# Indicators of Behavior Workshop

## TIPS 2025

Charlie Frick, Johns Hopkins Applied Physics Lab,
OCA IOB Sub-Project Chair
Charles.Frick@jhuapl.edu

# What are IOBs?

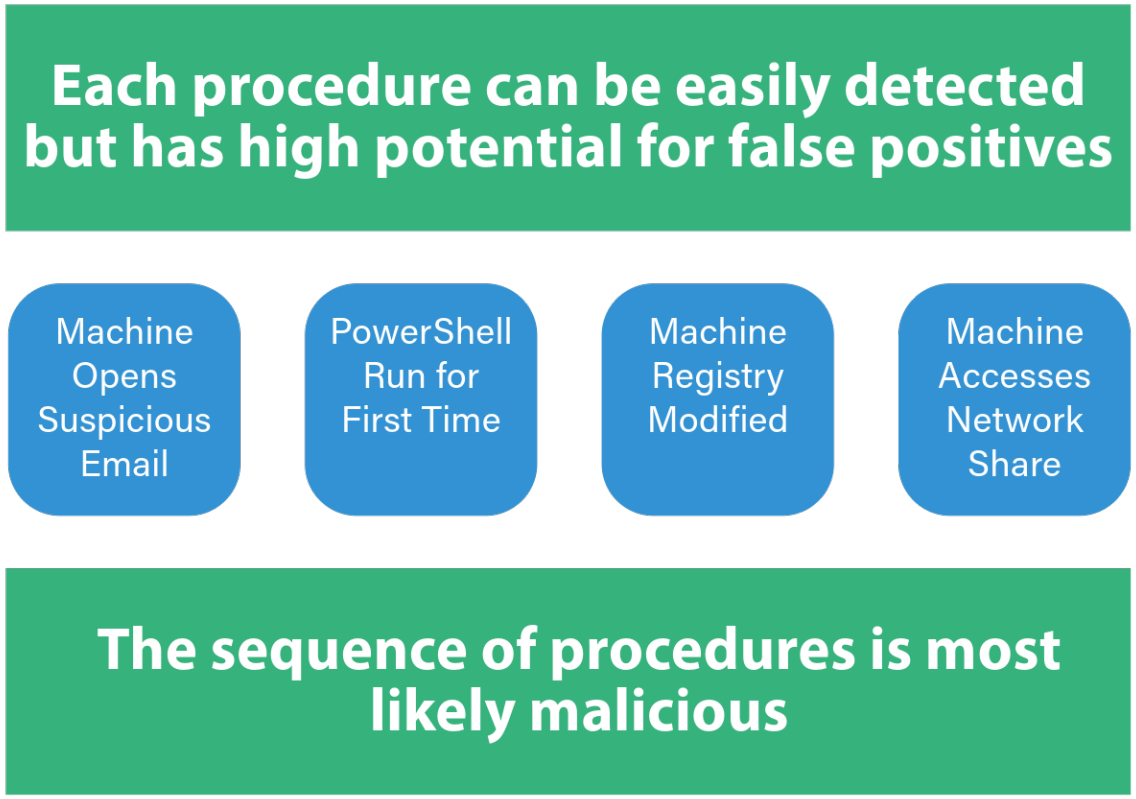
Full talk on Indicator of Behavior features during TIPS technical talks later this week.

# Overview

- In 2021, CISA and JHU/APL began a partnership with the Open Cybersecurity Alliance in their creation of the Indicators of Behavior (IOB) Sub-Project
- This workshop will provide an introduction to IOBs and demonstrate the process of creating Indicators of Behavior from Threat Intelligence
- Links for access to the reference implementation and additional resources are included in this presentation
  - Active collaboration on this research is welcome through the IOB Sub-Project (https://opencybersecurityalliance.org/iob/)

# Indicator of Behavior Concept

- Indicator of Behavior (IOB) STIX bundles provide repeatable sets of observed adversary behaviors to help defender tools & capabilities
  - Intelligence context provided in machine-readable graph representation
  - Relationships to relevant ATT&CK attack pattern objects
  - Relationships to detection analytics
  - Includes correlation workflows to address false positives
  - Includes response Courses of Action (COAs) and cybersecurity operations playbooks in standardized formats



# IOBs vs IOCs

- Indicators of Compromise (IOCs)
  - Static Artifacts: Represent known bad elements like file hashes, IP addresses, or domain names
  - Reactive: Useful after an attack has occurred
  - High Maintenance: Require constant updating and often have short shelf life
  - Low Context: Limited insight into attacker intent or methodology
- Indicators of Behavior (IOBs)
  - Dynamic Patterns: Capture sequences of adversary actions and tactics (e.g., "Receive phish → Execute macro → Beacon DNS")
  - Proactive: Enable detection of novel or evolving threats before known IOCs are available
  - Contextual & Correlatable: Include detection logic, alert correlation, and courses of action
  - Machine-Readable & Shareable: Designed in STIX format with links to MITRE ATT&CK, Sigma rules, and open automated playbooks

IOBs bridge the gap between low-level forensic artifacts and high-level attack patterns to power smarter, earlier detection and response.

# What is STIX?

- **S**tructured **T**hreat **I**nformation e**X**change
- Standard language and framework for describing and exchanging cyber threat intelligence (CTI)
- Managed by the Cyber Threat Intelligence Technical Committee (CTI TC) of the OASIS (Organization for the Advancement of Structured Information Standards) consortium
- JSON representation of CTI to enable machine readability and sharing of knowledge graphs
- Often sent via TAXII standard protocol also managed by OASIS CTI TC

For more information:
https://oasis-open.github.io/cti-documentation/

IOB Stix Extension Schema: https://github.com/opencybersecurityalliance/stix-extensions/tree/main/2.x/SDO
Documentation: https://github.com/opencybersecurityalliance/stix-extensions/tree/main/contexts/indicator-of-behavior

# STIX2NEO4J Script

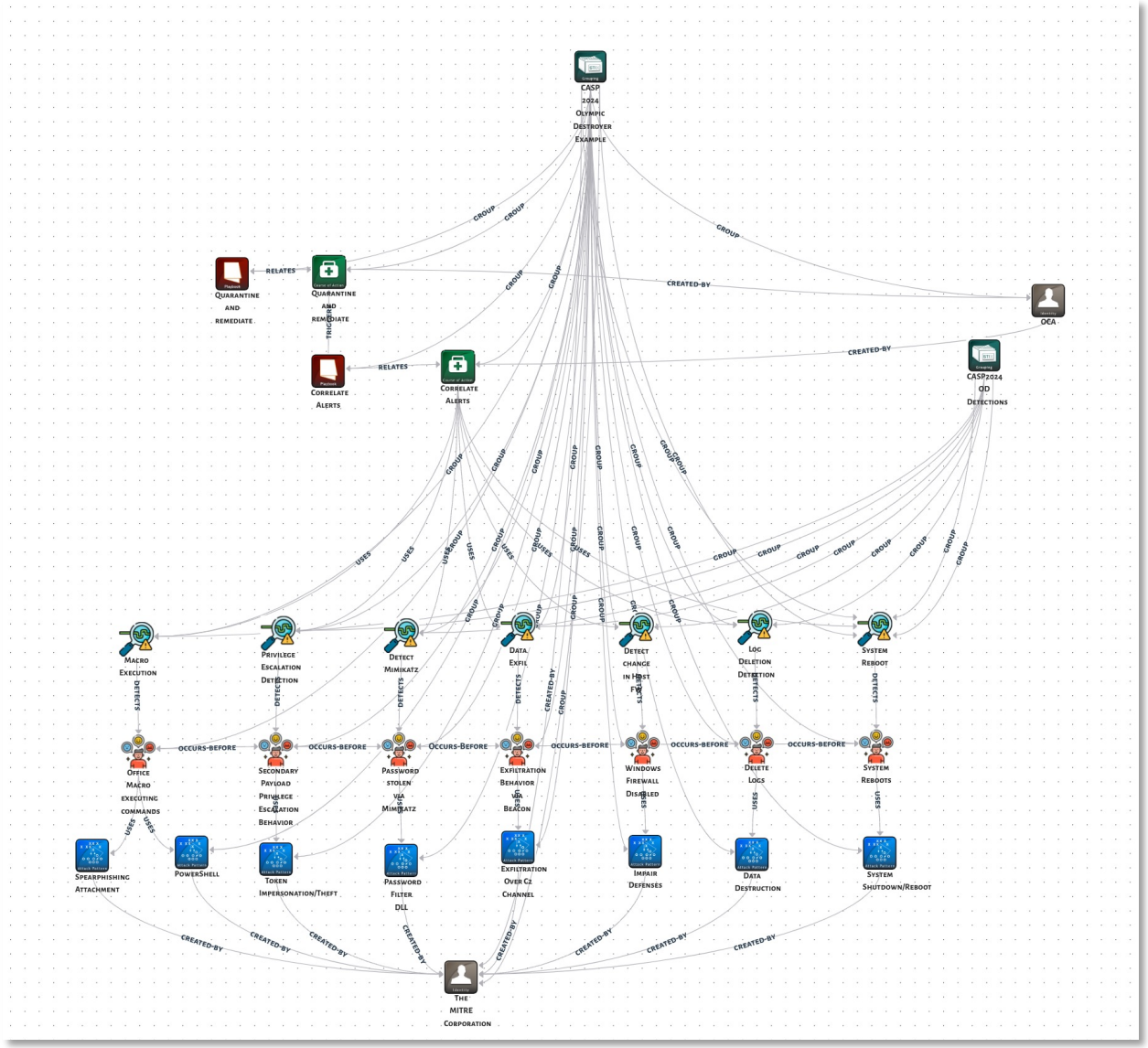
- Python script for analyzing STIX 2.x bundles in a neo4j graph database
- Provides additional analytical capabilities for investigating raw STIX messages without major modification of the data
  - Threat Intel Platforms often make significant changes to data model upon import
- Released under an Apache2 license through the Open Cybersecurity Alliance Indicator of Behavior Sub-Project
- Script repository link on GitHub:
  - https://github.com/opencybersecurityalliance/oca-iob/tree/main/STIX2NEO4J%20Converter

SCAN ME

INDICATORS OF BEHAVIOR

# STIX Modeler

- Refinement of Open Source STIX-Modeler Project on GitHub
  - Code available at: https://github.com/JHUAPL/STIXMODELER_UI
  - Fork of original Modeler project https://github.com/STIX-Modeler/UI
- GUI-based editor for creating STIX without coding
- Modernized code dependencies and visualization framework
- Created support for STIX extensions and custom STIX objects and relationships



# Identifying and Extracting IoBs

# Scenario: “Spark Rainstorm”


- Simulated cyber threat campaign used for this workshop
- Models advanced adversary behavior using fake APT group "RainDrift"
- Demonstrates tactics such as: Malware staging (meterpreter), Internal reconnaissance (PowerView), Credential theft (ntdsutil)
- Relevant Attack Patterns
  - T1608.001 – Upload Malware
  - T1059.001 – PowerShell Execution
  - T1555 – Credentials from Password Stores

# Identify Behaviors (Step 1 of 4)

- Start by extracting the known Attack Patterns and sequence from the report
  - Attack Flow demo will provide a useful example of how to do this later
- Focus on patterns that leave observable traces
  - For example, password cracking, while a key element of the campaign, cannot be detected.

| Attack Pattern | Behavior |
|---|---|
| T1608.001 – Upload Malware | |
| T1059.001 – PowerShell Execution | |
| T1555 – Credentials from Password Stores | |

# Identify Behaviors (Step 2 of 4)

- For creating Behaviors, some key steps
  - Focus on **WHAT** the adversary wants to do, not **HOW** they do it
  - We want things that the adversary cannot change without modifying their actual behavior, i.e. that is the task they want to do
  - We also want things that are observable from the victim network and endpoint machine behaviors

| Attack Pattern | Behavior |
|---|---|
| T1608.001 – Upload Malware | |
| T1059.001 – PowerShell Execution | |
| T1555 – Credentials from Password Stores | |

# Identify Behaviors (Step 3 of 4)

- For creating Behaviors, some key steps
  - Focus on **WHAT** the adversary wants to do, not **HOW** they do it
  - We want things that the adversary cannot change without modifying their actual behavior, that is the task they want to do
  - We also want things that are observable from the victim network and endpoint machine behaviors
- Do not worry about false positives yet…

| Attack Pattern | Behavior |
|---|---|
| T1608.001 – Upload Malware | An endpoint system is executing code from an “abnormal” location or process |
| T1059.001 – PowerShell Execution | PowerView commands were used to conduct reconnaissance |
| T1555 – Credentials from Password Stores | The ntds.dit database was copied from a domain controller |

# Identify Behaviors (Step 4 of 4)

- For creating Behaviors, some key steps
  - Focus on **WHAT** the adversary wants to do, not **HOW** they do it
  - We want things that the adversary cannot change without modifying their actual behavior, that is the task they want to do
  - We also want things that are observable from the victim network and endpoint machine behaviors
- Do not worry about false positives yet…

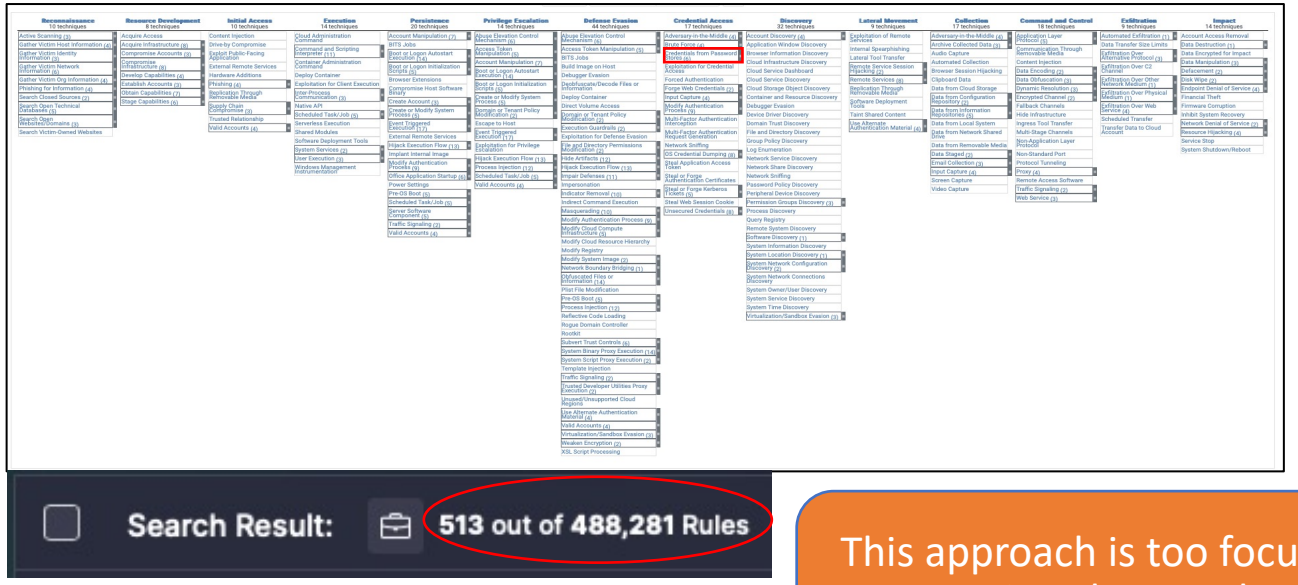
| Attack Pattern | Behavior |
|---|---|
| T1608.001 – Upload Malware | An endpoint system is executing code from an ”abnormal” location or process |
| T1059.001 – PowerShell Execution | PowerView commands were used to conduct reconnaissance |
| T1555 – Credentials from Password Stores | The ntds.dit database was copied from a domain controller |

Now, Let’s Discuss Detections

# Designing a Behavior Based Detection

## The challenge of only using ATT&CK

- Let's use an example from the Spark Rainstorm Campaign
- These adversaries often steal credentials through living off the land techniques such as accessing windows NTDS databases
- Threat reports tell us this is ATT&CK T1555
- Searching for detection analytics from that Attack Pattern alone yields hundreds of potential rules
- Most of these will not work in many environments because they are too specific




This approach is too focused on HOW the attacker conducted their TTPs

Example

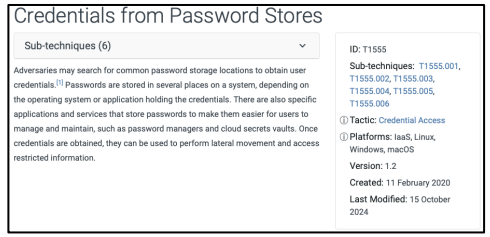
```
any where (winlog.channel : "Security" and (((winlog.event_id : "4663" and winlog.event_data.ObjectType : "File" and winlog.event_data.AccessMask : "0x1") and (winlog.event_data.ObjectName : ("*\\User Data\\Default\\Login Data*", "*\\User Data\\Local State*", "*\\User Data\\Default\\Network\\Cookies*") or file.path : ("*\\cookies.sqlite", "*\\places.sqlite", "*release\\key3.db", "*release\\key4.db", "*release\\logins.json"))) and  not ((winlog.event_data.ProcessName : "System" or process.executable : "System") or (winlog.event_data.ProcessName : ("C:\\Program Files (x86)\\*", "C:\\Program Files\\*", "C:\\Windows\\system32\\*", "C:\\Windows\\SysWOW64\\*") or process.executable : ("C:\\Program Files (x86)\\*", "C:\\Program Files\\*", "C:\\Windows\\system32\\*", "C:\\Windows\\SysWOW64\\*")))) and  not ((winlog.event_data.ProcessName : "C:\\ProgramData\\Microsoft\\Windows Defender\\*" or process.executable : "C:\\ProgramData\\Microsoft\\Windows Defender\\*") and (winlog.event_data.ProcessName : ("*\\MpCopyAccelerator.exe", "*\\MsMpEng.exe") or process.executable : ("*\\MpCopyAccelerator.exe", "*\\MsMpEng.exe"))))
```

# Designing a Behavior Based Detection

## Making repeatable detections through behavior

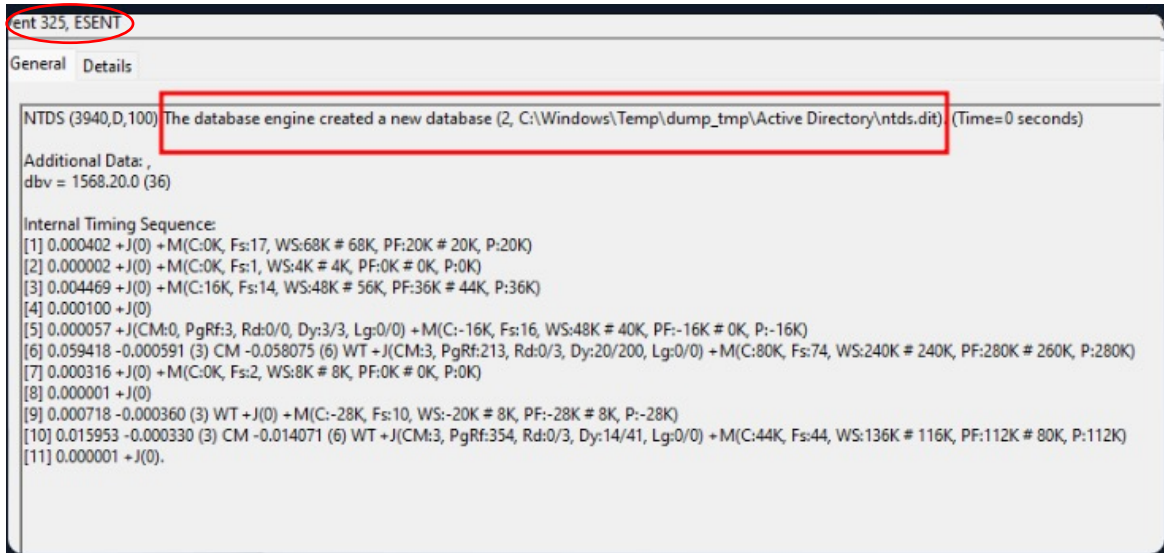
- Instead of looking at how the adversary did this, let's focus on what they are doing
- Behaviors translate the TTP into a more general adversary behavior based on their intention
  - This cannot change unless they no longer choose to do this part of their kill chain, regardless of implementation
- Focusing on common data sources allow us to identify more common logs for this behavior, regardless of how the attacker did it
- This yields a detection that works in more places and against multiple campaigns where the adversary has a behavior to copy NTDS

Attack Pattern




**Behavior**

Attacker copies the NTDS database




Behavior Based Detection analytic

```
event.code:325
```

Focus on what, not how

# Making our Detection

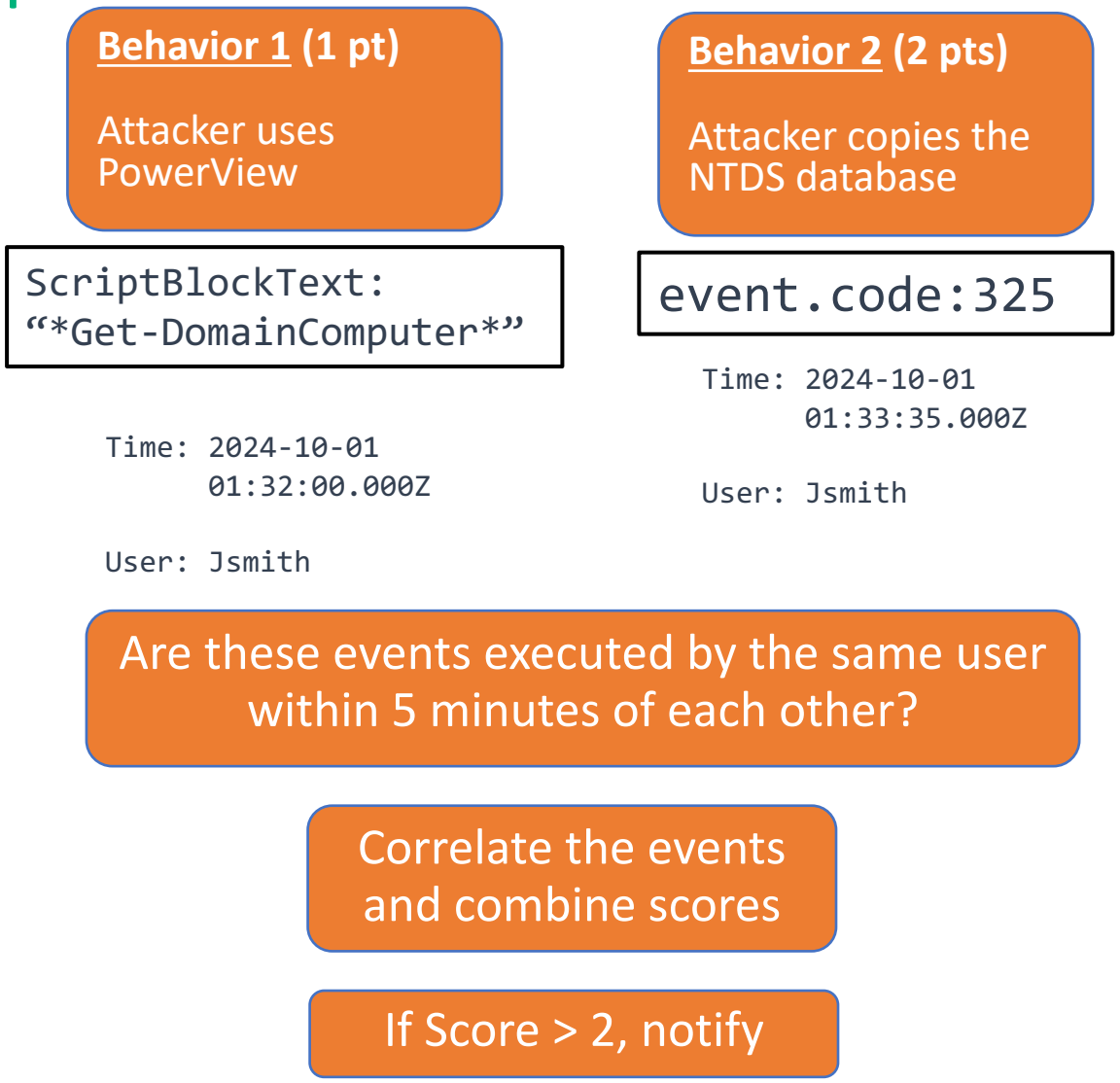
- Let's think on the NTDS database copying behavior
- The behavior is not very helpful without a detection
- Consider data sources common to your environment or across your community
  - Often standard endpoint and network logs
- Build out a detection using a standardized format
- False positives are not the primary concern but should be loosely considered
  - Threat emulation against existing controls/logs can be useful
  - These analytics are intended for machines but design may want to avoid thousands/millions of analytics running in production at the same time

```
title: Copying the NTDS database
id: q34np6k7-2m9r-7501-1842-x1v1j57b09t3
status: experimental
description: credentials sought to perform lateral
movement and access restricted information privileges
references:https://attack.mitre.org/techniques/T1555/
author: OCA
date: 2024/10/23
modified: 2024/10/23
tags:
    - attack.t1555
    - attack.credential_access
logsource:
    category: process_creation
    product: windows
detection:
    selection:
        event.code: 325
    condition: selection
falsepositives:
    - legitimate token
level: high
```

Note: These examples use Sigma format for the rules, that is not a requirement for IOB

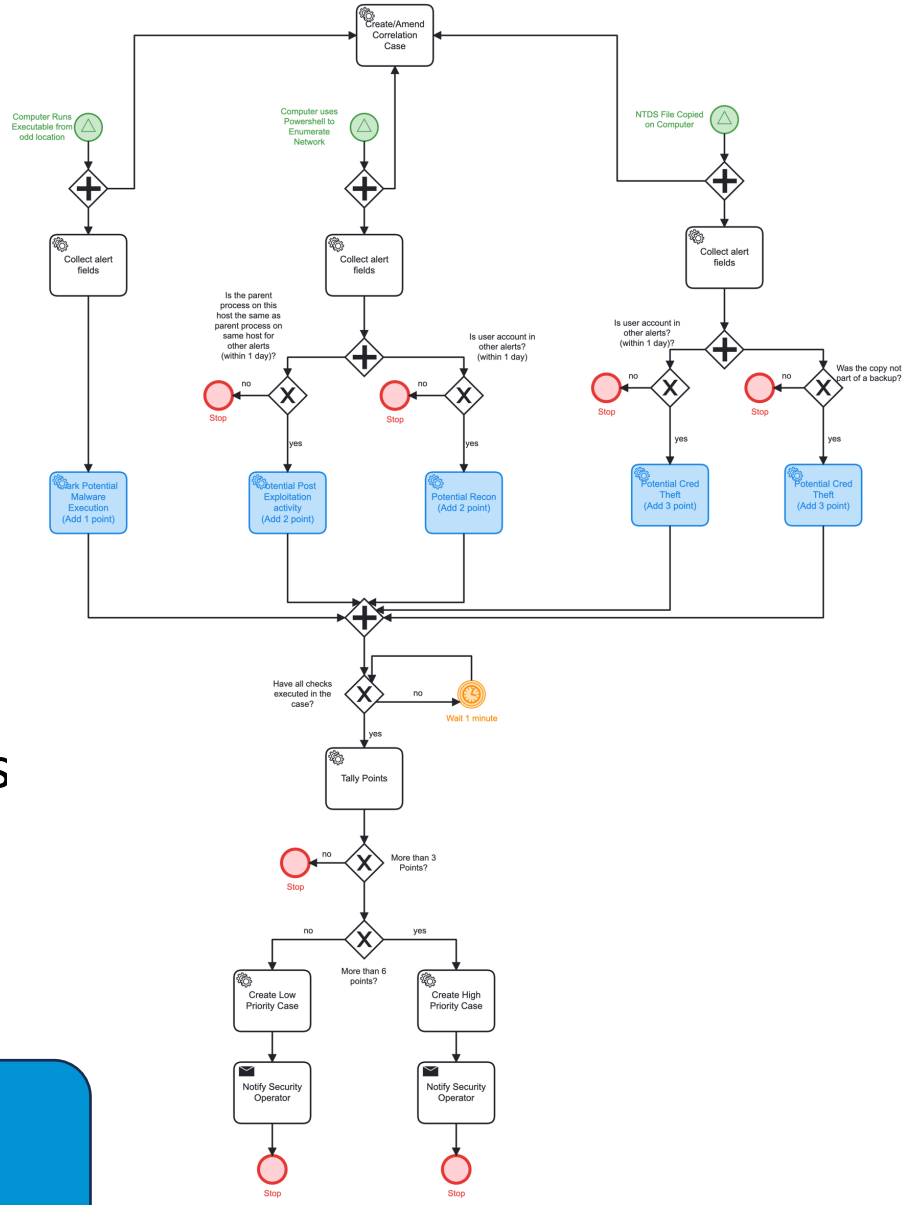
# IOB Correlation Approach

- Indicators of Behavior proposes a novel approach to alert correlation
- Behavior-based alerts are sent to an index to be monitored by automation
- Behaviors can be assigned “points” based on how indictive they are by themselves
- When an alert appears in the index, automation can look for correlated alerts from other behaviors
- If a threshold is met, then an analyst is notified



# Build a Correlation Workflow

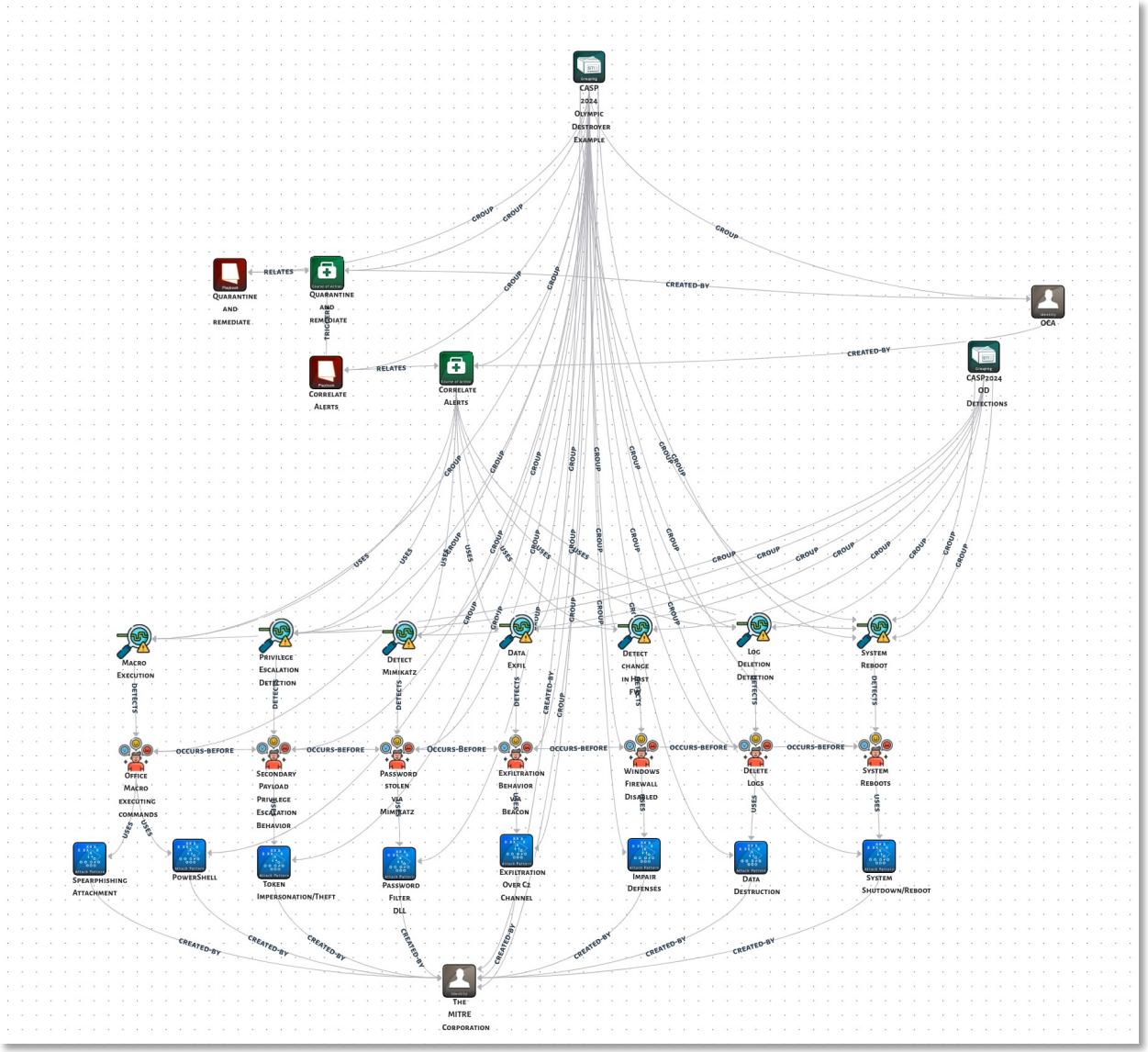
- Correlation playbooks are key to sharing IOB detection actions
- This is where we address false positives
- Each detection can be an alert to an index for machine monitoring
- Alerts trigger automated correlation workflows via playbook
- When “enough” alerts are correlated, then notify operators
- Recommend utilizing a standard playbook format, such as BPMN or CACAO
  - IOB will support a playbook in any file format




The goal is not a “Spark Rainstorm” Correlation, we want a “Malware -> Enumeration -> Credential Theft” Correlation

# Assemble the Artifacts

- The OCA maintains a set of STIX extensions at https://github.com/opencybersecurityalliance/stix-extensions
- These allow your modeling tools to incorporate IOB objects into your STIX bundles / datasets
- Compliant with STIX 2.1 standard
- This allows for the creation of an “IOB Bundle” to accompany your report



# Conclusion:
# Building Smarter Defenses Together

- Indicators of Behavior (IOBs) offer a transformative approach to cyber defense—proactively detecting adversary tactics through intent-based behavioral patterns instead of static artifacts.
- By representing behaviors in STIX-based, machine-readable formats, IOBs allow for:
  - Sharable, repeatable detections
  - Automated correlation workflows
  - Integrated response actions using standard playbooks like CACAO or BPMN
- The IOB framework bridges a critical gap between threat intel and defensive action—enabling earlier detection, deeper context, and stronger operational coordination across communities.

- Get Involved:
  - Explore tools, schemas, and reference bundles: https://github.com/opencybersecurityalliance/oca-iob
  - Join the IOB Sub-Project and attend public meetings: https://opencybersecurityalliance.org/iob/
  - Collaborate on pilots and share your feedback—your input helps shape the future of cyber defense.