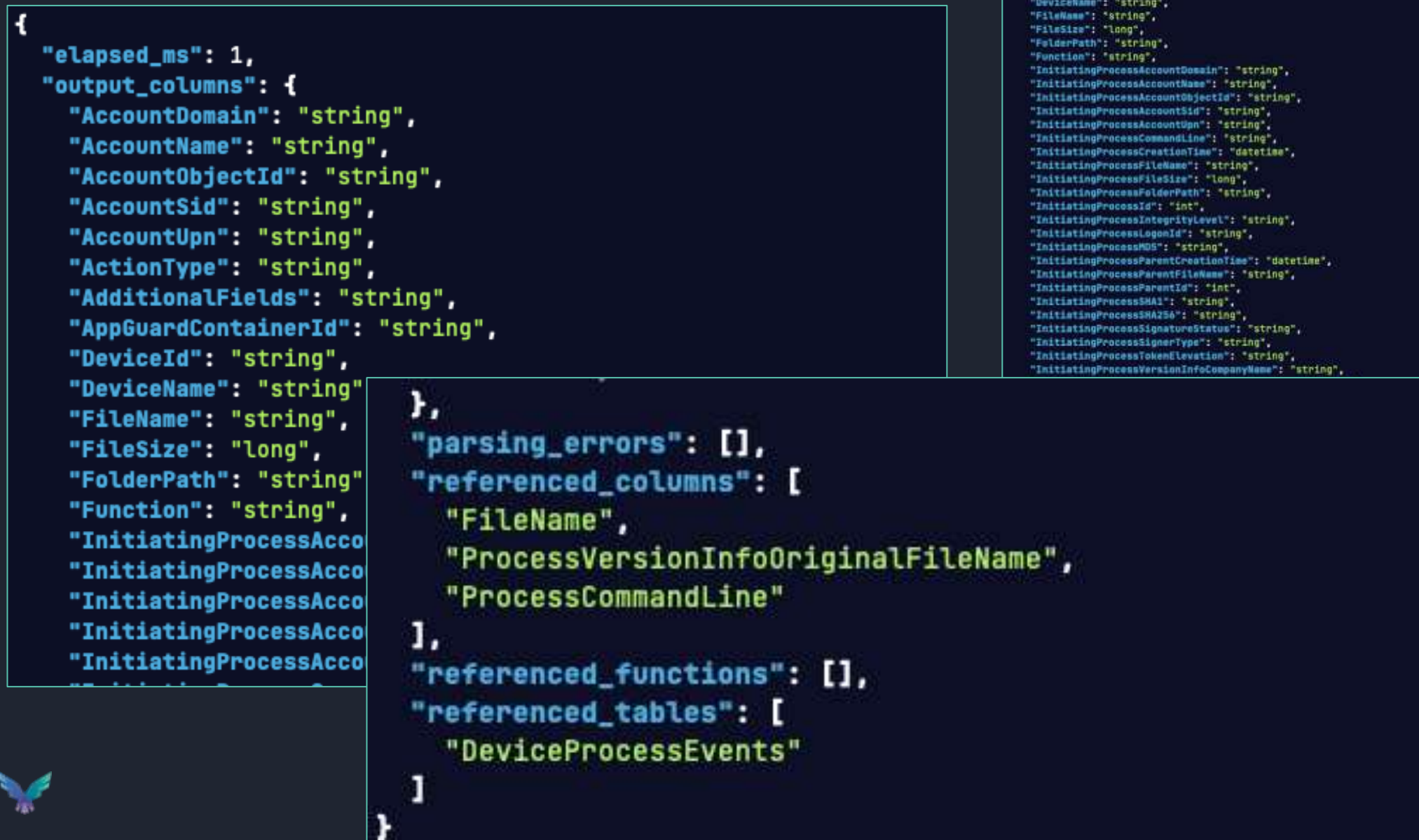
Define an ARM based JSON with your detection rule

```
olafhartong ~ ♥ 12:54 jq -S < 0034.json
{
  "environment": "m365",
  "query": "let timeframe = 2*1h;\nlet RuleId = '0034';\nDeviceProcessEvents\n| where ingestion_time() >= ago(timeframe)\n| where FileName =
~ \"rundll32.exe\" or ProcessVersionInfoOriginalFileName =~ \"rundll32.exe\"\n| extend Function=extract(@\"dll,\\s*(\\w+)\",1,ProcessCommand
Line)\n| where Function in~ (\"xlAutoOpen\", \"DllRegisterServer\", \"Uninstallfont\",\"DllUnregisterServer\",\"DispatchAPICall\",\"DllGetCl
assObject\",\"OpenURL\",\"TestRasm12\",\"UminslaIIF0mt\",\"UGCLQURCLQECD\",\"ShOpenVerbApplication\")\n// Begin client-specific filter.\n//
End client-specific filter.\n"
}
```

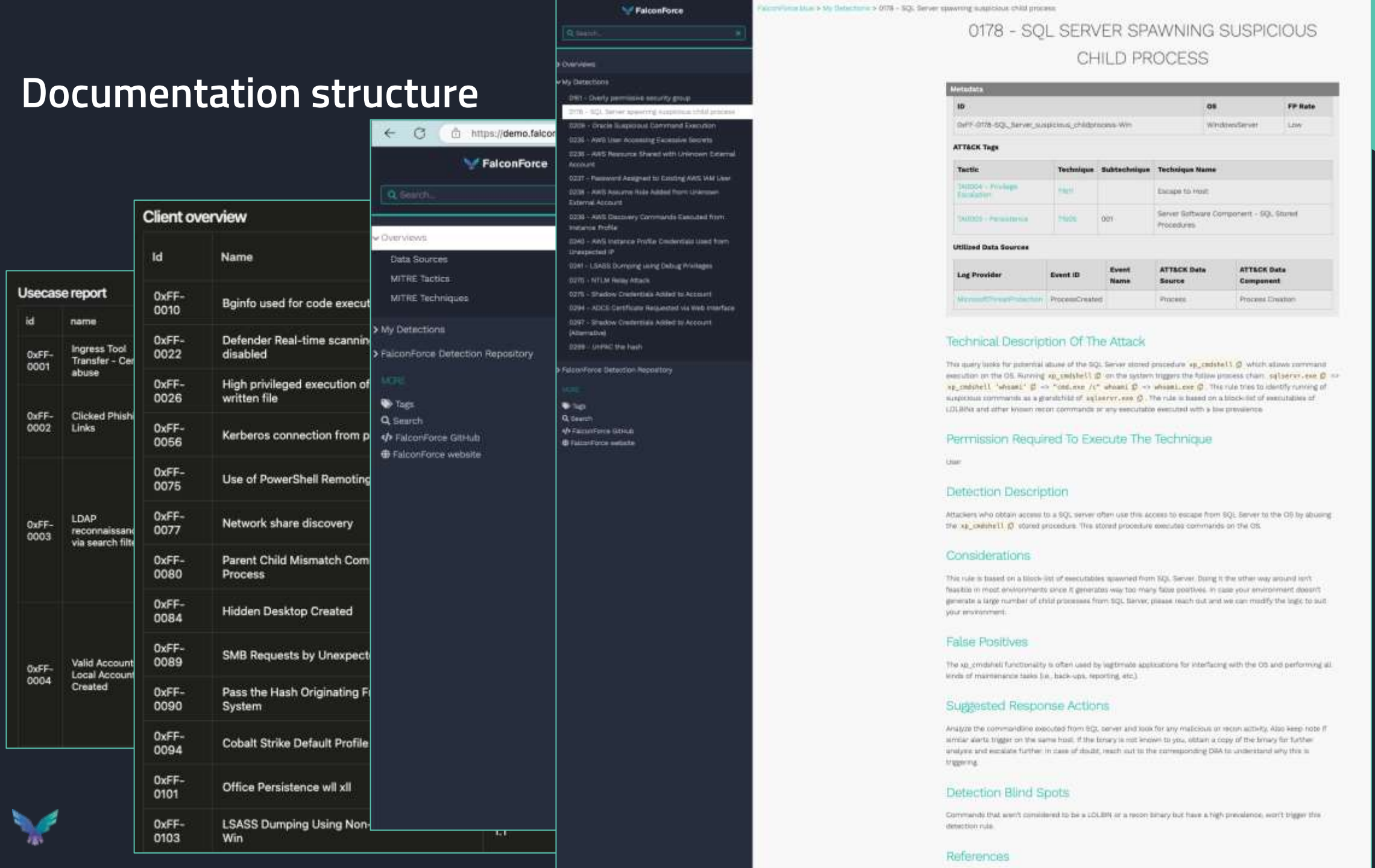
Feed it to the language server API

```
curl -H "Content-Type: application/json" -d@0034.json https://kql.falconforce.blue/api/kqlanalyzer -s | jq -S
```

KQL language server – result

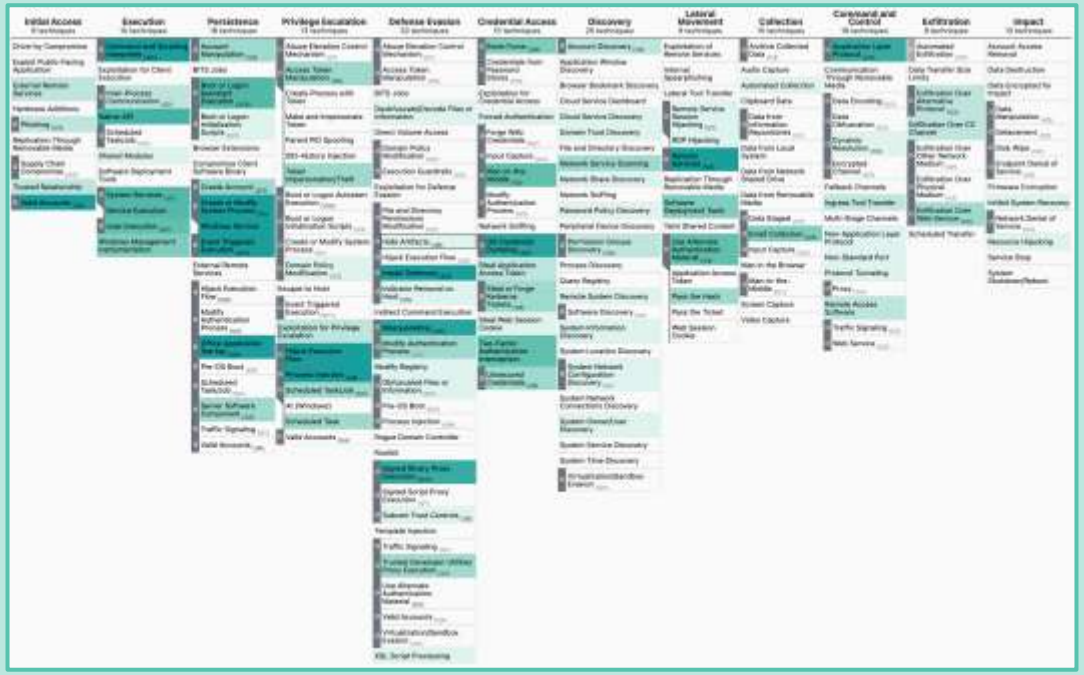


Documentation structure



MITRE ATT&CK coverage overviews

MITRE ATT&CK® is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. We use it as a foundation to connect the detections to real-world attack techniques.



Useful overviews

To provide an up-to-date overview of the detection library, per-customer or total overview layers for the MITRE ATT&CK Navigator can be generated based on the available content.

These layers can, for instance, be combined with other layers to show a comprehensive overview of a customer's detective and protective controls.

Unit testing

Use-case unit testing goals

End to end test of use-cases, testing from executing a real(alistic) attack, until the attack results in an alert.

By testing end-to-end, we validate all steps involved. For example, whether:

- The agent logs the expected events when the attack is performed
- The format of the logs is consistent
- Logs are properly ingested into Sentinel / MDE
- There is no out-of-the-box rule that makes ours redundant

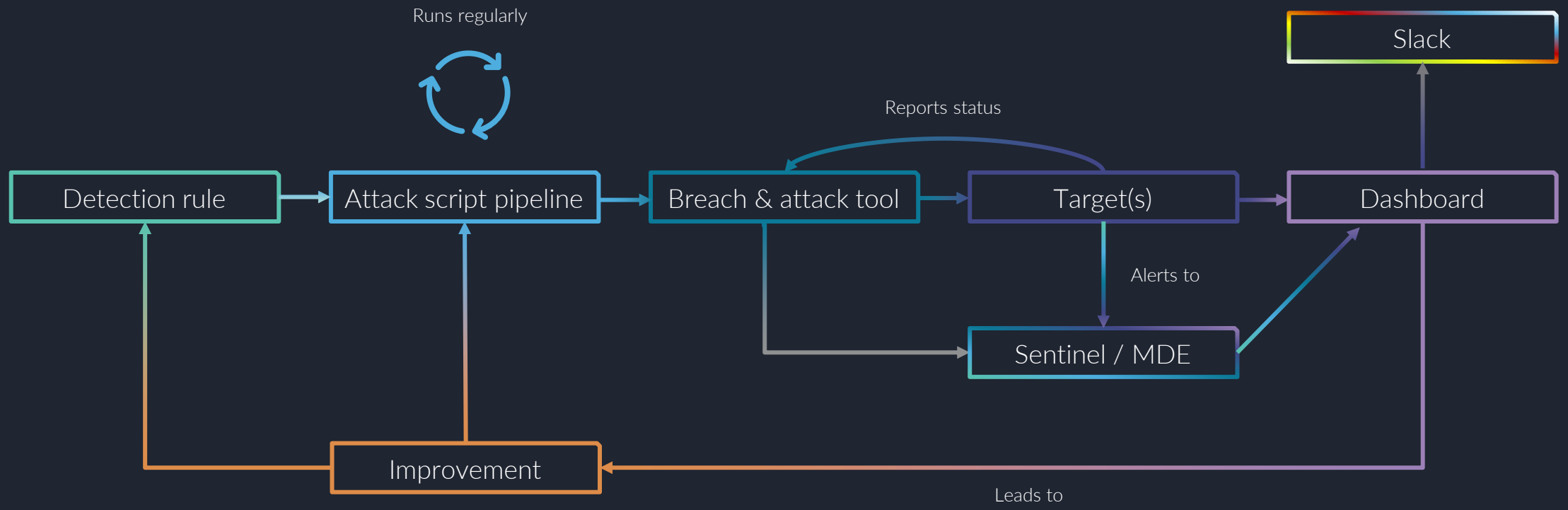
Ideally, each use-case has corresponding test case(s) that can trigger the use-case in an automated manner.

Test case design principles

Unit testing based on realistic attacks

- Where possible, perform an actual attack
- Test the attack was executed successfully
- Use variables for data that can differ per environment
- Focus on testing the EDR component, not the AV part
- Custom YAML format that is an extension of the format used by Atomic Red Team

Attack scripts logical flow



Test-case example (1/2)

```yaml
id: 0xFF-unit-0216-Phantom_DLL_Loaded-Win
description: |-
  Create a known phantom DLL (DLL that should no longer exist) and load it.
references: []
change_log:
  - {version: '1.0', date: '2021-12-02', impact: major, message: Initial version.}
expected_usecases:
  - platform: MDE
    type: hatchery
    id: 0xFF-0216-Phantom_DLL_Loaded-Win
required_variables:
  VAR_BIN:
    default: 'c:\temp\bin'
global_dependencies:
  - dll_hijacking_dll64.dll
```

Test-case example (2/2)

```yaml
execution:
  - platform: windows
    script: |-
      echo "Adding dll hijacking DLL."
      cp $ENV:VAR_BIN\dll_hijacking_dll64.dll
      $ENV:LOCALAPPDATA\Microsoft\WindowsApps\api-ms-win-core-kernel32-legacy-l1.dll -force
      -verbose
      rm -ErrorAction ignore -Force $ENV:TEMP\hellofalcon.txt

      sleep 1
      echo "Running cleanmanager that should load the hijacked DLL into dismhost."
      cleanmgr.exe /autoclean /d c:

      sleep 1
      if(Compare-Object (Get-Content $env:TEMP\hellofalcon.txt) "Hello Falcon") {
          echo "ERROR: Could not read the file."
          exit 1
      } else {
          echo "INFO: Operation completed successfully."
          exit 0
      }
    cleanup: |-
      echo "INFO: Removing artifacts."
      rm -ErrorAction ignore -Force
      $ENV:LOCALAPPDATA\Microsoft\WindowsApps\api-ms-win-core-kernel32-legacy-l1.dll
      rm -ErrorAction ignore -Force $ENV:TEMP\hellofalcon.txt
```

Dashboard – Test-case execution

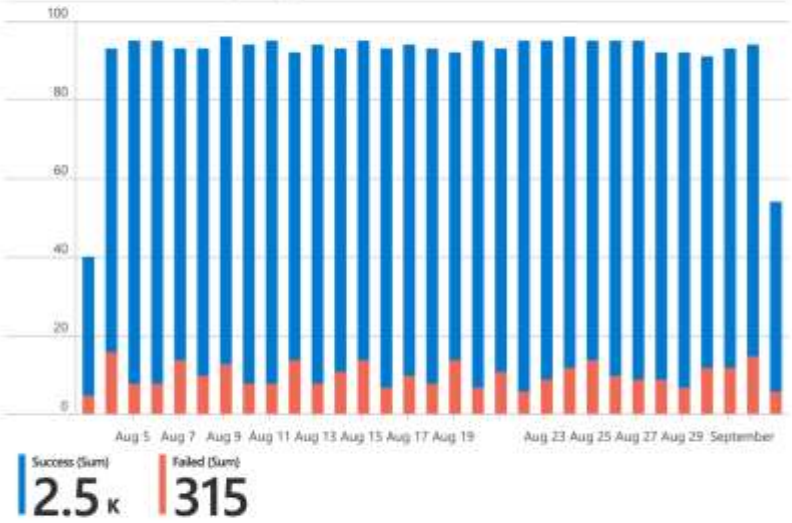
Test executions

All tests remain the same and will be executed every run

A successful result means the script or binary reported back to have run successfully

A failed result can indicate the EDR/AV or another mitigative measure blocked it or there was an additional error. Follow-up is required

Count of execution results per day



Test executions

TestName	SuccessRate	Success	Fails	ExecutionFrequency
0xFF-unit-0001-Ingress_Tool_Transfer_Certutil_abuse-Win	100	30	0	
0xFF-unit-0003-LDAP_reconnaissance_via_search_filters-...	0	0	42	
0xFF-unit-0004-Local_accounts_created-Win	100	37	0	
0xFF-unit-0010-Bginfo_used_for_code_execution	100	30	0	
0xFF-unit-0011-Code_execution_through_msxsl_and_wmi...	100	36	0	
0xFF-unit-0015-Masquerading-Renamed-executables-of-...	100	28	0	
0xFF-unit-0017-Admin_share_abuse_with_non_admin_acc...	100	34	0	
0xFF-unit-0019-Boot_or_Logon_Autostart_Execution_Winl...	100	24	0	
0xFF-unit-0020-Child_of_MSBuild_calling_external_system...	100	42	0	
0xFF-unit-0021-Cross_account_SMB_connections-Win	100	32	0	
0xFF-unit-0022-Defender_Real_time_scanning_has_been_...	100	34	0	
0xFF-unit-0023-Dumping_of_password_with_Esentutl_exe...	100	31	0	
0xFF-unit-0024-Event_Triggered_Execution_AppInit_DLLs-...	100	30	0	
0xFF-unit-0025-Event_Triggered_Execution_Image_File_Ex...	100	31	0	
0xFF-unit-0026-High_privileged_execution_of_low_privile...	0	0	29	

Dashboard – Test-case analysis

Detections

Timechart count activitytypes by 4 hours

Failed (Sum) 75

Succeeded (Sum) 271

Detection successrate

TestName	DetectionRate	Detected	Not Detected
0xFF-unit-0001-Ingress_Tool_Transfer_Certutil_abuse-Win	100	30	0
0xFF-unit-0003-LDAP_reconnaissance_via_search_filters-Win	0	0	42
0xFF-unit-0004-Local_accounts_created-Win	100	37	0
0xFF-unit-0010-Bginfo_used_for_code_execution	96	29	1
0xFF-unit-0011-Code_execution_through_msxsl_and_wmic-Win	91	33	3
0xFF-unit-0017-Admin_share_abuse_with_non_admin_account-Win	91	31	3
0xFF-unit-0020-Child_of_MSBuild_calling_external_system-Win	42	18	24
0xFF-unit-0022-Defender_Real_time_scanning_has_been_disabled-Win	97	33	1
0xFF-unit-0023-Dumping_of_password_with_Esentutl_exe-Win	100	31	0
0xFF-unit-0024-Event_Triggered_Execution_AppInit_DLLs-Win	96	29	1

Wrapping up - Detection as code

- Provides quality control by automation and review
- Ensures a single source of truth
- Allows for automated deployment, even across multiple environments
- Self-documenting
- Measure operating quality through automated validation testing.

Thank you!

Meet us at the FalconForce booth!

olaf@falconforce.nl

https://falconforce.nl

@olafhartong
@falconforceteam

https://linkedin.com/in/olafhartong
https://linkedin.com/company/falconforce