CoreTIDE

# OpenTIDE

*Threat-informed Detection Engineering management framework*

# When TI made actionable drives your Threat Detection


Amine BESSON
*(Behemoth Cyberdefence, NL)*

Remi SEGUY
*(EC DIGIT S2, LU)*

Claus HOUMANN
*(EC DIGIT S1, LU)*




European Commission

# Core Concepts

## Threat + Detection Modelling

- Structures how intelligence and other input should be processed
- Creates an ever-growing knowledge graph to support decision making
- Clarity to detection posture and prioritization
- As-code, YAML, Schema, LLM-ready

## Detection-as-Code

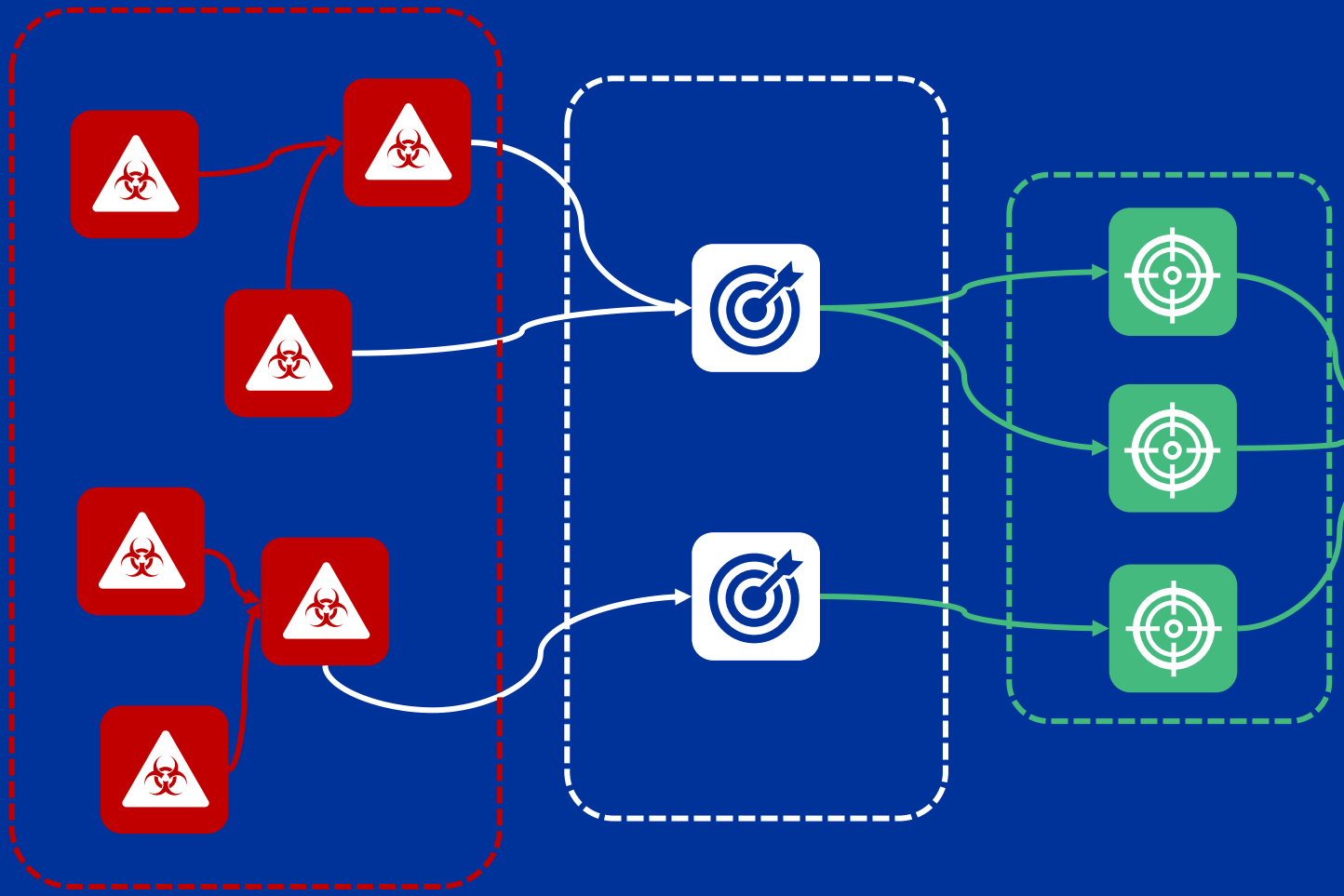
- Multi-system state of the art framework
- Multi-tenant ready
- Dynamic modifiers
- Staging and Production Workflows
- Powerful schemas and tooling
- Near or Full GUI Parity

Common DevOps Workflows, Threat Driven, Repeatable, Measurable

# OpenTide Objects and Graph

Standard Schemas and CI/CD Automation

Unstructured Threat Intelligence, Red Teaming Reports, Lessons Learned, Risk Analysis…



*Atomic Threat Vector Graph*

*Detection Objectives*

*Detection Rules*

splunk> Microsoft Defender SentinelOne CROWDSTRIKE VMware Carbon Black™

*Platform Deployment (extensible plugin framework)*

*PUSH only via API*

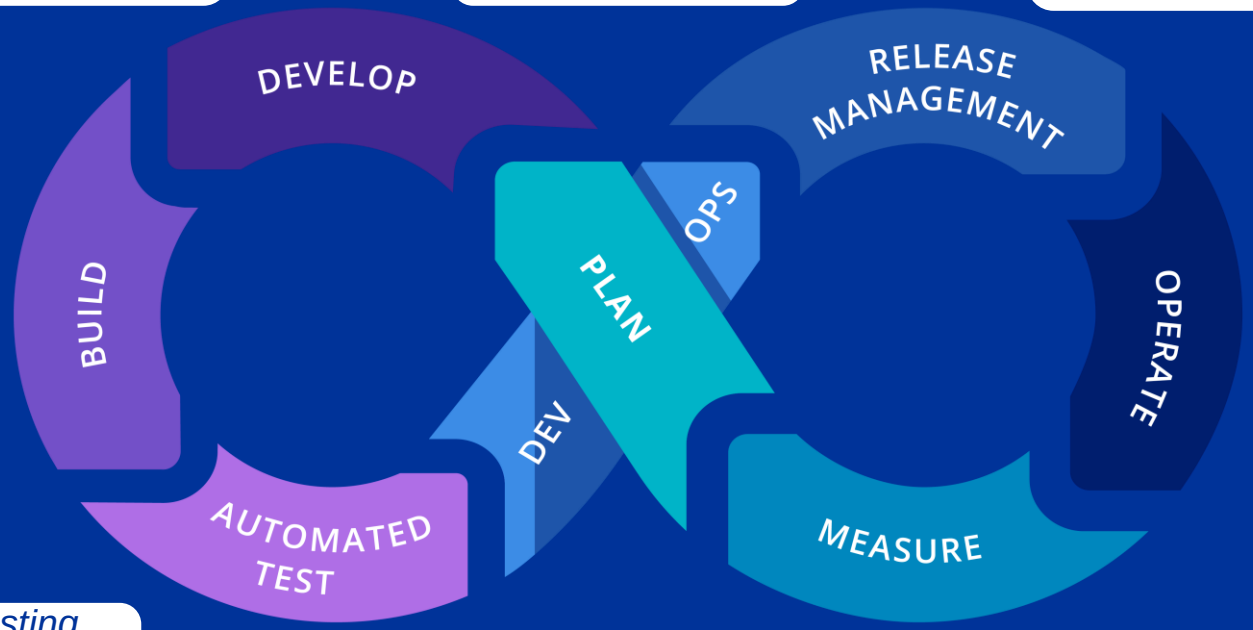
# DetectionOps

Thesis : we can map out all Detection Engineering tasks on a DevOps-like loop, and leverage existing tools, know-how and approaches across all stages – but we need a framework.
**OpenTide supports the end-to-end Detection lifecycle by structuring the DetectionOps workflow**

*Schema & pipeline validation, IDE support*

*Threat-Driven Planning*

*Production deployment process, approvals, documentation automation*




*As-code deployments*

*Feedback loop from Incident Response*

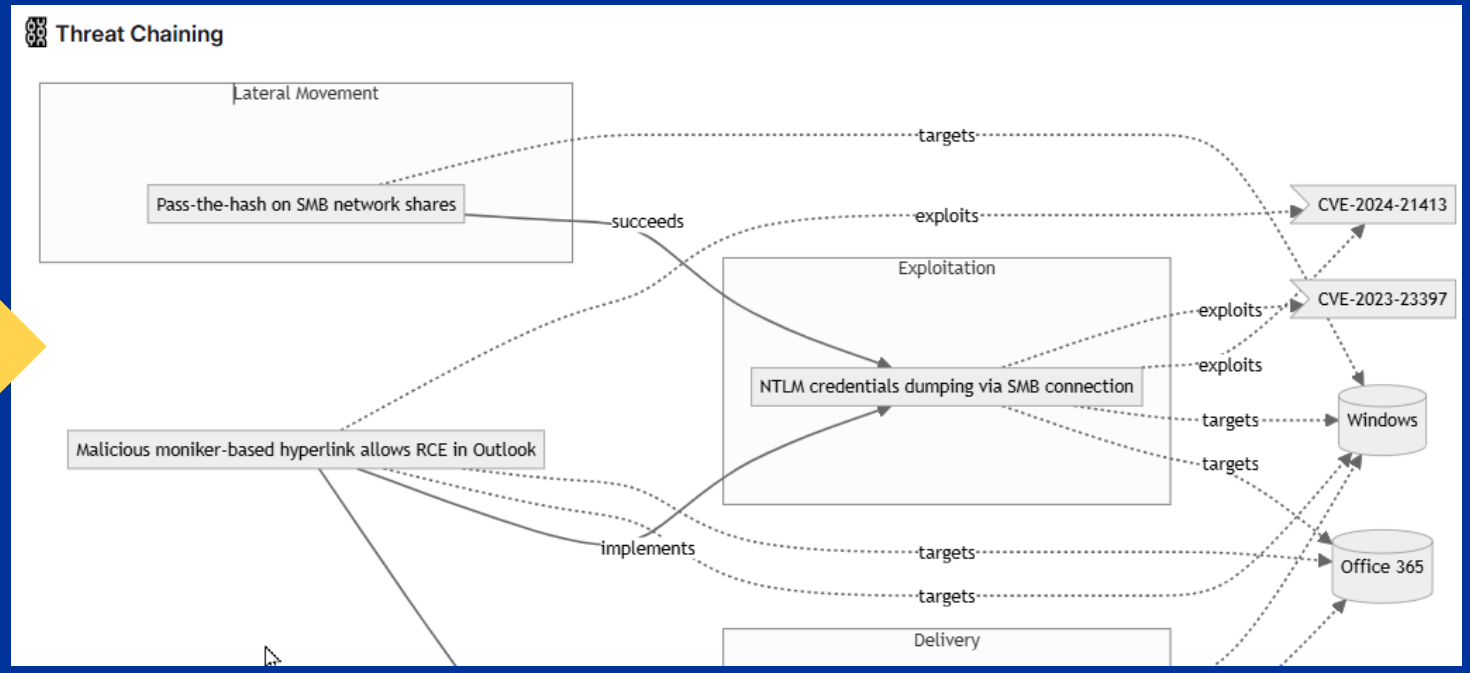
*Detection Unit Testing, False Positive evaluation…*

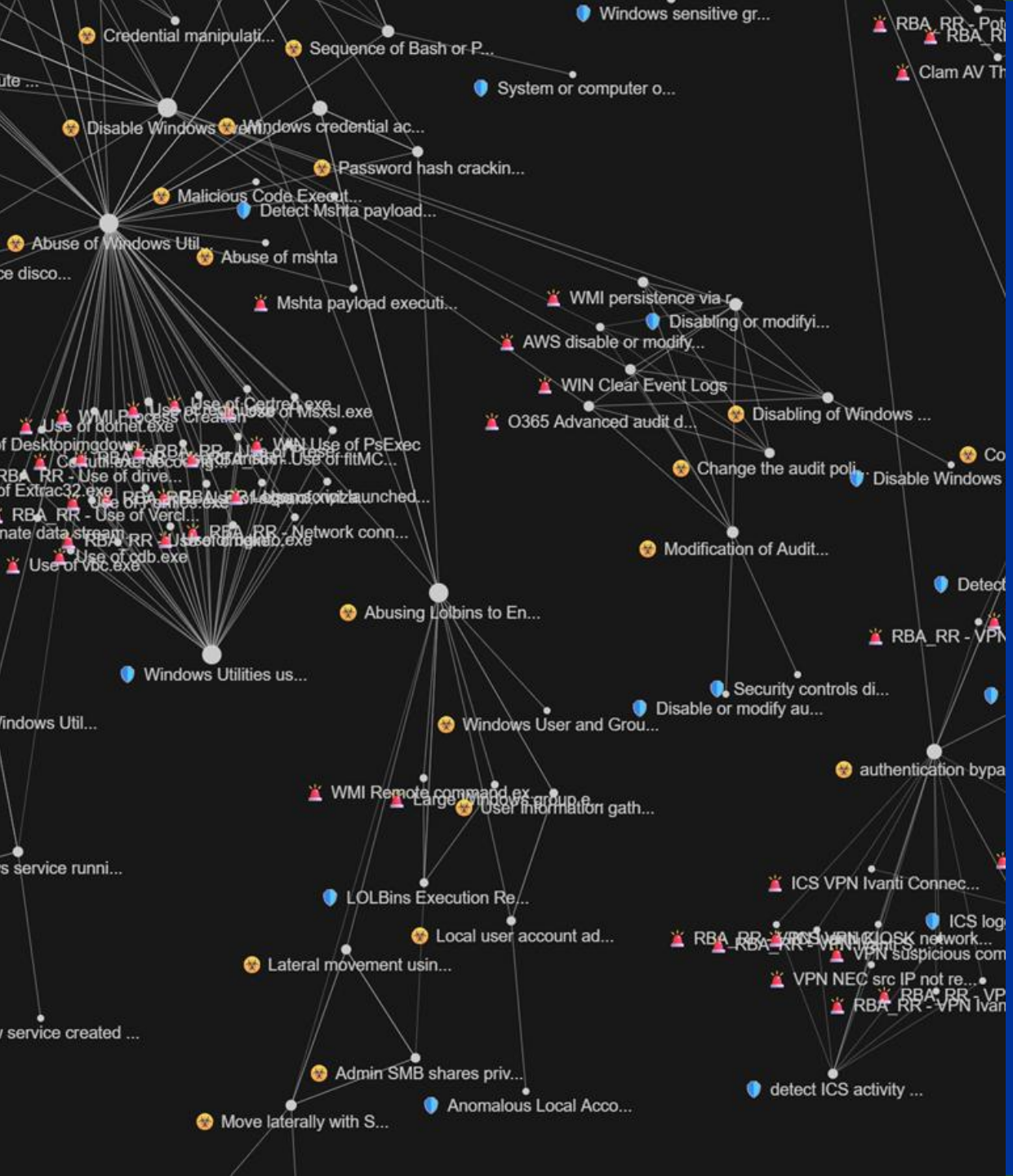
*False Positives tracking and detection coverage analysis*

# Threat Vector Chaining =

## build dynamic attack path
## without knowing the full shape of the graph

```
att&ck:
  - T1566.002 #Phishing: Spearphishing Link
  - T1189     #Drive-by Compromise
  - T1557.001 #Adversary-in-the-Middle: LLMNR/NBT-NS Poisoning and SMB Re
  - T1204.002 #User Execution: Malicious File
chaining:
  - relation: sequence::succeeds
    vector: 1a68b5eb-0112-424d-a21f-88dda0b6b8df # Spearphishing Link
    description: |
      Threat actor uses crafted e-mails with embedded hyperlinks
      in attempt to trick the user to click on them and grant
      access bypassing the protected view mechanism of
      Microsoft Outlook.
  - relation: atomicity::implements
    vector: 02311e3e-b7b8-4369-9e1e-74c0a844ae0f # NTLM credentials dumpi
    description: |
      In an initially prepared campaign, a threat actor can steal NTLM
      hashes, using Moniker-based hyperlink which points to SMB share.
cve:
  - CVE-2024-21413 # Successful exploitation of this vulnerability would
```






# Detection Coverage Knowledge Graph

- Next-generation detection coverage evaluation
- Move away from static “taxonomies” and move at the speed of threats
- Full of metadata to slice and dice – e.g. what is my coverage for Evasion type of TTPs in Azure ?
- The Detection Engineering backlog is now a constant, continuously changing picture but accurate against threats we know of

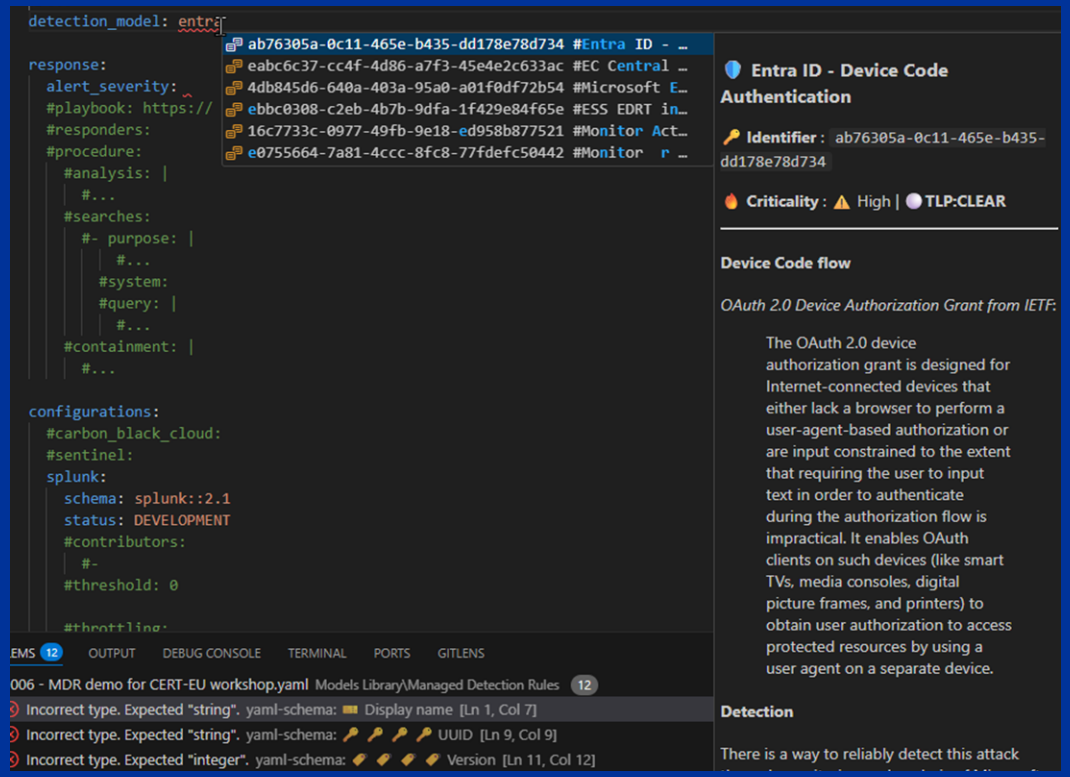

# What about ATT&CK then ?

- ATT&CK is still a great way to identify if there's any gaps we don't know about
- All Threat Vector Objects map on ATT&CK
- By extension, every related artefact (down to the rule) has a connection to ATT&CK
- OpenTide leverages always the latest version of ATT&CK, and updates its dependencies frequently (no deprecated versions like in other tools or projects)

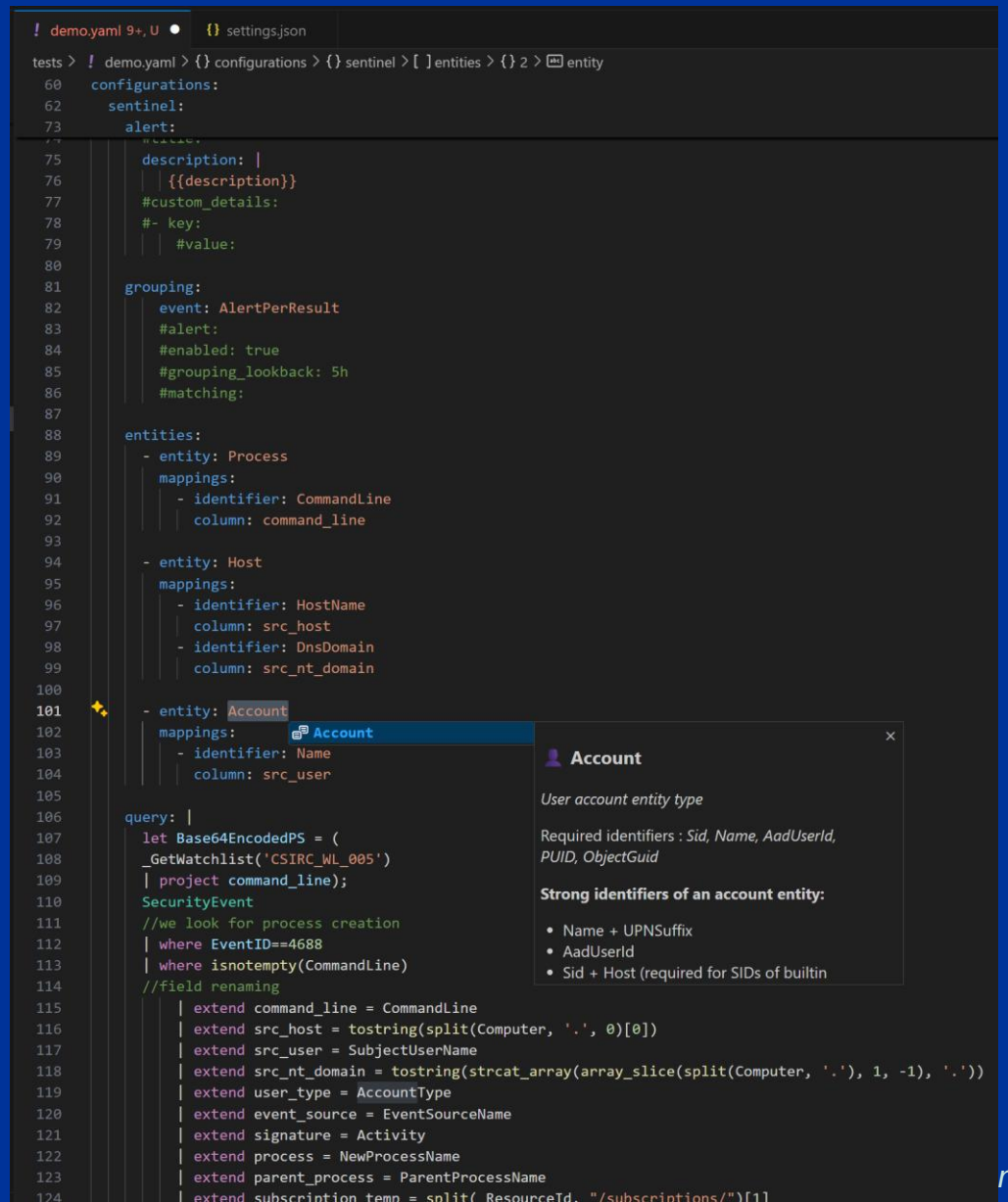
# Best in class tooling to draft YAML objects

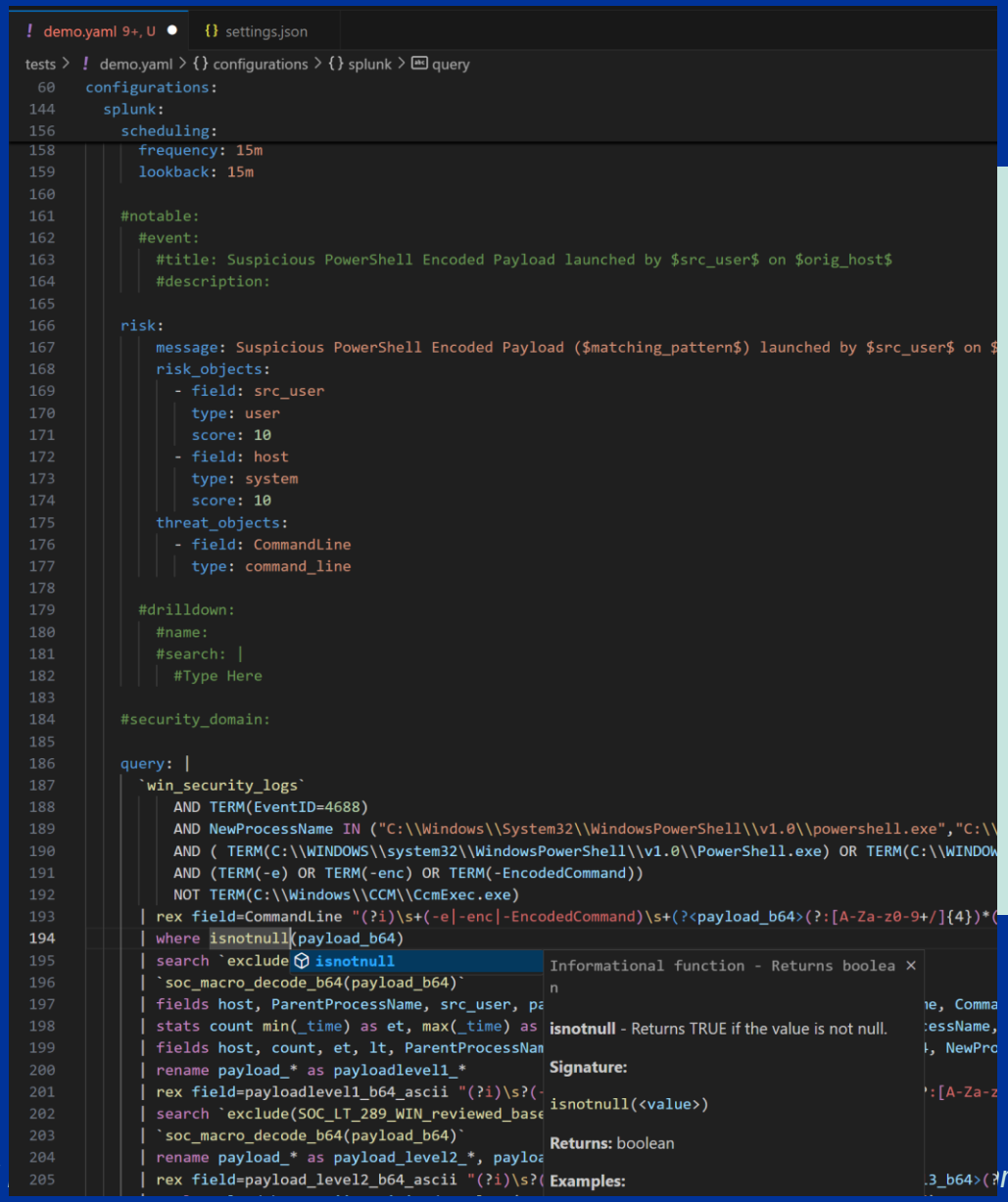
OpenTide goes very far with pushing JSON Schemas to create an IDE-style experience

- Auto-completion
- Continuous validation
- Self-indexation to reference other objects as they get merged
- Templates and snippets that evolve with schema evolution
- For users : once familiar, from branch to PR in less than a minute

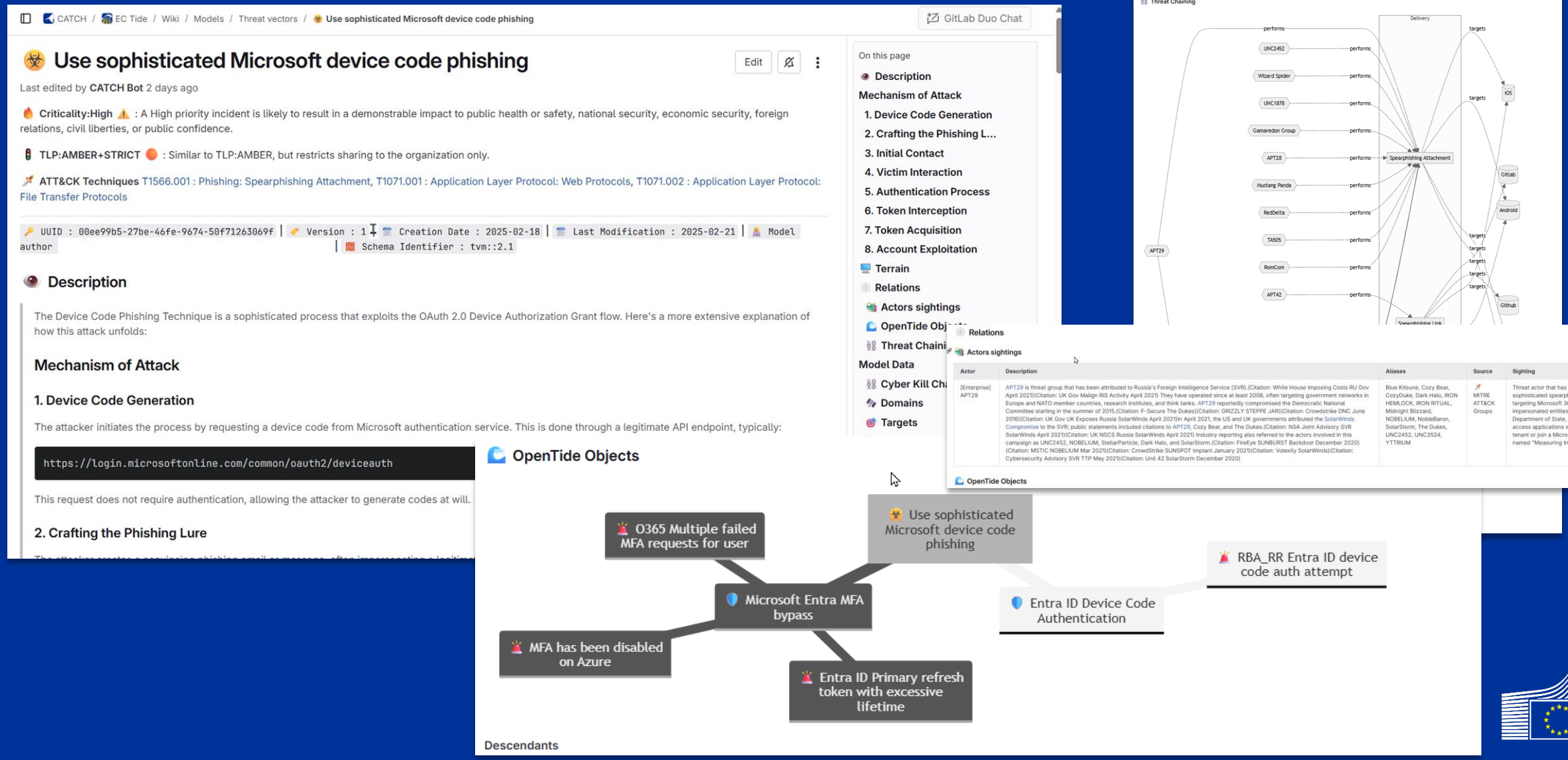


# Drafting Managed Detection rules example Sentinel & Splunk sub-sections



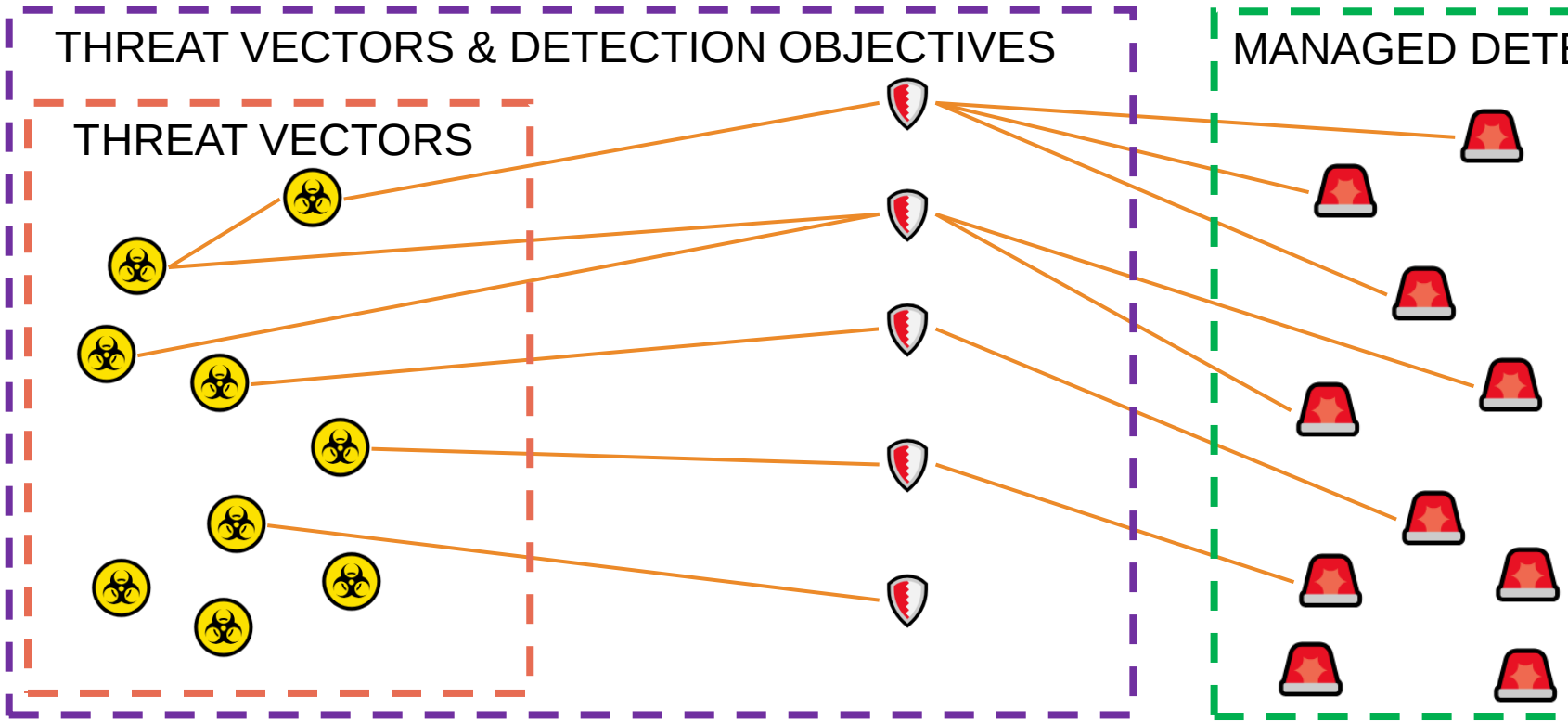
# Nobody wants to write documentation



CoreTIDE

# Use cases for OpenTIDE

## OpenTIDE




435 Threat Vectors  215 Detection Models  475 Active Detection Rules

European Commission

# OpenTide key points

- Managed Knowledge **Graph**
- Rich documentation including metadata
  - Linking threats to detection scenarios to deployed rules
  - Consistency and quality of documentation, enforced via validation
- Detection “as-code” – Automation
  - YAML objects structured and validated against a schema
  - Prioritisation of the backlog
  - Versioning, history, BCP (rules can be rolled out in short time)

CoreTIDE

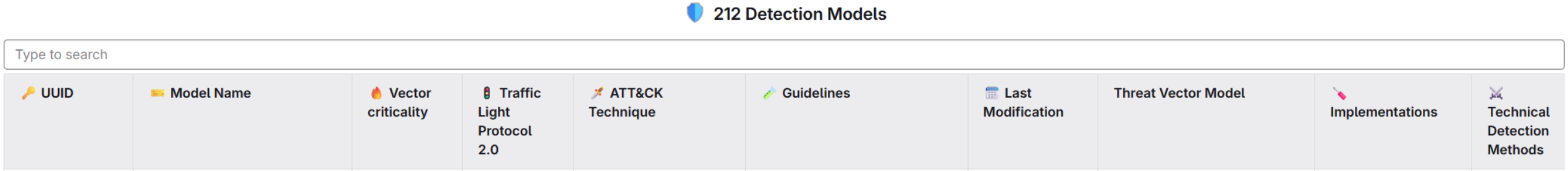
# Threat Vector object key points

- Turn TI inputs into actionable objects (purple teaming)
- Allow to document the attack vectors and attack paths
  - Threat vector relevant for the organisation
  - Granularity can go as deep as needed below the ATT&CK sub-techniques
- Prioritise the backlog of threat to be covered


European Commission

# Detection Objective object key points

- Define clearly WHAT to detect
  - Linked to threat vectors
  - Document the detection strategies and which data sources are needed
  - Documentation of detection use cases that didn't work out
- Facilitate communication with system owners and risk managers
  - Detection objectives are descriptive vs rules that are platform specific code
- Drive the log collection to support the objective (visibility)
  - List the data sources required to implement the detection strategies



# Managed Detection Rule object key points

- DE focuses on the quality of the query
- All the rest is automated within EC-Tide
  (no need to configure via GUI - 100+ clicks/copy/paste replaced by a YAML file)
- Peer review at the lines of code level
- **Joint workflow btw DE and SOC to review the rule & draft alert handling guidelines**
  - MDR contains the response section providing guidelines on handling the alert.

# OpenTIDEhq Project Structure

Decoupled code backend, made to run without maintenance on every pipeline run

**CoreTide**

**InitTide**

Starting point template to provide the folder structure, minimal set of files, and CI to connect to CoreTide

Copy

Pull from public

**Private clone of CoreTide**

Point to your local clone of CoreTide

**Your private Git content repo**

Review and copy

TLP:CLEAR content repository
YAML opentide objects

200+ Threat Vectors shared
More to come

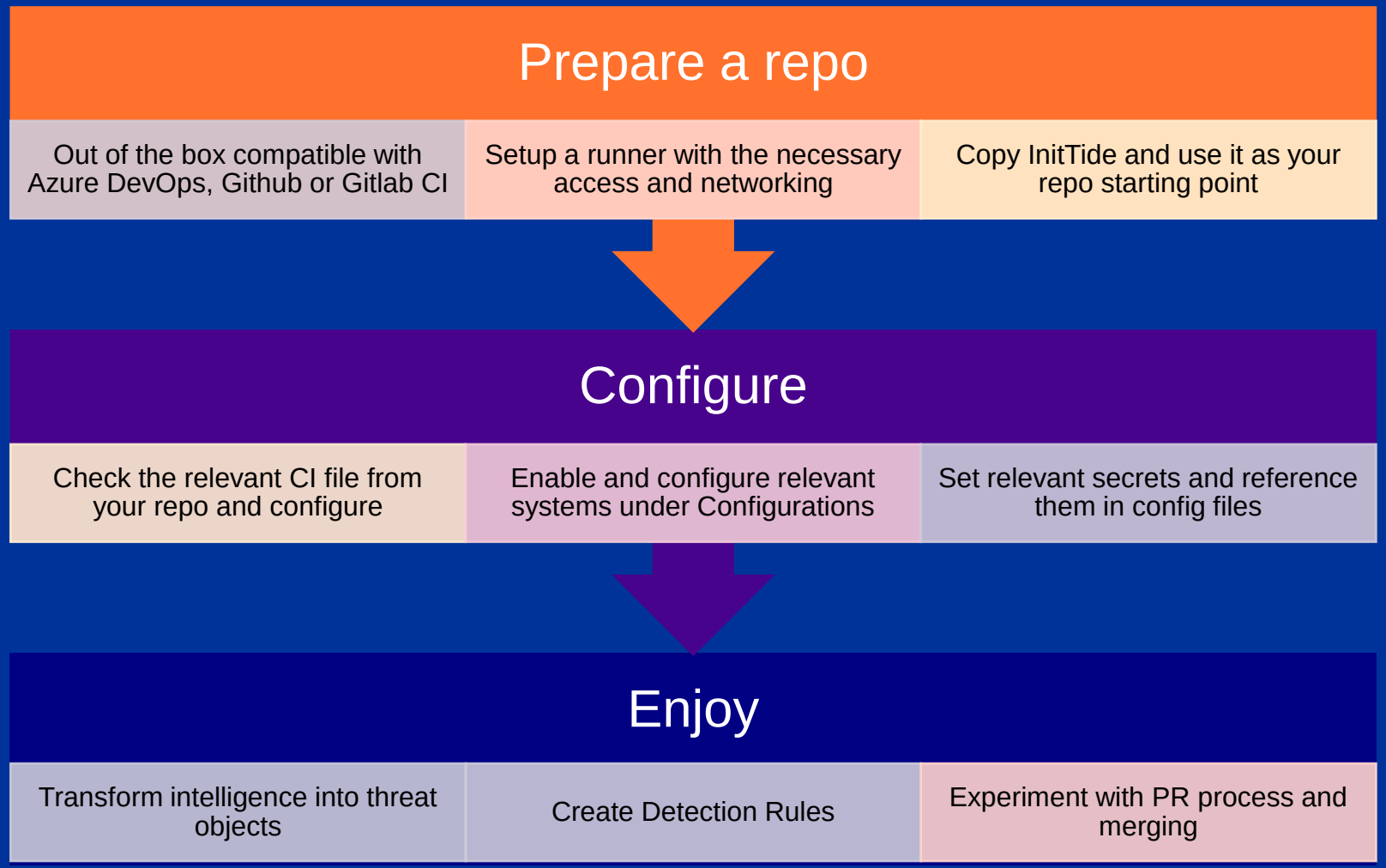
**ShareTide**

**WikiTide**

TLP:CLEAR Wiki content
Markdown documentation
of
Objects published under
ShareTIDE

*Disclaimer: The views expressed are solely those of the writers and may not be regarded as stating an official position of the European Commission*

# Give it a shot



# With OpenTIDE do

- Detection Engineering
  - Include TI context into your detection rules
  - Prioritise and measure your threat detection coverage
- DetectionOps
  - Backlog, iterations, issues, merge requests, review,
- Detection-as-Code
  - YAML, versioning, CI/CD pipelines, deployers, validation of queries

# OpenTide

- OpenTide project
  - https://github.com/OpenTideHQ
  - https://code.europa.eu/ec-digit-s2/opentide

Please open issues or pull requests here !

- Mailbox: DIGIT-EC-OPENTIDE@ec.europa.eu

- FIRST SIG DE&TH Detection Engineering & Threat Hunting

# Thank you!