Palo Alto XSOAR Integration Guide for Vectra AI (QUX)

Version: November 2025

Table of Contents

Introduction

Vectra AI for Palo Alto Networks XSOAR (formerly Demisto) empowers the SOC to create and manage incidents using Vectra AI's Attack Signal Intelligence for the Respond User Experience.

This integration allows the security operations center to create and manage incidents based on entities, powered by Vectra AI's Attack Signal Intelligence. Integrating Vectra and XSOAR enables security teams to synchronize Vectra Entities with XSOAR incidents in real time, making it feasible to manage operations from a single place.

Integration value is achieved by injecting Vectra's integrated signal into the security operations center in a structured and highly efficient approach to ultimately transform the analyst experience to enable:

- Rich Prioritization
- Incident Management
- Detailed Investigations
- Enrichment
- Enforcement
- Resolution
- Reporting

What's New in v2.1.0

New/Enhanced Fetch & Mirroring Options

- Advanced: Fetch escalated Accounts and Hosts
- Added support for an advanced parameter to fetch escalated accounts and hosts.
    - When set to true, the integration will fetch accounts and hosts that have been updated based on the provided filter parameters, even if their detection timestamps are older than the first fetch time.

Re-Fetch closed incidents via mirroring

- Added support for a parameter that controls how closed incidents are handled when mirrored:
    - If selected – new incidents are created (via outgoing mirroring).
    - If not selected – previously closed incidents are reopened (via incoming mirroring).
- This behavior is triggered only when the relevant account or host is still active and the previously fetched incident is closed.

Authentication Enhancements

OAuth 2.0 support

- Added OAuth 2.0 as a supported authentication method.
- Requires Client ID and Client Secret, which can be retrieved from: Vectra UI > Manage > API Clients.

Authentication Type parameter

- Added an Authentication Type parameter to choose the auth method:
- API Token (default)
- OAuth 2.0

New Commands

- **vectra-detections-mark-asopen** – Marks detections as open using the provided detection IDs.
- **vectra-account-markall-detections-asclosed** – Marks active detections as closed for a given account ID.
- **vectra-detections-mark-asclosed** – Marks detections as closed using the provided detection IDs.
- **vectra-host-markall-detections-asclosed** – Marks active detections as closed for a given host ID.

Fixes

- Fixed an issue with Outgoing Mirroring when invalid tags are provided.

Docker Image

- Updated the Docker image to: demisto/python3:3.12.12.549095

Document and Release Information

This section provides key metadata for the Vectra AI integration guide, including document versioning, supported platforms, intended audience, and the purpose of the document. It serves as a reference point to ensure that users are working with the correct release information before proceeding with deployment or configuration activities.

Field	**Details**
Document Title	Palo Alto XSOAR Integration Guide for Vectra AI (QUX)
Document Version	November 2025
Package Version	2.1.0
Supported Platform	Vectra Quadrant UX (QUX) and Palo Alto Networks Cortex XSOAR
Audience	SOC Analysts, Incident Responders, XSOAR Administrators, Security Architects
Purpose	This document provides comprehensive guidance for deploying, configuring, and operationalizing the Vectra QUX integration within Cortex XSOAR. It covers installation prerequisites, implementation steps, incident handling, playbook operations, workflows, troubleshooting procedures, and best practices to support effective SOC investigation and response.

Terminology

There are several terms that can be used interchangeably. The following table provides the XSOAR term as well as other terms that may be used to refer to the same.

XSOAR Term	Additional Terms
App	Vectra Detect for Palo Alto XSOAR
Instance	Configuration Profile, Asset
Action	Function, Command
Incident	Event, Alert, Container

Architecture

Integrating Vectra with XSOAR utilizes the REST API in a PULL model. The Vectra app resides inside the XSOAR platform and uses REST API calls to pull the appropriate data following the operator configured polling interval (figure 1).

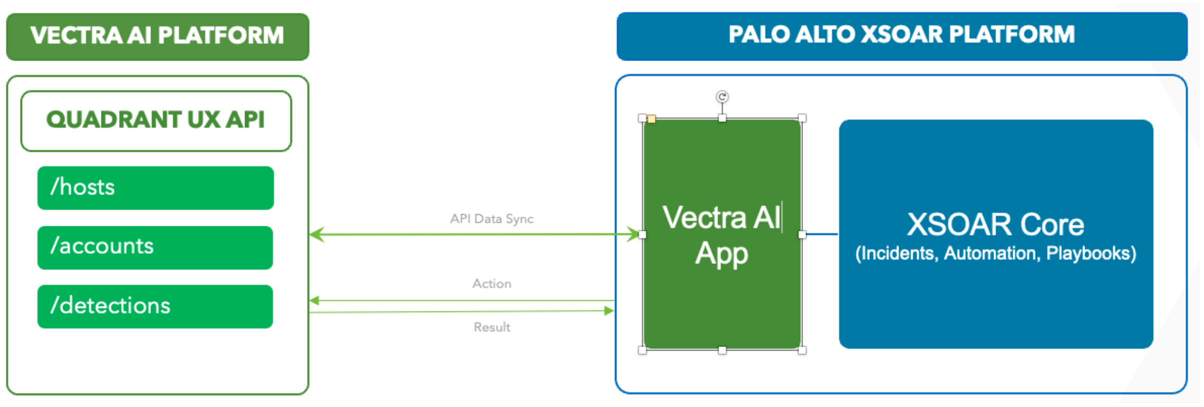


Figure 1 - Simplified Architecture

One or more Vectra AI Quadrant UX tenants provide the source of the data. Configuration Profiles (instances) are used to configure the details of how to communicate with a specific instance (i.e., tenant URL and API credentials), polling schedule, as well as API error handling. The integration is designed to retrieve entity (host and account) and detection data (along with all associated components such as notes, tags, and assignments) and ingestion filters enable the operator to fine tune the data that is considered for ingestion.

Multiple instances are required when there are multiple Vectra platforms but can also be used when there is a single platform that requires competing ingestion filters.

Once data begins ingestion, an app-embedded de-duplication mechanism controls if something is new and unique or if an update is warranted. If a previous incident does not exist a new incident will be created. If there is an existing incident, then the appropriate updates are made to ensure no duplicates.

Incidents that have been previously closed and start to re-offend again will cause the incident to re-open.

The final components of the app include the supported actions as well as automations and playbooks. These will be covered later in this document.

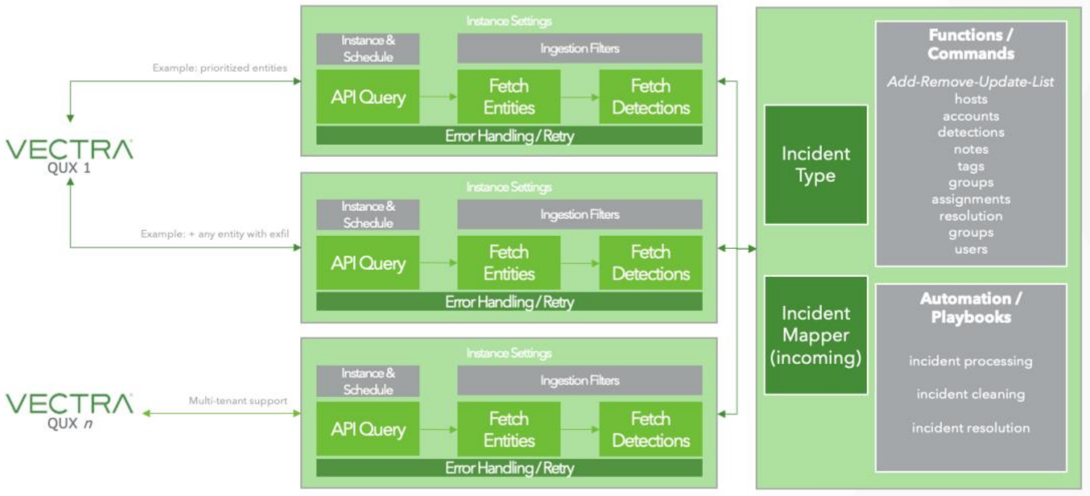


Figure 2 - Block Diagram

The next order of detail includes the operational components. As data makes it through ingestion as per the configuration profile, incidents are created. Each incident will have incident information in the form of context data. The context data includes the entity, detection and assignment data received from the Vectra platform. The integration includes support for several actions where each action is a command that can be issued to communicate with the Vectra platform (ex. add tag). Playbooks are used to define automation flows and can consist of multiple commands as well instructions for interfacing with other apps configured in the XSOAR environment.

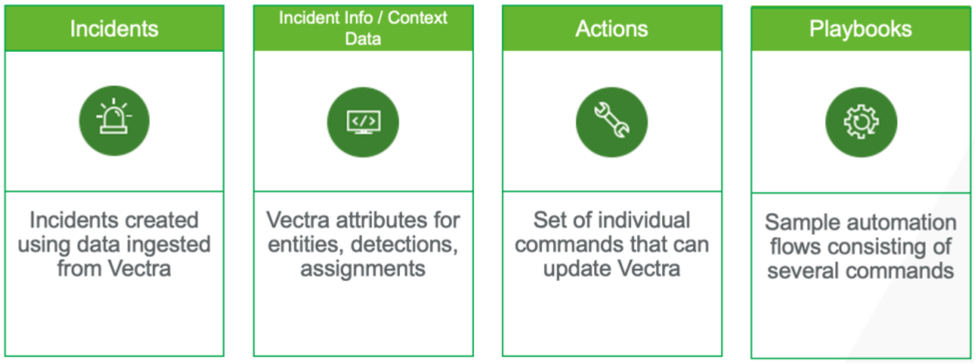

Incidents
Incidents created using data ingested from Vectra
Incident Info / Context Data
Vectra attributes for entities, detections, assignments
Actions
Set of individual commands that can update Vectra
Playbooks
Sample automation flows consisting of several commands

Figure 3 - Operational Components

Incidents are a foundational component of the integration as that is the starting point for any investigative or response workflow. Best practice is to implement an instance profile that generates incidents based on Vectra critical or high entities (hosts and accounts). These incidents are funneled into an analyst queue in XSOAR. While it's possible to generate incidents based on detections, this is NOT advised. Incidents are generated at the Vectra entity level and detections are associated with an entity. Vectra has already prioritized the entity and the detections (evidence) are attached to it so this ensures the operator is working on the most important incidents while simultaneously significantly reducing the number of alerts. An incident includes all the Vectra attributes that make up the entity and its detections. There are several other XSOAR components that are part of the incident layout. In addition to the Incident Info and Detections, these components include the War Room and Workplan, as well as several other components that may be blank if not utilized (ex. evidence board, related items, canvas). It's possible to attach a workplan to an incident, run individual actions or launch playbooks.

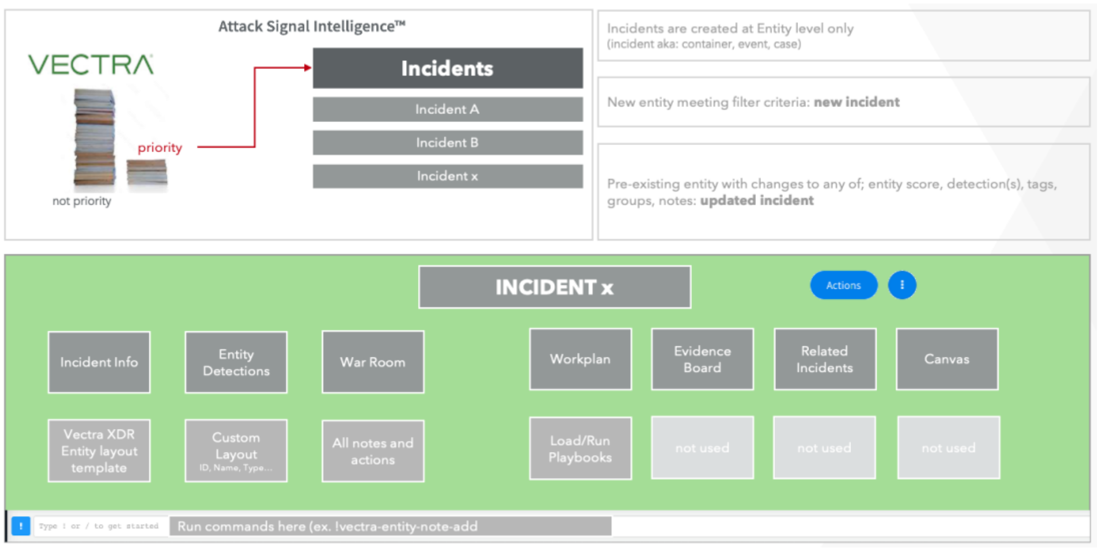

Attack Signal Intelligence™
VECTRA
priority
not priority
Incidents
Incident A
Incident B
Incident x
Incidents are created at Entity level only
(incident aka: container, event, case)
New entity meeting filter criteria: **new incident**
Pre-existing entity with changes to any of; entity score, detection(s), tags, groups, notes: **updated incident**
INCIDENT x
Actions
Incident Info
Entity Detections
War Room
Workplan
Evidence Board
Related Incidents
Canvas
Vectra XDR Entity layout template
Custom Layout
ID, Name, Type…
All notes and actions
Load/Run Playbooks
not used
not used
not used
Type ! or / to get started
Run commands here (ex. !vectra-entity-note-add

Figure 4 - Incident Structure

Implementation

Vectra Pre-requisites

The minimum requirements on the Vectra side to configure this integration include:

- Vectra Quadrant UX Tenant running API version 2.5 or higher.

  If you aren't sure which analyst experience is being utilized, please refer to this support article: https://support.vectra.ai/s/article/KB-VS-1673

- API Token from an account with the admin role.

  To retrieve the API Token:

  1. Log in to your Vectra Quadrant UX with an account that has the admin role.
  2. Navigate to *My Profile > View API Token* and re-enter your password to view the token.
  3. Copy the API Token and store in a safe location.

XSOAR Pre-requisites

The minimum requirements on the XSOAR side to configure this integration include:

- Palo Alto account that has access to download content from XSOAR Marketplace.
- XSOAR software installed (on-prem or cloud) v6.12.0 or higher.
- XSOAR local account with access to install apps.
- XSOAR platform must be able to communicate with the Vectra instance over port 443.

The Vectra integration does not impose any specific modifications or requirements of the XSOAR platform. Please refer to the XSOAR documentation for recommendations on system requirements.

Downloading and Installing the App

The integration (app) is available for download from the XSOAR Marketplace at this location:

https://cortex.marketplace.pan.dev/marketplace/details/Vectra_AI/

Publisher: Vectra
App Version: 2.1.0
Content Pack: 2.1.0
Supported Versions: Vectra Quadrant UX with Vectra API v2.5+

For easiest implementation, navigate to Marketplace from within your XSOAR instance and search for Vectra AI. Select the Vectra AI application and then click the install button.

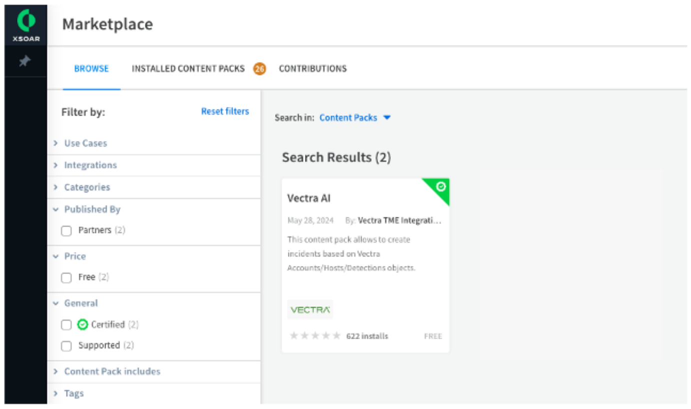


Figure 5 - App Install

Implementation Checklist

Prior to configuring a new instance, it may be helpful to review the following checklist as there are several parameters that may be required for configuration.

Parameter	Content
Name	Ex. Vectra_*tenantID*
Fetching	Enable or disable fetch
Classifier	Vectra Detect (included with app)
Incident Type	No incident type
Mapper (incoming)	Vectra Detect – Incoming Mapper (included with app)
Vectra Detect FQDN or IP	Ex. my.vectra.local or 192.168.1.1
API Token	Obtained from Vectra – My Profile
Certificate and Proxy	If using proxy, must be first configured in the system
First Fetch Timestamp	Not recommended to go back more than 7 days
Mirroring Direction	Incoming and Outgoing
Mirror tag for notes	Default is 'note' but can be changed – used to signal which XSOAR notes to mirror
Entity types to fetch	Hosts and/or Accounts (best practice is both) – Creating incidents from individual detections is not recommended as they are included in the entity
Tags	Hosts or Accounts that contain the tag will be retrieved
Detection category	Single selection (Botnet, C2, Recon, Lateral, Exfil, Info)
Detection type filtering	Single item free text – only entities with the detection specified will be ingested
Hosts fetch query	Recommend: host.threat:>=50
Accounts fetch query	Recommend: account.threat:>=50
Detections fetch query	Not recommended. Only used if Detections included in entity types to fetch
Max created incidents per fetch	Recommend 50 with maximum of 200
Advanced: Minutes of lookback	Recommend not to exceed 60
Incidents Fetch Interval	Specific fetch interval time

Figure 6 - Implementation Parameters

Initial Configuration of a New Instance

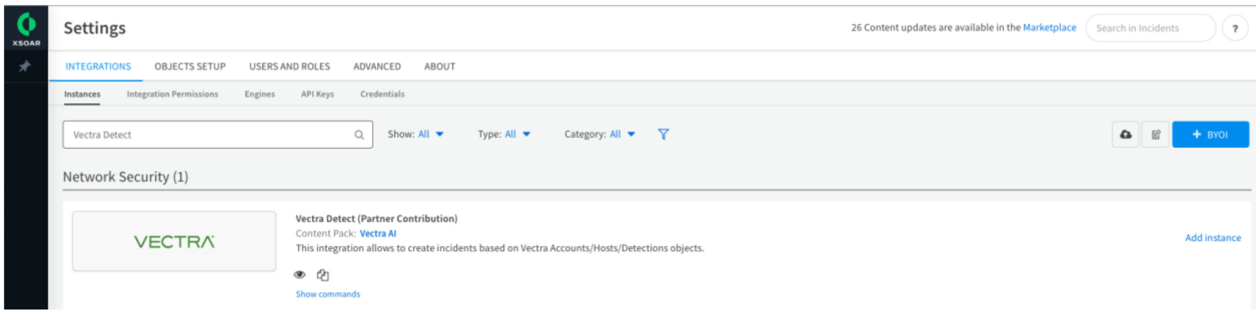
An asset configuration is required to allow XSOAR to communicate with the Vectra platform and pull data.

With the **Vectra Detect for Palo Alto XSOAR** pack installed:

- In the XSOAR UI, go to Settings → Integrations → Instances.
- Locate the Vectra DETECT integration and click Add instance (see *Figure 7 – Configure New Instance*).

Note:

- You can configure **multiple instances** for the same or different tenants.
- Multiple instances are required when you have:
    - Multiple Vectra tenants, or
    - Different/overlapping filters that cannot be represented in a single instance.
- Follow these procedures for configuring each instance as needed.



Vectra Detect (Partner Contribution)
Content Pack: Vectra AI
This integration allows to create incidents based on Vectra Accounts/Hosts/Detections objects.

Figure 7 - Configure New Instance

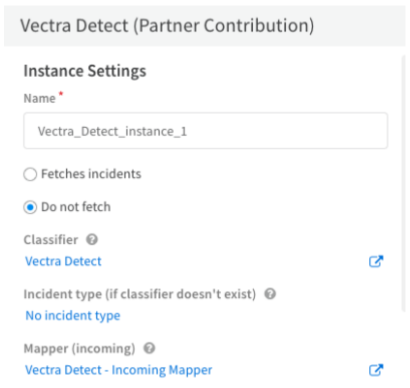
Initial Configuration of a New Instance

After clicking **Add instance**, the **New Instance** configuration page is displayed (Figure 7).

Although the instance configuration is presented as a **single page element**, it is logically organized into multiple sections (for documentation purposes). The first section contains the base application settings, including the instance name.

- When choosing an instance name, it is recommended to include an indicator for the **Vectra tenant** being queried (for example, Vectra- Detect-EU-Tenant or Vectra-Detect-Prod).
- A common best practice is to:
    - Configure **all settings** with **"Do not fetch"** enabled.
    - **Save** the configuration.
    - Run **Test** to validate connectivity and credentials.
    - Only after successful testing, **enable fetching**.

To save your work, click the **Save** icon before initiating a **Test**. Once saved, the full instance configuration view is displayed (Figure 8), including all sections relevant to connectivity, authentication, mirroring, and fetching.



Vectra Detect (Partner Contribution)

Instance Settings

Name *

Vectra_Detect_instance_1

Fetches incidents [✓ Do not fetch]

Classifier: Vectra Detect

Incident type (if classifier doesn't exist): No incident type

Mapper (incoming): Vectra Detect - Incoming Mapper

Figure 8a – Instance Settings

Classification and Mapping Requirements

For Vectra Detect incidents to be created and enriched correctly in XSOAR, the following settings are required on the instance:

- Incident Type:
  - Must be set to Vectra Detect Entity.
- Incoming Mapper:
  - Must be set to Vectra Detect.
- Classifier / Outgoing Mapper:
  - A classifier and outgoing mapper are not required for basic operation.
  - They can be configured if you need custom routing, normalization, or outbound mapping to other systems.

Configure these values before enabling fetching to ensure that incidents are created with the correct type and mapped fields.

Tenant Settings and Connectivity

The **Tenant Settings and Connectivity** section defines how Cortex XSOAR reaches the **Vectra Detect** platform and which credentials are used to authenticate.

These settings typically include:

- **Vectra Detect FQDN or IP** – The fully qualified domain name or IP address of the Vectra Detect platform.
- **Authentication Type & Credentials** – API Token or OAuth 2.0 (described below).
- **Certificate and Proxy Options** – Control how XSOAR establishes the HTTPS connection and whether a proxy is used.

Authentication Type

The integration supports two authentication methods. The fields displayed in the instance configuration depend on the selected **Authentication Type**.

API Token

Use this option when you want to authenticate directly with a long-lived API token.

- **API Token** – Required. The Vectra Detect API token generated for the integration user.
- **Client Secret Key** – Required. Secret key associated with the API token (as provided by Vectra Detect).

This method is straightforward and is typically used when a simple token-based integration is sufficient and you do not need an OAuth flow.

OAuth 2.0

Use this option when your environment requires an OAuth-based flow for security or SSO alignment.

- **Client ID** – Required. The OAuth client ID registered for the Vectra Detect integration.
- **Client Secret Key** – Required. The OAuth client secret associated with the client ID.

With OAuth 2.0, XSOAR uses the client credentials to obtain access tokens for calling the Vectra Detect API.

Certificates

The Vectra tenant uses **valid certificates** by default.

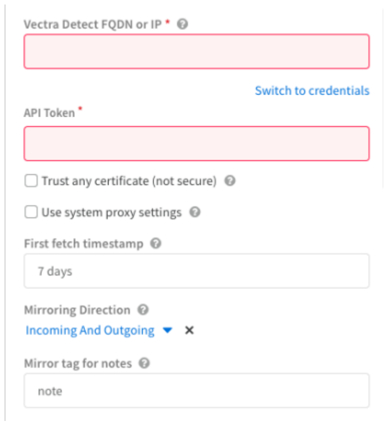
- In most environments, you **should not enable** the option to “trust any certificate”.
- Only enable “trust any certificate” if:
  - You are using an interception proxy or non-standard TLS setup **and**
  - You fully understand the security implications of trusting arbitrary certificates.

Proxy Configuration

If XSOAR must use a **proxy** to reach the (external) Vectra tenant:

- First, configure the proxy at the **XSOAR server level**:
  - Go to **Settings** → **About** → **Troubleshooting** → **Server Configuration**.
  - Define the proxy settings there (host, port, authentication, etc.).
- After the server proxy is configured, return to the Vectra Detect instance settings and:
  - Enable the **“Use system proxy”** (or equivalent **proxy checkbox**) to instruct this integration to use that proxy configuration.

This ensures all communication from the Vectra Detect integration to the Vectra platform is routed correctly through your enterprise proxy.


Vectra Detect FQDN or IP *

Switch to credentials

API Token *

☐ Trust any certificate (not secure)

☐ Use system proxy settings

First fetch timestamp: 7 days

Mirroring Direction: Incoming And Outgoing

Mirror tag for notes: note

Figure 8b – Instance Settings

First Fetch Time

First Fetch Time defines the **starting point in time** from which XSOAR begins fetching entities from Vectra.

You can specify this as either:

- A **relative time** (e.g., last few hours/days), or
- An **absolute timestamp**.

Supported formats include:

- Relative:
  - 2 minutes
  - 2 hours
  - 2 days
  - 2 weeks
  - 2 months
  - 2 years
- Absolute:
  - yyyy-mm-dd
  - yyyy-mm-ddTHH:MM:SSZ

Examples:

- 2 days
- 01 May 2023
- 01 Mar 2021 04:45:33
- 2022-04-17T14:05:44Z

Recommendations:

- For **new deployments**, use a **recent relative time** (for example, 3 days) to avoid flooding XSOAR with historical data on first run.
- For **migration / replay scenarios**, use an **absolute timestamp** aligned with your cutover date so that incidents start from a controlled point in time.

Once the first fetch has completed, subsequent fetches will continue from the **last fetched time** regardless of the original First Fetch Time setting.

Mirroring Settings

Mirroring controls the direction of data synchronization between **XSOAR** and **Vectra XDR**.

- **Outgoing mirroring**
  - Sends updates from XSOAR to Vectra, such as:
    - Tags
    - Assignments / ownership
    - Resolution / status
  - These changes made in XSOAR are reflected back on the Vectra side.
- **Incoming mirroring**
  - Pulls updates from Vectra to XSOAR, such as:
    - Tags
    - Assignments / ownership
    - Resolution / status
    - Detections
  - These changes made in Vectra are mirrored into the corresponding XSOAR incidents.

Recommended:

- In most deployments, **both Incoming and Outgoing** are enabled to maintain **bi-directional synchronization** between XSOAR and Vectra.

Re-Fetch Closed Incidents via Mirroring

The **Re-Fetch closed incidents via mirroring** option controls how XSOAR handles entities that become active again in Vectra XDR after their related incidents have already been closed in XSOAR.

This behavior is driven by a single checkbox in the Vectra XDR integration instance:

Checkbox: Re-Fetch closed incidents via mirroring

- **If selected** – new incidents will be created (**via Outgoing Mirroring**).
- **If not selected** – it reopens previously closed incidents (**via Incoming Mirroring**).

Note: This flow is triggered only when the relevant entity is still active and the previously fetched incident is closed.

When This Flow Is Triggered

The re-fetch logic runs as part of the normal mirroring / fetch cycle. It applies only when all of the following are true:

- The entity exists in **Vectra XDR** and is currently **active**.
- XSOAR has previously created an incident for that entity from the **Vectra XDR integration**.
- That previously created incident is now **closed** in XSOAR.

If any of these conditions are not met (for example, the entity is no longer active, or there was never an incident for it), the re-fetch flow does **not** run, and the mirroring behaves in the standard way.

Choose Your Desired Behavior

A. Create New Incidents (Checkbox Selected)

- Check **Re-Fetch closed incidents via mirroring.**
- Resulting behavior:
    - When an entity with a previously closed incident becomes active again:
        - XSOAR **creates a new incident** for that entity (**via Outgoing Mirroring**).
        - The old incident remains closed.
- Use this setting if:
    - You want **separate incidents** for each new wave of activity.
    - Different teams / shifts own different time windows.
    - Reporting is incident-based (e.g., count incidents per month, SLA metrics, ticket queues).

B. Reopen Existing Incidents (Checkbox Not Selected)

- Leave **Re-Fetch closed incidents via mirroring unchecked.**
- Resulting behavior:
    - When an entity with a previously closed incident becomes active again:
        - XSOAR reopens the previously closed incident for that entity (via Incoming Mirroring).
        - New detections and context are appended to the same incident.
- Use this setting if:
    - You prefer a single continuous investigation per entity.
    - Analysts should see full history over time in one incident.
    - You track long-lived or recurring adversary behavior under a single case.

Save & Validate

- Click **Save** on the integration instance.
- After the next mirroring/fetch cycle:
    - Close a test incident tied to an active entity.
    - Trigger new detections for the same entity in Vectra XDR.
    - Confirm that XSOAR either **created a new incident** or **reopened the existing one**, according to your checkbox selection.

Example Flows

Example 1 – Checkbox Selected (New Incident via Outgoing Mirroring)

- An entity in Vectra XDR generates detections → XSOAR creates **Incident #101**.
- Analysts investigate and **close Incident #101**.
- Later, the same entity becomes active again with new detections.
- During mirroring:
    - The entity is still active.
    - The last related incident (Incident #101) is closed.
    - The checkbox is **selected**.
- XSOAR creates a new incident (e.g., Incident #145) via Outgoing Mirroring.

Result:

- Incident #101 = prior investigation (closed).
- Incident #145 = new investigation for the same entity.

Example 2 – Checkbox Not Selected (Incident Reopened via Incoming Mirroring)

- An entity in Vectra XDR generates detections → XSOAR creates **Incident #202**.
- Analysts investigate and **close Incident #202**.
- Later, the same entity becomes active again.
- During mirroring:
  - The entity is still active.
  - The last related incident (Incident #202) is closed.
  - The checkbox is **not selected**.
- XSOAR **reopens Incident #202** via **Incoming Mirroring** and adds the new detections/context.

Result:

- Analysts see **one incident (#202)** containing both the original and the new activity.

Mirror Tag for Notes

Mirror tag for notes controls which XSOAR notes are sent (mirrored) to Vectra.

- The value you configure here is a **tag string** (prefix) that must appear in the **note body** in XSOAR.
- Only notes that **start with** this tag value will be mirrored from XSOAR to Vectra as entity notes.

The **default tag value is** note.

Filtering Settings

The **Filtering** section is critical, as it determines **which Vectra entities become XSOAR incidents**.

Vectra defines **two entity types**:

- Accounts
- Hosts

Entity Type

- Leaving the entity type field blank means all entity types are included.
- Best practice: Explicitly select Accounts and Hosts to avoid ambiguity and to be explicit about what you are ingesting.
- You can also choose to ingest from only one entity type if required by the use case (for example, only accounts for identity-focused workflows).

Additional Filters

The following filters allow for **fine-tuning** which entities and detections are ingested into XSOAR. These are typically used in **secondary instances** that target more specific use cases.

Tags
Retrieve entities that contain **any** of the specified tags.

- Supports comma-separated values.

- If multiple tags are provided, an entity that matches at least one of the tags is included.

Detection Category

Retrieve detections belonging to a specified **detection category**.

This is a **single-select** field. Example categories include:

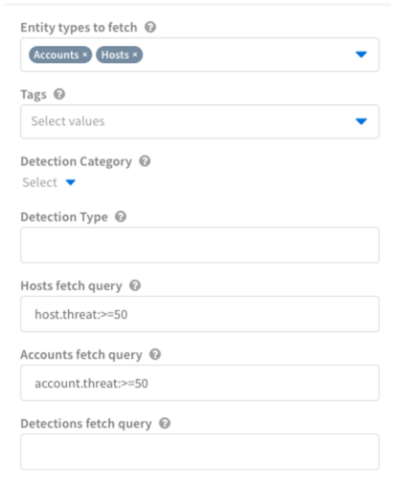
- Command & Control
- Botnet
- Reconnaissance
- Lateral Movement
- Exfiltration
- Info

Use this when you want an instance focused on **specific stages of the attack lifecycle** (for example, one instance dedicated to **Command & Control** and **Exfiltration**).

Detection Type

Retrieve detections belonging to a specified **detection type**.

- Use this to target **specific detection names/types**
- This is typically used when a use case or playbook is designed to respond to a **very specific detection** rather than an entire category.



Entity types to fetch: Accounts, Hosts

Tags: Select values

Detection Category: Select

Detection Type:

Hosts fetch query: host.threat:>=50

Accounts fetch query: account.threat:>=50

Detections fetch query:

Figure 8c – Instance Settings

Fetch and Lookback Settings

This section controls **which entities are fetched as incidents**, how many new incidents are created per cycle, and how far back the integration looks to avoid missing detections.

Hosts fetch query

Hosts fetch query defines which **active Hosts** are eligible to be fetched as incidents.

- Only **active Hosts** that match this query will be fetched.
- This query is used **only if "Hosts"** is selected in **Entity types to fetch**.
- **Default:** host.threat:>=50

Use this field to further restrict host ingestion (for example, to specific tags, subnets, or higher threat thresholds) while still relying on Vectra's scoring.

Accounts fetch query

Accounts fetch query defines which **active Accounts** are eligible to be fetched as incidents.

- Only **active Accounts** that match this query will be fetched.
- This query is used **only if "Accounts"** is selected in **Entity types to fetch**.
- **Default:** account.threat:>=50

This can be tuned if you want to bring in only higher-risk accounts or target particular environments (e.g., privileged user sets).

Detections fetch query

Detections fetch query defines which **active Detections** are eligible to be fetched as incidents.

- Only **active Detections** that match this query will be fetched.
- This query is used **only if "Detections"** is selected in **Entity types to fetch**.
- **Default:** detection.threat:>=50 AND detection.certainty:>=50

Adjust this query if you want to focus on higher-confidence detections, specific detection categories, or tighter filters for high-volume environments.

Max created incidents per fetch

Max created incidents per fetch limits how many **new incidents** can be created in each fetch cycle.

- The total value is **shared across** all selected **Entity types to fetch** (Hosts, Accounts, Detections).
- If this value is set **greater than 200**, it is internally capped at **200**.
- **Maximum effective value:** 200
- **Default:** 50

Example:

- If Max created incidents per fetch = 50 and both Hosts and Accounts are enabled:
    - The 50-incident budget is split across whatever entities match the queries in that cycle.

Best practice:

- Keep the default (**50**) when:
    - You are using an aggressive fetch interval (e.g., **5 minutes or less**), or
    - You are in an environment with high detection volume.
- Increase towards **200** only if you need to clear large backlogs and your XSOAR environment can handle the incident volume.

Advanced: Minutes to look back when fetching

Advanced: Minutes to look back when fetching controls the **lookback window** used on each fetch.

- The integration searches **backward in time** from the last run to find entities/detections that:
  - Were created before the last run time, and
  - **Did not match** the fetch query at that time, but **do match now** (for example, threat/urgency increased).

This protects against:

- API pipeline delays.
- Detections being published with timestamps that appear slightly "in the past".

Defaults & recommendations:

- **Default:** 60 minutes
  - Recommended for most environments to ensure that no detections are missed due to timing delays.
- Looking back **longer than 60 minutes is usually not required**.
- In environments under **heavy load** (many detections and high event volume):
  - Consider reducing this value to **20 minutes** to limit re-evaluation overhead while still handling typical delay windows.

Advanced: Fetch escalated Accounts and Hosts

Advanced: Fetch escalated Accounts and Hosts is a **select/boolean option** that adds special handling for escalated entities.

- When set to **True**, the integration will fetch **escalated Accounts and Hosts** that have been **updated** according to your filter parameters, even if:
  - Their detection timestamps are **older than the First Fetch Time**.

This is useful when:

- You want to ensure that escalations or status changes on existing entities are not missed just because their detections are older.
- SOC workflows rely on **escalation events** and **assignment outcomes** as triggers in XSOAR.

Fetch / Polling Interval

The **fetch interval** controls how frequently XSOAR polls Vectra for new or updated entities:

- Configure this value according to your SOC's operational needs and platform capacity.
- XSOAR can fetch as frequently as **every 1 minute** for near real-time updates.
- In many environments, a slightly longer interval (e.g., 3–5 minutes) can be chosen to balance **timeliness** with **load**.

Make sure the chosen interval is aligned with:

- Your SOC's responsiveness expectations.
- The volume of entities and detections in your Vectra deployment.
- Any broader platform performance considerations.

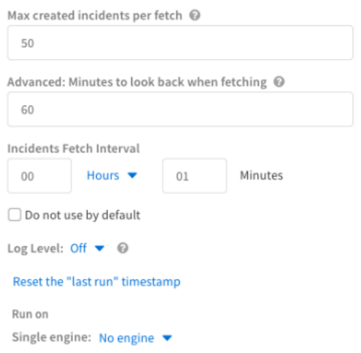

Max created incidents per fetch
50
Advanced: Minutes to look back when fetching
60
Incidents Fetch Interval
00 Hours 01 Minutes
☐ Do not use by default
Log Level: Off
Reset the "last run" timestamp
Run on
Single engine: No engine

Figure 8d – Instance Settings

Operational Components

At this stage, the Vectra app is installed, configured and the system should be receiving data. The architecture and implementation are complete so next we will take a deeper look into the operational perspective in more detail.

Incidents

Also referred to as containers, events, and alerts, an incident is the starting point for incident management, workflow, and automation. Vectra incidents are accessible via the Incidents menu in the XSOAR UI (figure 9).

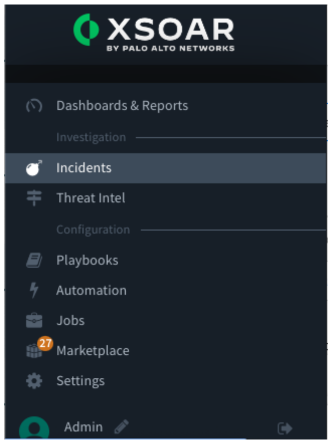
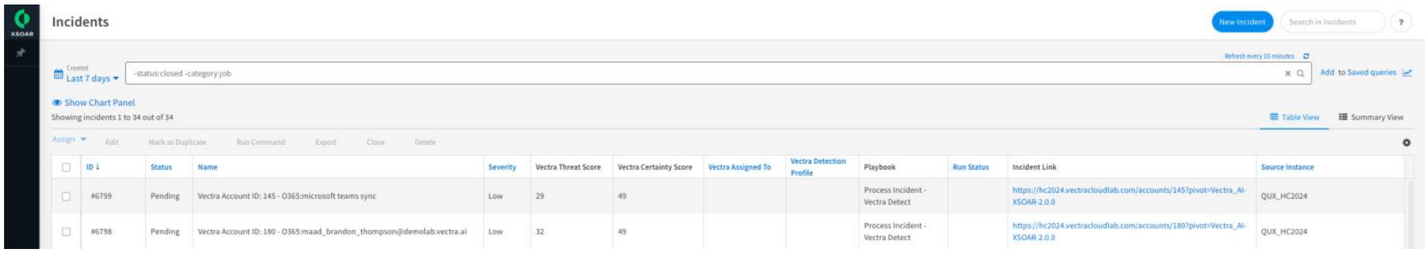


Figure 9 – Incidents

The ingestion mechanism includes de-duplication logic natively to ensure that if an incident doesn't already exist a new one is created. Conversely, if an incident already exists then the incident is updated rather than duplicated. Security analysts should strive to resolve events as they appear as they are already prioritized via threat/certainty risk scoring. The list of incidents can be viewed from the incidents page and the operator can specify filters, column, and sort ordering. The recommended column order (ID, Status, Name, Severity, Threat Score, Certainty Score, Assigned To, Detection Profile, Playbook, Run Status, Incident Link, Source Instance) is shown in the figure below (Figure 10).



-status:closed -category:job

Showing incidents 1 to 34 out of 34

ID ↓	Status	Name	Severity	Vectra Threat Score	Vectra Certainty Score	Vectra Assigned To	Vectra Detection Profile	Playbook	Run Status	Incident Link	Source Instance
#6799	Pending	Vectra Account ID: 145 - O365:microsoft teams sync	Low	29	49			Process Incident - Vectra Detect		https://hc2024.vectracloudlab.com/accounts/145?pivot=Vectra_AI-XSOAR-2.0.0	QUX_HC2024
#6798	Pending	Vectra Account ID: 180 - O365:maad_brandon_thompson@demolab.vectra.ai	Low	32	49			Process Incident - Vectra Detect		https://hc2024.vectracloudlab.com/accounts/180?pivot=Vectra_AI-XSOAR-2.0.0	QUX_HC2024

Figure 10 – Incidents Page

Customizing the column order is done using the gear icon. The columns will appear in the order they are selected so it's recommended to deselect all columns and then select the columns required in the order you want them to appear from left to right (Figure 11).

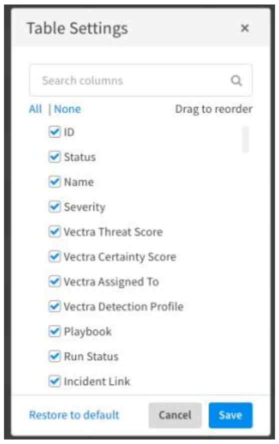


Figure 11 - Column Order

Incident Template

The Vectra Detect integration for XSOAR includes a pre-defined incident layout template. This layout template includes multiple tabs as shown in Figure 12. This layout is displayed as each incident is accessed. Details for the relevant tabs are covered next.

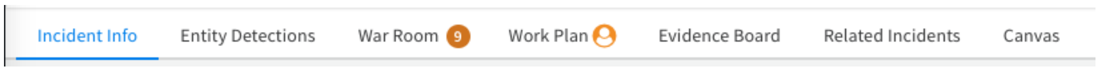


Figure 12 - Layout Template

Incident Info

To view the details of a single incident, select the incident ID field (Figure 10). Selecting a single incident will bring up the incident layout (Figure 13) starting with the Incident Info. Incident Info is broken into several sections and provides a single page view with access to the most relevant data and commands.

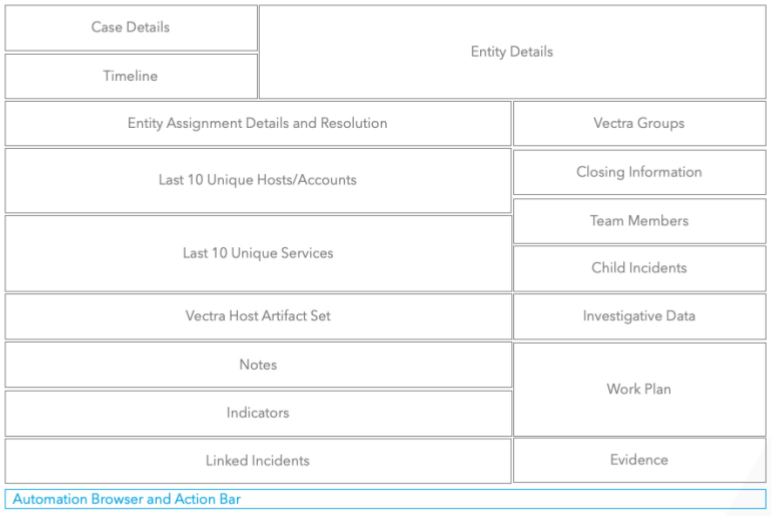


Figure 13 - Incident Layout

Entity Detections

Each incident is tied to an entity and each entity can have one or more Detections. The Detections that are associated with an entity are displayed in table form within the Detections tab of the incident template.

War Room

Each incident has its own war room. The war room facilitates real-time investigation and keeps track of all activities related to an incident.

Work Plan

The work plan for each Vectra incident will include the Process Incident – Vectra Detect playbook. The work plan will show the run status of the playbook. The operator can also change the playbook to run from within the work plan.

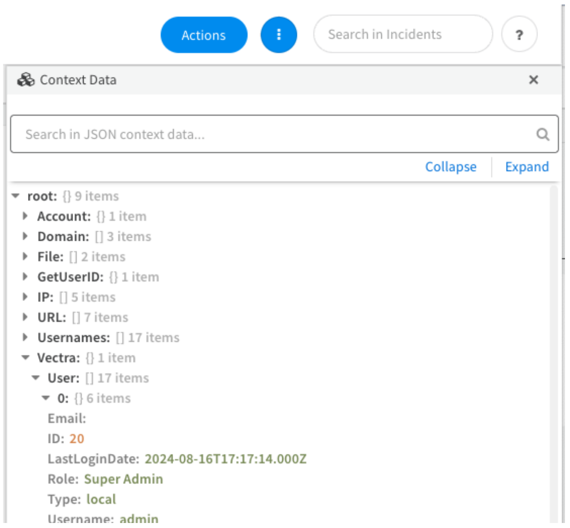
Remaining Tabs

The remaining tabs (Evidence Board, Related Incidents, Canvas) aren't initially engaged as part of Vectra incidents. These tabs may be used as required during the lifecycle of the incident.

Context Data

The layout contains the high-level data for an incident but additional data for Vectra and XSOAR attributes are available as "Context Data" (Figure 14). This data is accessed by selecting "Context Data" from the ⋮ menu. Context data is expandable JSON that is searchable, and it contains all attributes relevant to Vectra and XSOAR. Vectra data includes all API attributes returned from calls to the following endpoints: (/entities,

/Detections, /assignments). Elements including notes, tags, groups, etc.… are included in these endpoints. Context Data is available from several locations and not just the Incident Info view.

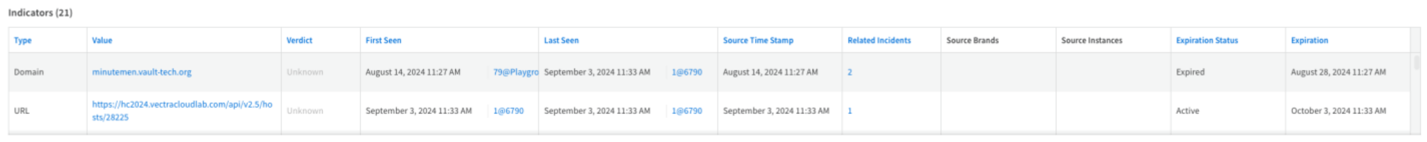


```
root: {} 9 items
  Account: {} 1 item
  Domain: [] 3 items
  File: [] 2 items
  GetUserID: {} 1 item
  IP: [] 5 items
  URL: [] 7 items
  Usernames: [] 17 items
  Vectra: {} 1 item
    User: [] 17 items
      0: {} 6 items
        Email:
        ID: 20
        LastLoginDate: 2024-08-16T17:17:14.000Z
        Role: Super Admin
        Type: local
        Username: admin
```

Figure 14 - Context Data

Indicators

Artifacts such as domain names, URLs, accounts, and IP addresses are automatically mapped by XSOAR as indicators. These indicators are present in the Context Data (Figure 14) in JSON as well as in the indicators section of the Incident Layout (Figure 15). The default XSOAR indicator settings are employed so there is not a custom incoming indicator mapper.



Indicators (21)

Type	Value	Verdict	First Seen	Last Seen	Source Time Stamp	Related Incidents	Source Brands	Source Instances	Expiration Status	Expiration
Domain	minutemen.vault-tech.org	Unknown	August 14, 2024 11:27 AM 79@Playgro	September 3, 2024 11:33 AM 1@6790	August 14, 2024 11:27 AM	2			Expired	August 28, 2024 11:27 AM
URL	https://hc2024.vectracloudlab.com/api/v2.5/hosts/28225	Unknown	September 3, 2024 11:33 AM 1@6790	September 3, 2024 11:33 AM 1@6790	September 3, 2024 11:33 AM	1			Active	October 3, 2024 11:33 AM

Figure 15 - Indicators

Actions

Actions are commands that can be run against the Vectra AI Platform. The following table (Table 1) outlines all the supported commands for this integration and indicates which actions have been added for this update.

Table 1 - Actions List

Action	Description	Parameters (summary)
vectra-search-accounts	Returns a list of accounts matching the given filters.	min_id, max_id, min_threat, max_threat, min_certainty, max_certainty, state, search_query, search_query_only, min_privilege_level, max_privilege_level, privilege_category, tags
vectra-search-hosts	Returns a list of hosts matching the given filters.	min_id, max_id, min_threat, max_threat, min_certainty, max_certainty, state, search_query, search_query_only

vectra-search-detections	Returns a list of detections matching the given filters.	min_id, max_id, min_threat, max_threat, min_certainty, max_certainty, state, search_query, search_query_only
vectra-search-assignments	Returns a list of assignments matching the given filters.	account_ids, assignee_ids, host_ids, outcome_ids, resolved
vectra-search-users	Returns a list of users (operators) matching the given filters.	username, role, type, last_login_datetime
vectra-search-outcomes	Returns a list of assignment outcomes.	*(no parameters)*
vectra-account-describe	Returns details for a single account.	id
vectra-account-add-tags	Adds tags to an account.	id, tags
vectra-account-del-tags	Removes tags from an account.	id, tags
vectra-account-tag-list	Lists tags for a specific account.	id
vectra-account-note-add	Adds a note to an account.	account_id, note
vectra-account-note-update	Updates an existing note on an account.	account_id, note_id, note
vectra-account-note-remove	Removes a note from an account.	account_id, note_id
vectra-account-note-list	Lists all notes for a specific account.	account_id
vectra-account-markall-detections-asfixed	Marks all active detections for an account as fixed.	account_id
vectra-account-markall-detections-asclosed	Closes all active detections for an account with a given reason.	account_id, close_reason
vectra-host-describe	Returns details for a single host.	id
vectra-host-add-tags	Adds tags to a host.	id, tags
vectra-host-del-tags	Removes tags from a host.	id, tags
vectra-host-tag-list	Lists tags for a specific host.	id
vectra-host-note-add	Adds a note to a host.	host_id, note

vectra-host-note-update	Updates an existing note on a host.	host_id, note_id, note
vectra-host-note-remove	Removes a note from a host.	host_id, note_id
vectra-host-note-list	Lists all notes for a specific host.	host_id
vectra-host-markall-detections-asfixed	Marks all active detections for a host as fixed.	host_id
vectra-host-markall-detections-asclosed	Closes all active detections for a host with a given reason.	host_id, close_reason
vectra-detection-describe	Returns details for a single detection.	id
vectra-detection-get-pcap	Retrieves the PCAP file for a detection (if available).	id
vectra-detection-markasfixed	Marks or unmarks a detection as fixed.	id, fixed
vectra-detection-add-tags	Adds tags to a detection.	id, tags
vectra-detection-del-tags	Removes tags from a detection.	id, tags
vectra-detection-tag-list	Lists tags for a specific detection.	id
vectra-detection-note-add	Adds a note to a detection.	detection_id, note
vectra-detection-note-update	Updates an existing note on a detection.	detection_id, note_id, note
vectra-detection-note-remove	Removes a note from a detection.	detection_id, note_id
vectra-detection-note-list	Lists all notes for a specific detection.	detection_id
vectra-detections-mark-asclosed	Closes one or more detections with a given reason.	detection_ids, close_reason
vectra-detections-mark-asopen	Reopens one or more detections.	detection_ids
vectra-outcome-describe	Returns details for a single assignment outcome.	id
vectra-outcome-create	Creates a new assignment outcome.	title, category

vectra-assignment-describe	Returns details for a single assignment.	id
vectra-assignment-assign	Assigns an account/host entity to a Vectra user for investigation.	assignee_id, assignment_id, account_id, host_id
vectra-assignment-resolve	Resolves an assignment, optionally filtering which detections are included.	assignment_id, outcome_id, note, detections_filter, filter_rule_name, detections_list
vectra-user-describe	Returns details for a single Vectra user.	id
vectra-group-list	Returns a list of groups with optional filters.	group_type, account_names, domains, host_ids, host_names, importance, ips, description, last_modified_timestamp, last_modified_by, group_name
vectra-group-assign	Assigns members to the specified group.	group_id, members
vectra-group-unassign	Unassigns members from the specified group.	group_id, members

Vectra Detect Commands Overview

These commands are available from the Cortex XSOAR War Room, automations, and playbooks. Each command calls the **Vectra Detect** API, writes data into incident context under Vectra.*, and renders a human-readable table (where applicable).

vectra-search-accounts

- **Purpose:** Search for accounts in Vectra Detect.
- **Description:** Returns account entities that match the provided score, state, privilege, tag, and Lucene search filters. All filters are cumulative unless search_query_only is used.
- **Input Arguments:**
  - min_id – (Optional) Minimum account ID.
  - max_id – (Optional) Maximum account ID.
  - min_threat / max_threat – (Optional) Threat score range.
  - min_certainty / max_certainty – (Optional) Certainty score range.
  - state – (Optional) active or inactive.
  - search_query – (Optional) Lucene-style query; combined with other filters.
  - search_query_only – (Optional) Use only this query, ignore other filters.
  - min_privilege_level / max_privilege_level – (Optional) Privilege level range.
  - privilege_category – (Optional) low, medium, or high.
  - tags – (Optional) One or more tags (comma-separated).

vectra-search-hosts

- **Purpose:** Search for hosts in Vectra Detect.

- **Description:** Returns host entities that match the given score, state, and Lucene query filters. All filters are cumulative unless search_query_only is used.
- **Input Arguments:**
  - min_id, max_id – (Optional) Host ID range.
  - min_threat, max_threat – (Optional) Threat score range.
  - min_certainty, max_certainty – (Optional) Certainty score range.
  - state – (Optional) active or inactive.
  - search_query – (Optional) Lucene-style query.
  - search_query_only – (Optional) Use only this query.

vectra-search-detections

- **Purpose:** Search for detections in Vectra Detect.
- **Description:** Returns detections that match score, state, and Lucene query filters. Filters are cumulative unless search_query_only is used.
- **Input Arguments:**
  - min_id, max_id – (Optional) Detection ID range.
  - min_threat, max_threat – (Optional) Threat score range.
  - min_certainty, max_certainty – (Optional) Certainty score range.
  - state – (Optional) active or inactive.
  - search_query – (Optional) Lucene-style query.
  - search_query_only – (Optional) Use only this query.

vectra-search-assignments

- **Purpose:** Search for assignments.
- **Description:** Returns assignments for accounts/hosts, optionally filtering on resolution state, assignee, or outcome. Resolved assignments are excluded by default.
- **Input Arguments:**
  - account_ids – (Optional) Filter by account IDs.
  - assignee_ids – (Optional) Filter by Vectra user IDs.
  - host_ids – (Optional) Filter by host IDs.
  - outcome_ids – (Optional) Filter by outcome IDs.
  - resolved – (Optional) Filter by resolution state.

vectra-search-users

- **Purpose:** List Vectra Detect users/operators.
- **Description:** Returns users with optional filters for username, role, type, and last login time.
- **Input Arguments:**
  - username – (Optional) Filter by username.
  - role – (Optional) Filter by role.
  - type – (Optional) Filter by authentication type (for example, local or SAML).

- last_login_datetime – (Optional) Only users who logged in on/after this time.

vectra-search-outcomes

- **Purpose:** List assignment outcomes.
- **Description:** Returns all configured assignment outcomes, including built-in and custom results.
- **Input Arguments:**
  - *None.*

vectra-account-describe

- **Purpose:** Get details for a specific account.
- **Description:** Returns metadata, scores, state, tags, and URL for a single Vectra account.
- **Input Arguments:**
  - id – (Optional) Account ID to retrieve.

vectra-account-add-tags

- **Purpose:** Tag an account.
- **Description:** Adds one or more tags to the specified account.
- **Input Arguments:**
  - id – (Optional) Account ID.
  - tags – (Optional) Tags to add (comma-separated).

vectra-account-del-tags

- **Purpose:** Remove tags from an account.
- **Description:** Deletes one or more tags from the specified account.
- **Input Arguments:**
  - id – (Optional) Account ID.
  - tags – (Optional) Tags to remove (comma-separated).

vectra-account-tag-list

- **Purpose:** List tags on an account.
- **Description:** Returns all tags currently associated with the specified account.
- **Input Arguments:**
  - id – (Required) Account ID.

vectra-account-note-add

- **Purpose:** Add a note to an account.

- **Description:** Creates a new note on the specified account and returns note metadata.
- **Input Arguments:**
  - account_id – (Required) Account ID.
  - note – (Required) Note text.

vectra-account-note-update

- **Purpose:** Update an existing account note.
- **Description:** Modifies an existing note on the account, keeping note history fields updated.
- **Input Arguments:**
  - account_id – (Required) Account ID.
  - note_id – (Required) ID of the note to update.
  - note – (Required) New note text.

vectra-account-note-remove

- **Purpose:** Remove an account note.
- **Description:** Deletes a specific note from the given account.
- **Input Arguments:**
  - account_id – (Required) Account ID.
  - note_id – (Required) Note ID to remove.

vectra-account-note-list

- **Purpose:** List all notes on an account.
- **Description:** Returns all notes for a specified account, including IDs, timestamps, and authors.
- **Input Arguments:**
  - account_id – (Required) Account ID.

vectra-account-markall-detections-asfixed

- **Purpose:** Mark all detections on an account as fixed.
- **Description:** Flags all active detections for the given account as fixed in Vectra.
- **Input Arguments:**
  - account_id – (Required) Account ID.

vectra-account-markall-detections-asclosed

- **Purpose:** Close all detections on an account.
- **Description:** Closes all active detections for the specified account with the chosen close reason.
- **Input Arguments:**
  - account_id – (Required) Account ID.

- close_reason – (Required) Close reason (for example, benign or remediated).

vectra-host-describe

- **Purpose:** Get details for a specific host.
- **Description:** Returns metadata, scores, tags, and URLs for a single host.
- **Input Arguments:**
  - id – (Optional) Host ID.

vectra-host-add-tags

- **Purpose:** Tag a host.
- **Description:** Adds one or more tags to the specified host.
- **Input Arguments:**
  - id – (Optional) Host ID.
  - tags – (Optional) Tags (comma-separated).

vectra-host-del-tags

- **Purpose:** Remove tags from a host.
- **Description:** Deletes one or more tags from the specified host.
- **Input Arguments:**
  - id – (Optional) Host ID.
  - tags – (Optional) Tags (comma-separated).

vectra-host-tag-list

- **Purpose:** List tags on a host.
- **Description:** Returns all tags currently associated with the host.
- **Input Arguments:**
  - id – (Required) Host ID.

vectra-host-note-add

- **Purpose:** Add a note to a host.
- **Description:** Creates a new note on the specified host and returns note metadata.
- **Input Arguments:**
  - host_id – (Required) Host ID.
  - note – (Required) Note text.

vectra-host-note-update

- **Purpose:** Update an existing host note.
- **Description:** Updates an existing note on a host while preserving audit fields.
- **Input Arguments:**
  - host_id – (Required) Host ID.
  - note_id – (Required) Note ID.
  - note – (Required) New note text.

vectra-host-note-remove

- **Purpose:** Remove a host note.
- **Description:** Deletes a note from the specified host.
- **Input Arguments:**
  - host_id – (Required) Host ID.
  - note_id – (Required) Note ID.

vectra-host-note-list

- **Purpose:** List all notes on a host.
- **Description:** Returns all notes on the specified host, including IDs and timestamps.
- **Input Arguments:**
  - host_id – (Required) Host ID.

vectra-host-markall-detections-asfixed

- **Purpose:** Mark all detections on a host as fixed.
- **Description:** Sets all active detections associated with the host to fixed.
- **Input Arguments:**
  - host_id – (Required) Host ID.

vectra-host-markall-detections-asclosed

- **Purpose:** Close all detections on a host.
- **Description:** Closes all active detections on the host with the specified reason.
- **Input Arguments:**
  - host_id – (Required) Host ID.
  - close_reason – (Required) Close reason.

vectra-detection-describe

- **Purpose:** Get details for a specific detection.
- **Description:** Returns full detection metadata, including category, type, scores, timestamps, and

related entities.

- **Input Arguments:**
  - id – (Optional) Detection ID.

vectra-detection-get-pcap

- **Purpose:** Download detection PCAP.
- **Description:** Retrieves the packet capture file associated with a detection, when available.
- **Input Arguments:**
  - id – (Optional) Detection ID whose PCAP should be retrieved.

vectra-detection-markasfixed

- **Purpose:** Mark/unmark a detection as fixed.
- **Description:** Sets the fixed flag for the specified detection based on the fixed argument.
- **Input Arguments:**
  - id – (Optional) Detection ID.
  - fixed – (Optional) true or false.

vectra-detection-add-tags

- **Purpose:** Tag a detection.
- **Description:** Adds tags to the specified detection.
- **Input Arguments:**
  - id – (Optional) Detection ID.
  - tags – (Optional) Tags (comma-separated).

vectra-detection-del-tags

- **Purpose:** Remove tags from a detection.
- **Description:** Removes the specified tags from a detection.
- **Input Arguments:**
  - id – (Optional) Detection ID.
  - tags – (Optional) Tags (comma-separated).

vectra-detection-tag-list

- **Purpose:** List tags on a detection.
- **Description:** Returns all tags associated with a given detection ID.
- **Input Arguments:**
  - id – (Required) Detection ID.

vectra-detection-note-add

- **Purpose:** Add a note to a detection.
- **Description:** Creates a new note for the specified detection and returns its metadata.
- **Input Arguments:**
  - detection_id – (Required) Detection ID.
  - note – (Required) Note text.

vectra-detection-note-update

- **Purpose:** Update a detection note.
- **Description:** Updates the content of an existing note on a detection.
- **Input Arguments:**
  - detection_id – (Required) Detection ID.
  - note_id – (Required) Note ID (obtainable via vectra-detection-note-list).
  - note – (Required) Updated note text.

vectra-detection-note-remove

- **Purpose:** Remove a detection note.
- **Description:** Deletes the specified note from a detection.
- **Input Arguments:**
  - detection_id – (Required) Detection ID.
  - note_id – (Required) Note ID.

vectra-detection-note-list

- **Purpose:** List all notes for a detection.
- **Description:** Returns all notes attached to the specified detection, including authors and timestamps.
- **Input Arguments:**
  - detection_id – (Required) Detection ID.

vectra-detections-mark-asclosed

- **Purpose:** Close one or more detections.
- **Description:** Marks the provided detection IDs as closed with the given close reason.
- **Input Arguments:**
  - detection_ids – (Required) One or more detection IDs (comma-separated).
  - close_reason – (Required) Close reason (for example, benign or remediated).

vectra-detections-mark-asopen

- **Purpose:** Reopen detections.

- **Description:** Reopens the detections specified by ID so they appear as active again.
- **Input Arguments:**
  - detection_ids – (Required) One or more detection IDs (comma-separated).

vectra-outcome-describe

- **Purpose:** Get details for a specific outcome.
- **Description:** Returns metadata for a single assignment outcome, including category and title.
- **Input Arguments:**
  - id – (Optional) Outcome ID.

vectra-outcome-create

- **Purpose:** Create a new assignment outcome.
- **Description:** Creates a custom outcome that can be used when resolving assignments.
- **Input Arguments:**
  - title – (Optional) Outcome title (displayed in the UI).
  - category – (Optional) Outcome category (Benign True Positive, Malicious True Positive, False Positive).

vectra-assignment-describe

- **Purpose:** Get details for a specific assignment.
- **Description:** Returns full details for an assignment, including linked entity, outcome, and resolution status.
- **Input Arguments:**
  - id – (Optional) Assignment ID.

vectra-assignment-assign

- **Purpose:** Assign an account/host for investigation.
- **Description:** Assigns an account or host to a Vectra user; if an assignment already exists, it is reassigned.
- **Input Arguments:**
  - assignee_id – (Optional) Vectra user ID to assign to.
  - assignment_id – (Optional) Existing assignment ID (for reassignment).
  - account_id – (Optional) Account ID.
  - host_id – (Optional) Host ID.

vectra-assignment-resolve

- **Purpose:** Resolve an assignment.

- **Description:** Resolves an assignment either for all detections or for a filtered subset using rules or explicit detection lists.
- **Input Arguments:**
  - assignment_id – (Optional) Assignment ID.
  - outcome_id – (Optional) Outcome ID to apply.
  - note – (Optional) Resolution note.
  - detections_filter – (Optional) Whether to filter detections (None or Filter Rule).
  - filter_rule_name – (Optional) Name of the filter rule (when using filter-rule mode).
  - detections_list – (Optional) List of detection IDs to include.

vectra-user-describe

- **Purpose:** Get details for a specific user.
- **Description:** Returns account information for a Vectra user, including role, type, and last login.
- **Input Arguments:**
  - id – (Optional) Vectra user ID.

vectra-group-list

- **Purpose:** List groups and optionally filter them.
- **Description:** Returns groups of type account, host, IP, or domain, with optional filters on members, importance, timestamps, and name.
- **Input Arguments:**
  - group_type – (Optional) Group type (account, host, ip, domain).
  - account_names – (Optional) Account names (comma-separated; for account groups).
  - domains – (Optional) Domains (comma-separated; for domain groups).
  - host_ids – (Optional) Host IDs (comma-separated; for host groups).
  - host_names – (Optional) Host names (comma-separated; for host groups).
  - importance – (Optional) Group importance (high, medium, low, never_prioritize).
  - ips – (Optional) IPs (comma-separated; for IP groups).
  - description – (Optional) Filter by group description.
  - last_modified_timestamp – (Optional) Only groups modified on/after this timestamp (supports relative/absolute formats).
  - last_modified_by – (Optional) Filter by last modifying user ID.
  - group_name – (Optional) Filter by group name.

vectra-group-assign

- **Purpose:** Add members to a group.
- **Description:** Assigns hosts, accounts, IPs, or domains to an existing Vectra group.
- **Input Arguments:**
  - group_id – (Required) Group ID (from vectra-group-list).
  - members – (Required) Comma-separated list of member identifiers appropriate for the group

type (host IDs, account names, IPs, or domains).

vectra-group-unassign

- **Purpose:** Remove members from a group.
- **Description:** Unassigns the provided members from the given group and returns the updated group details.
- **Input Arguments:**
    - group_id – (Required) Group ID (from vectra-group-list).
    - members – (Required) Comma-separated list of member identifiers to remove (host IDs, account names, IPs, or domains depending on type).

Playbooks

Playbooks are used to define automation flows and can consist of multiple actions as well instructions for interfacing with other apps configured in the XSOAR environment. There are some playbooks included as they are used with the integration. These playbooks will be covered later but the included playbooks also serve as a starting point that demonstrate key techniques. The operator can take specific techniques from the included playbooks to create their own automations based on their individual use cases. The included playbooks, their descriptions, and the techniques demonstrated are outlined in Table 2. A deeper dive into playbooks is covered later in this document.

Table 2 - Included Playbooks

Playbook Name	Description	Techniques Demonstrated
Process Incident – Vectra Detect (previously *Prepare Incident*)	Starting point to demonstrate workflow to manage incidents. Changes the incident from pending to active, lists available Vectra users, and prompts for the user ID to assign before handing off to Dispatch.	Changing incident state, prompting for input, collecting context, calling another playbook and passing data to it
Dispatch Incident – Vectra Detect	Prepare Detections and assign to analyst. Fetches all active Detections for the entity, checks for an existing assignment, then assigns or reassigns and adds a note in Vectra.	Fetch Detection details, adding or updating assignment, adding notes
Add Note – Vectra Detect	Looks up the entity, checks if it's a host or account, and writes a note to the entity in Vectra based on its type.	Attribute lookup, condition check, adding notes
Close All Duplicate XSOAR Incidents – Vectra Detect	Cleans up incidents in XSOAR by finding and closing duplicate incidents originating from Vectra Detect; can also close the corresponding assignment in Vectra.	Operational playbook to use as constructed, scheduled/recurring cleanup, calling a sub-playbook
Close Duplicate XSOAR Incidents – Vectra Detect	Child playbook called by *Close All Duplicate XSOAR Incidents – Vectra Detect* that closes duplicate XSOAR incidents and resolves their assignments in Vectra.	Automated duplicate handling, resolving assignments in Vectra, context cleanup (DeleteContext, helper scripts)

Operations

Incident Creation Philosophy

Best practice: Generate incidents on an entity-by-entity basis versus Detection-by-Detection.

Why: A key pillar of Vectra AI's value proposition to organizations is SOC efficiency. Vectra accomplishes this by attributing behavioral Detections to entities (currently, hosts & accounts), by leveraging AI to compute an urgency score that considers multiple factors such as Detections, velocity of progression, significance of the entity itself, and finally bringing all Detection and non-Detection context together in one prioritized place. For this reason, we promote the generation of incidents based on entities in external tools to mirror the Vectra AI Platform value proposition which results in decreased ticket volume, alert fatigue, and false positives. The following tables show a real-world difference between a Detection-centric (Table 3) approach (which competitors employ) to an entity-centric (Table 4) approach with Vectra. The result is an average reduction of 80% in ticket load, all while being laser-focused on what is most urgent.

Table 3 - Detection Centric

Month	# of Detections	Avg # of Tickets / Day
June 2023	418	14
July 2023	472	15
Aug 2023	762	25

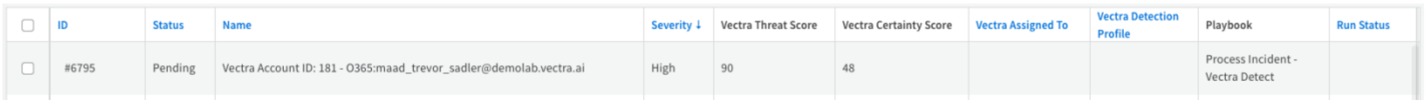
Table 4 - Entity Centric

Month	# of Entities	Avg # of Tickets / Day	% change of tickets created
June 2023	107	4	-74%
July 2023	107	3	-77%
Aug 2023	88	3	-88%

With Vectra's entity-centric prioritization, the Detections are still available and relevant but instead of managing each Detection as an isolated incident, they are managed holistically at the entity.

Incident Priority

The recommended starting point is to configure the incident list column order to match figure 11. With that order defined, sort the Severity column from highest to lowest. Incidents should then be investigated from top down. The default filter `-status:closed -category:job` will not show any closed incidents so the incidents will either be "Active" or "Pending". All incidents start as pending and when you select that incident ID, it will automatically initiate a playbook and will turn active. After sorting, the first incident in our list that requires attention is #6795 (Figure 16).

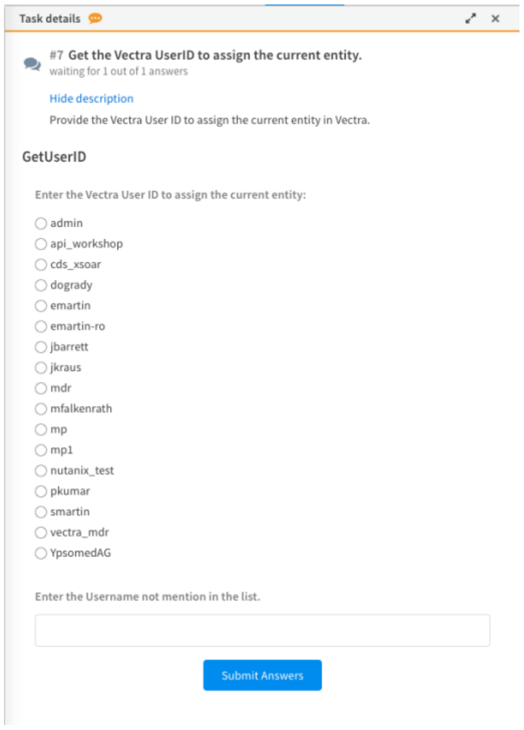


ID	Status	Name	Severity ↓	Vectra Threat Score	Vectra Certainty Score	Vectra Assigned To	Vectra Detection Profile	Playbook	Run Status
#6795	Pending	Vectra Account ID: 181 - O365:maad_trevor_sadler@demolab.vectra.ai	High	90	48			Process Incident - Vectra Detect	

Figure 16 - Incident Priority

Investigate

Selecting the pending incident will initiate the investigate process which will open the incident layout. The initial tab contains all the incident info details grouped into various sections. There is a section in the incident info layout titled Work Plan and upon first run there should be one work item waiting user attention (Figure 17). The work plan is waiting for the operator to provide the user ID (in Vectra) to assign the incident to. It's possible to view in Task or Work Plan but viewing in the War Room is preferred.



#7 Get the Vectra UserID to assign the current entity.
waiting for 1 out of 1 answers

Provide the Vectra User ID to assign the current entity in Vectra.

GetUserID

Enter the Vectra User ID to assign the current entity:

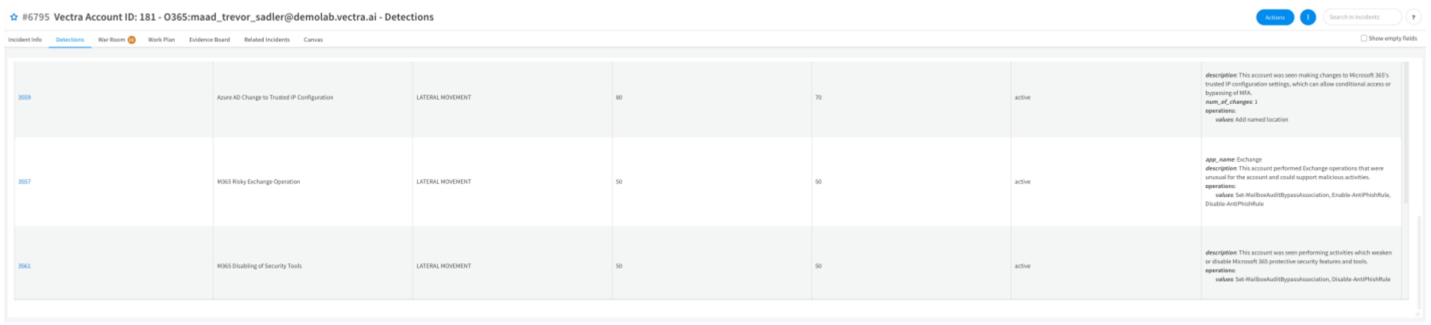
- admin
- api_workshop
- cds_xsoar
- dogrady
- emartin
- emartin-ro
- jbarrett
- jkraus
- mdr
- mfalkenrath
- mp
- mp1
- nutanix_test
- pkumar
- smartin
- vectra_mdr
- YpsomedAG

Enter the Username not mention in the list.

Figure 17 - Work Plan

Detections

Prior to assigning (dispatching) the incident, the operator may wish to review the Detections associated with the incident. The Detection tab will outline all the Detections (Figure 18). Selecting any Detection ID will spawn a new tab directed at the specific Detection inside the Vectra UI so the operator can review specific details.



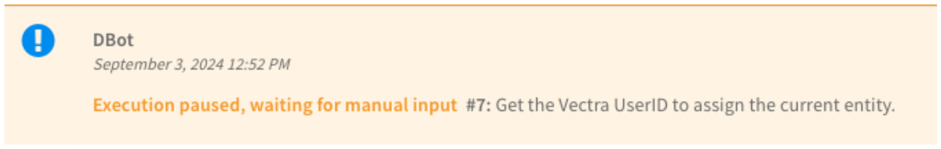
#6795 Vectra Account ID: 181 - O365:maad_trevor_sadler@demolab.vectra.ai - Detections

3559	Azure AD Change to Trusted IP Configuration	LATERAL MOVEMENT	80	70	active	***description***: This account was seen making changes to Microsoft 365's trusted IP configuration settings, which can allow conditional access or bypassing of MFA. ***num_of_changes***: 1 **operations:** **values**: Add named location
3557	M365 Risky Exchange Operation	LATERAL MOVEMENT	50	50	active	***app_name***: Exchange ***description***: This account performed Exchange operations that were unusual for the account and could support malicious activities. **operations:** **values**: Set-MailboxAuditBypassAssociation, Enable-AntiPhishRule, Disable-AntiPhishRule
3561	M365 Disabling of Security Tools	LATERAL MOVEMENT	50	50	active	***description***: This account was seen performing activities which weaken or disable Microsoft 365 protective security features and tools. **operations:** **values**: Set-MailboxAuditBypassAssociation, Disable-AntiPhishRule

Figure 18 - Detections

War Room

Loading the War Room and scrolling to the bottom (most recent information) the operator will notice that an execution has been paused, waiting for manual input. This is the task to assign the Vectra user to the incident (Figure 19). The operator can complete in the task pane or navigate to the Work Plan to follow the workflow.



DBot
September 3, 2024 12:52 PM
Execution paused, waiting for manual input **#7:** Get the Vectra UserID to assign the current entity.

Figure 19 - User Assignment

Work Plan

The initial playbook to process the incident begins with incident investigation. The playbook runs automatically until it reaches a block where it requests input from the operator. After the operator supplies a user ID for assignment, the playbook will continue (Figure 20).

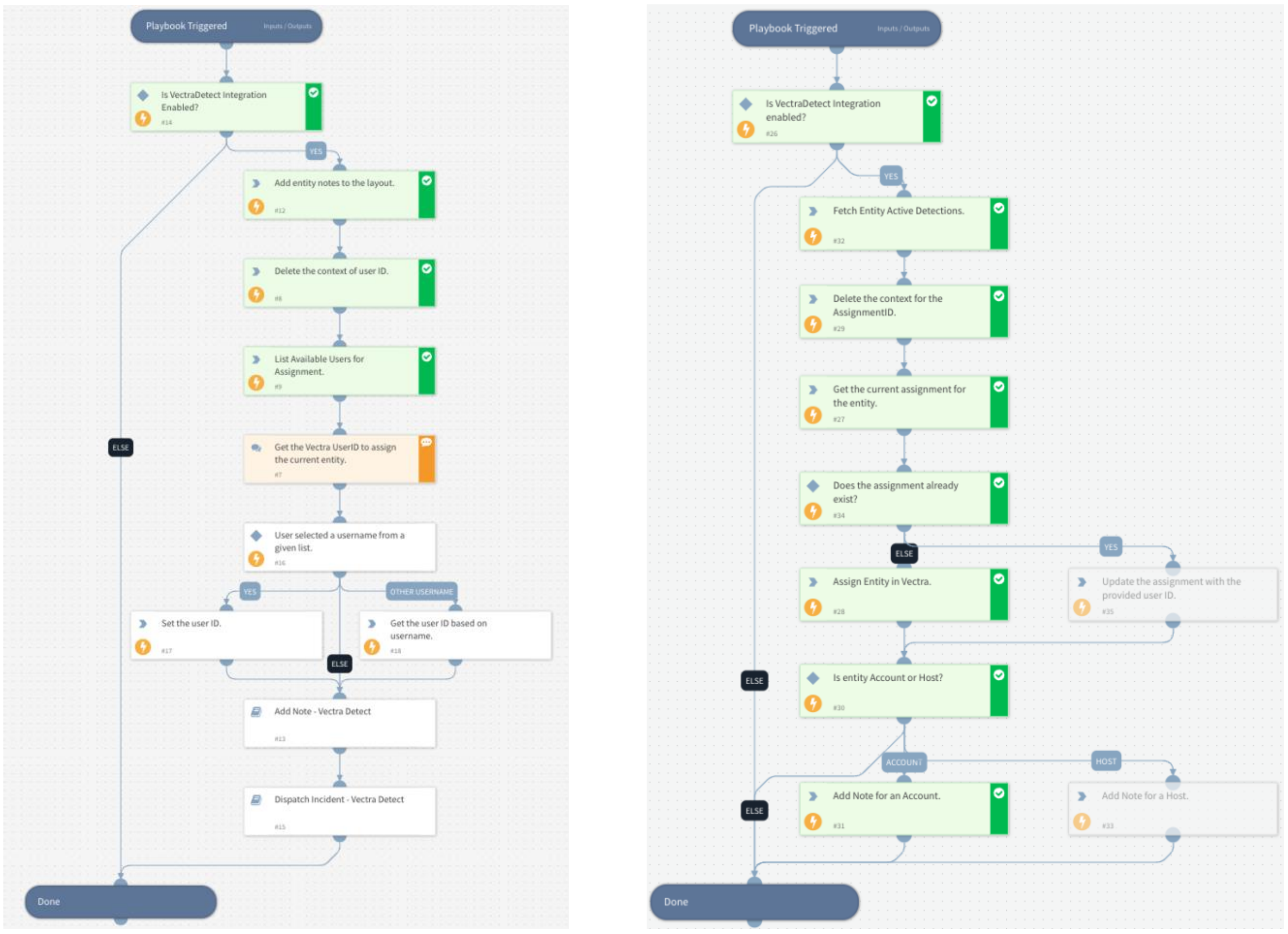

Playbook Triggered
Inputs / Outputs
Is VectraDetect Integration Enabled?
YES
ELSE
Add entity notes to the layout.
Delete the context of user ID.
List Available Users for Assignment.
Get the Vectra UserID to assign the current entity.
User selected a username from a given list.
YES
OTHER USERNAME
Set the user ID.
Get the user ID based on username.
Add Note - Vectra Detect
Dispatch Incident - Vectra Detect
Done
Playbook Triggered
Inputs / Outputs
Is VectraDetect Integration enabled?
YES
Fetch Entity Active Detections.
Delete the context for the AssignmentID.
Get the current assignment for the entity.
Does the assignment already exist?
ELSE
YES
Assign Entity in Vectra.
Update the assignment with the provided user ID.
Is entity Account or Host?
ACCOUNT
HOST
Add Note for an Account.
Add Note for a Host.
ELSE
Done

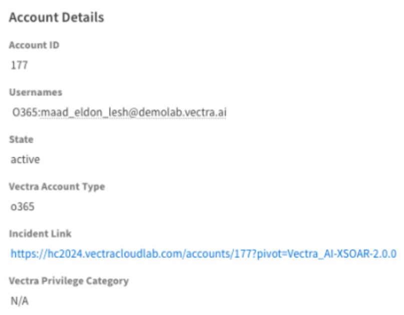
Figure 20 - Process Incident Work Plan

Assignment

The incident info page contains a section on assignment. The assigned user will automatically update on the next poll thanks to mirroring.

Running Actions - General

It's possible to run actions from within the incident container. Where the action is run from will depend on the required parameters, but the output of actions will be displayed in the 'War Room. For the most part, the action names include the artifact type they apply to. For instance, actions such as 'vectra-host-describe' or 'vectra-account-add-tags', apply to entity artifacts, whereas 'vectra-Detection-add-tags' or 'vectra-Detection-get-pcap', applies to Detection artifacts. Table 1 identifies the available actions and in the following example (Figure 23), I wish to describe an account entity, so I will need to provide the account ID.


Account Details

Account ID
177

Usernames
O365:maad_eldon_lesh@demolab.vectra.ai

State
active

Vectra Account Type
o365

Incident Link
https://hc2024.vectracloudlab.com/accounts/177?pivot=Vectra_AI-XSOAR-2.0.0

Vectra Privilege Category
N/A

Figure 23 - Action Requirements

The action is initiated from the command bar at the bottom of the screen. Once you start typing a command that is prefixed with "!", autocomplete will assist and it will also provide context to the required parameters (Figure 24).

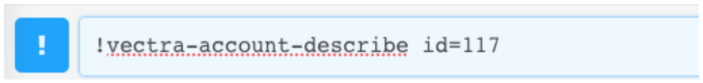
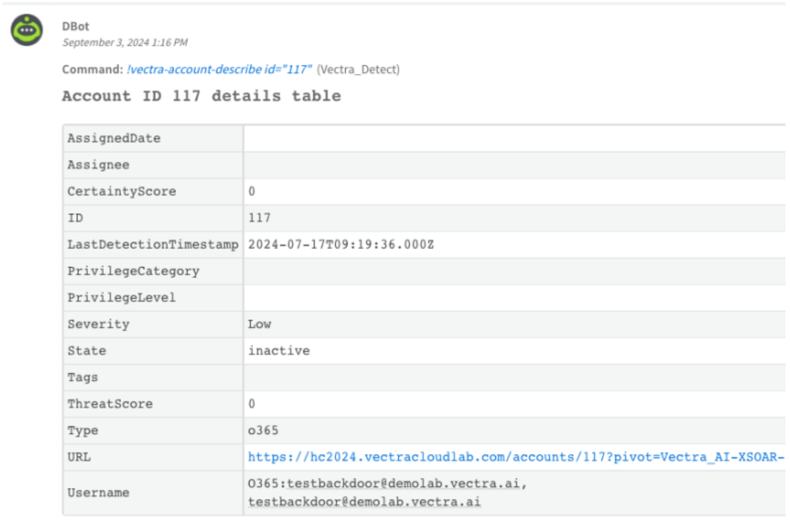

```
!vectra-account-describe id=117
```

Figure 24 - Run Action

The output of the action is displayed in the 'War Room (Figure 25).


DBot
September 3, 2024 1:16 PM

Command: !vectra-account-describe id="117" (Vectra_Detect)

Account ID 117 details table

AssignedDate	
Assignee	
CertaintyScore	0
ID	117
LastDetectionTimestamp	2024-07-17T09:19:36.000Z
PrivilegeCategory	
PrivilegeLevel	
Severity	Low
State	inactive
Tags	
ThreatScore	0
Type	o365
URL	https://hc2024.vectracloudlab.com/accounts/117?pivot=Vectra_AI-XSOAR-
Username	O365:testbackdoor@demolab.vectra.ai, testbackdoor@demolab.vectra.ai

Figure 25 - Action Output

Running Actions – Resolve Assignment

Most actions are self-explanatory and are relatively finite so it's not necessary to cover each one in detail. The action that requires the most explanation is the 'Resolve Assignment' action. This action covers several components and is very powerful, but it does require some prep work. When run, this action will take an entity that has been assigned and will mark it resolved including adding a resolution outcome, adding a note for the Vectra operational metrics report, optionally triage the Detections associated with the entity and place the desired label on the triaged Detections. The following table (Table 5) highlights the input that is required to run vectra-assignment-resolve (Figure 26) and where to find the relevant data.

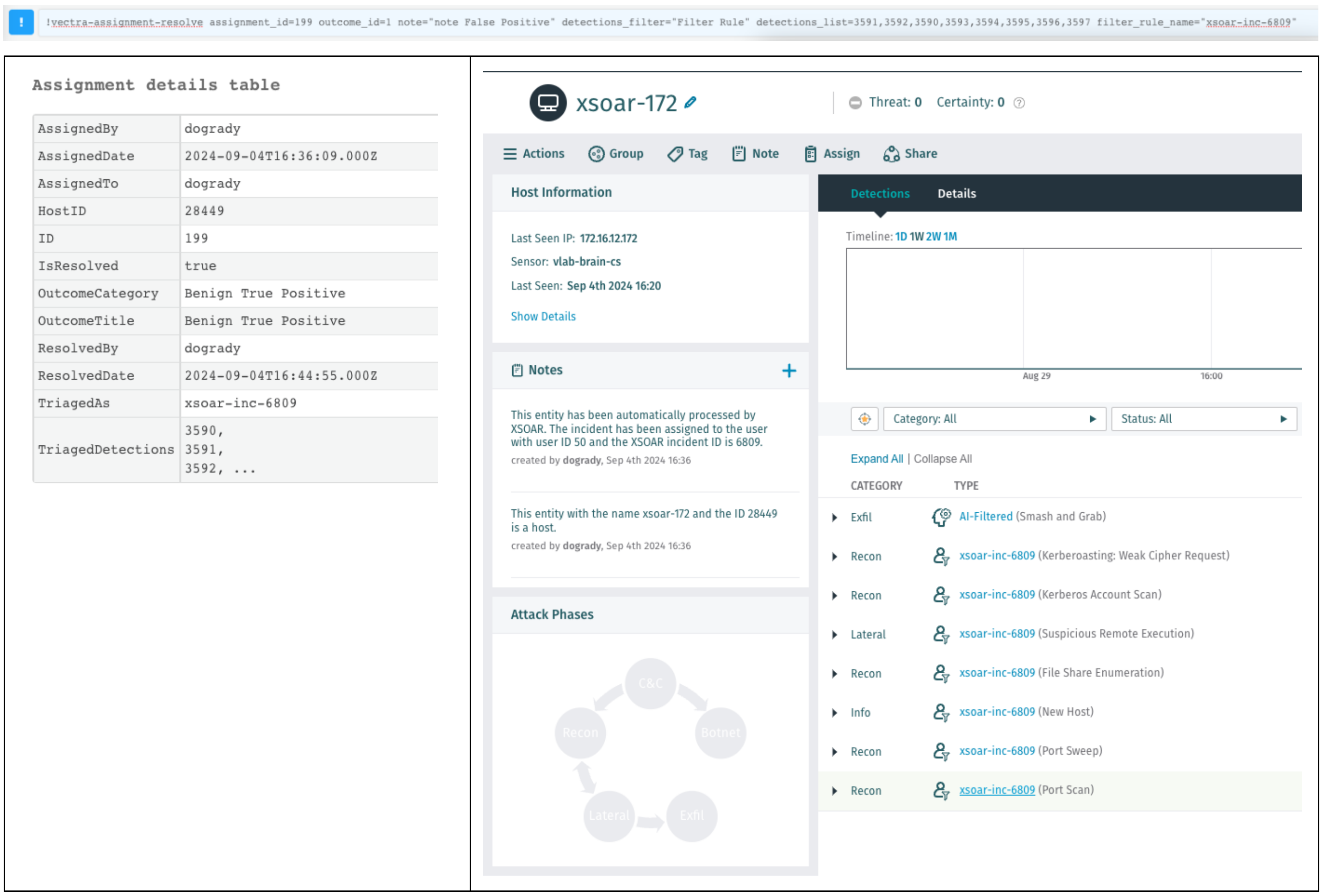
Table 5 - Resolve Assignment Parameters

Parameter	Format	Source
assignment_id	Single integer	Incident Info – entity assignment details
outcome_id	Integer	1-Benign True Positive, 2-Malicious True Positive, 3-False Positive
note	Free Form String	Free form – this note only appears in operational metrics report
Detections_filter	Two Options Only	"None" – nothing else required or "Filter Rule" – requires the following two parameters
filter_rule_name	Free Form String	Short label that shows up beside the triaged Detection in Vectra
Detections_list	Multiple integers csv	Run describe entity Detections action and obtain the data

When running vectra-assignment-resolve one or more Detections can be triaged when the assignment is resolved, or just the assignment can be resolved without touching the Detections. If the parameter Detection_filter = "None" is added, then Detection_list and filter_rule_name should not be provided, and this will resolve the assignment without touching the Detections. To close (triage) one or more Detections, use the parameter Detection_filter="Filter Rule" and supply values for filter_rule_name and Detections_list. Detections_list is the list of all Detection ID's to triage as part of the resolution.

```
!vectra-assignment-resolve assignment_id=199 outcome_id=1 note="note False Positive" Detections_filter="Filter Rule"
Detections_list=3591,3592,3590,3593,3594,3595,3596,3597 filter_rule_name="xsoar-inc-6809"
```

The command above, as well as Figure 26 show an example of running the vectra-assignment-resolve action. The details are seen in the war room and the results can be seen in Vectra.



```
!vectra-assignment-resolve assignment_id=199 outcome_id=1 note="note False Positive" detections_filter="Filter Rule" detections_list=3591,3592,3590,3593,3594,3595,3596,3597 filter_rule_name="xsoar-inc-6809"
```

Assignment details table

AssignedBy	dogrady
AssignedDate	2024-09-04T16:36:09.000Z
AssignedTo	dogrady
HostID	28449
ID	199
IsResolved	true
OutcomeCategory	Benign True Positive
OutcomeTitle	Benign True Positive
ResolvedBy	dogrady
ResolvedDate	2024-09-04T16:44:55.000Z
TriagedAs	xsoar-inc-6809
TriagedDetections	3590, 3591, 3592, ...

xsoar-172 Threat: 0 Certainty: 0

Actions Group Tag Note Assign Share

Host Information

Last Seen IP: 172.16.12.172
Sensor: vlab-brain-cs
Last Seen: Sep 4th 2024 16:20
Show Details

Notes

This entity has been automatically processed by XSOAR. The incident has been assigned to the user with user ID 50 and the XSOAR incident ID is 6809.
created by dogrady, Sep 4th 2024 16:36

This entity with the name xsoar-172 and the ID 28449 is a host.
created by dogrady, Sep 4th 2024 16:36

Attack Phases

CATEGORY	TYPE
Exfil	AI-Filtered (Smash and Grab)
Recon	xsoar-inc-6809 (Kerberoasting: Weak Cipher Request)
Recon	xsoar-inc-6809 (Kerberos Account Scan)
Lateral	xsoar-inc-6809 (Suspicious Remote Execution)
Recon	xsoar-inc-6809 (File Share Enumeration)
Info	xsoar-inc-6809 (New Host)
Recon	xsoar-inc-6809 (Port Sweep)
Recon	xsoar-inc-6809 (Port Scan)

Figure 26 - Resolve Assignment Action

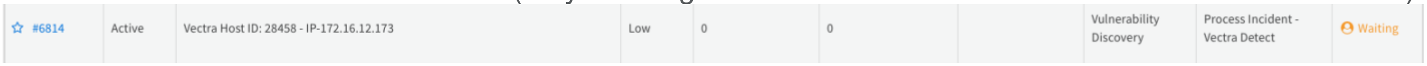
As a best practice, it's always advisable to try and close all Detections associated with an incident. When running the action manually, it's important to change the status of the incident in XSOAR to closed.

Cleanup Playbook

The integration application includes a special playbook that is used to cleanup empty incidents. Incidents can become empty if an aggressive polling cycle is used, and Vectra is attaching Detections to an entity before host ID names a host. These will most commonly be seen as hosts that start with IP-n.n.n.n. These hosts may be new devices on the network that trigger one or more Detections. Once Vectra can identify the host (Host ID process), a new Host record and the Detections from the IP-host will move to the properly named host. This is referred to as host ID merge and this results in the initial incident to be void of Detections as they all exist on the named host (incident). The cleanup playbook is designed to run as a job (but can also be run manually) and it seeks out empty incidents and closes them as duplicate, so they don't burden the SOC.

Palo Alto Networks requires all incidents to be in an active state (Status = Active) before it can be closed so this requires the operator to click on the incidents that show Threat Score = 0 and Certainty Score = 0 so their status changes to active. Incidents don't need to be assigned, and the Run Status can remain in the waiting state if desired.

During the next run of the automation job, incident #6814 from Figure 27 will be automatically closed because it has no Detections associated with it (they all merged into Host ID: xsoar-173 which is still a live incident).



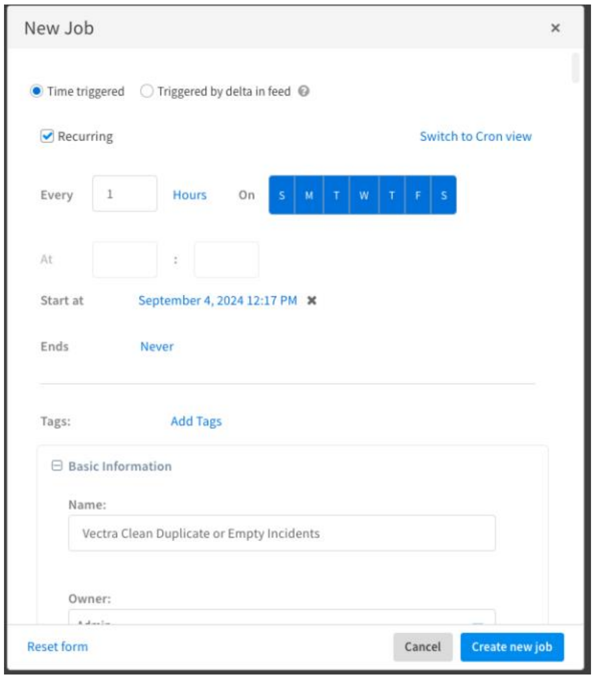
#6814	Active	Vectra Host ID: 28458 - IP-172.16.12.173	Low	0	0		Vulnerability Discovery	Process Incident - Vectra Detect	Waiting

Figure 27 – Vectra Close Duplicate Incidents

The playbook to clean up empty incidents should be configured to run as a job. To create a job, navigate to **Jobs** from XSOAR user interface and select **New Job**.

The key parameters for configuring this job (Figure 28) are as follows:

- Time triggered – Recurring each day of the week
- Recommend recurring every six hours
- Name: any relevant name
- Owner: admin
- Role: analyst
- Playbook: Close **<u>All</u>** Duplicate XSOAR Incidents – Vectra Detect
  note: The playbook with the similar name but without 'All' is the sub-playbook
- Queue Handling: Cancel the previous job and configure a new job



New Job

[✓ Time triggered] Triggered by delta in feed

[✓] Recurring — Switch to Cron view

Every: 1 Hours On: S M T W T F S

At: :

Start at: September 4, 2024 12:17 PM

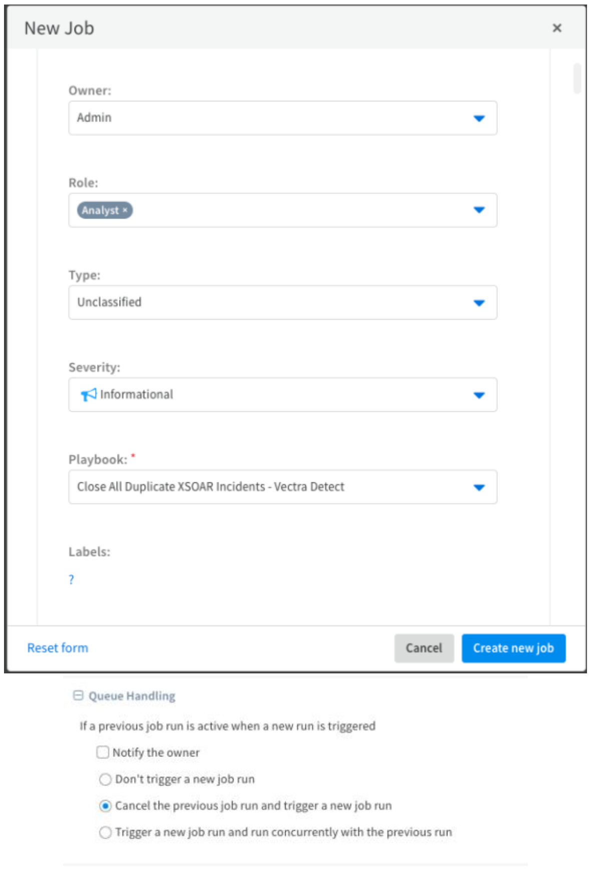
Ends: Never

Tags: Add Tags

Basic Information

Name: Vectra Clean Duplicate or Empty Incidents

Owner:



New Job

Owner: Admin

Role: Analyst

Type: Unclassified

Severity: Informational

Playbook: * Close All Duplicate XSOAR Incidents - Vectra Detect

Labels:

Queue Handling

If a previous job run is active when a new run is triggered

- [ ] Notify the owner
- Don't trigger a new job run
- [✓ Cancel the previous job run and trigger a new job run]
- Trigger a new job run and run concurrently with the previous run

Scroll to the very bottom of the configuration parameters and set the Queue Handling

Figure 28 – Vectra Close Duplicate Incidents Job Configuration

The playbook to close empty incidents is shown in Figure 29. The main playbook (left) calls the sub-playbook (right). The 'Close Duplicate Incidents' playbook task in the sub-playbook calls VectraDetectCloseDuplicateIncidents automation which can be reviewed under the Automation menu.

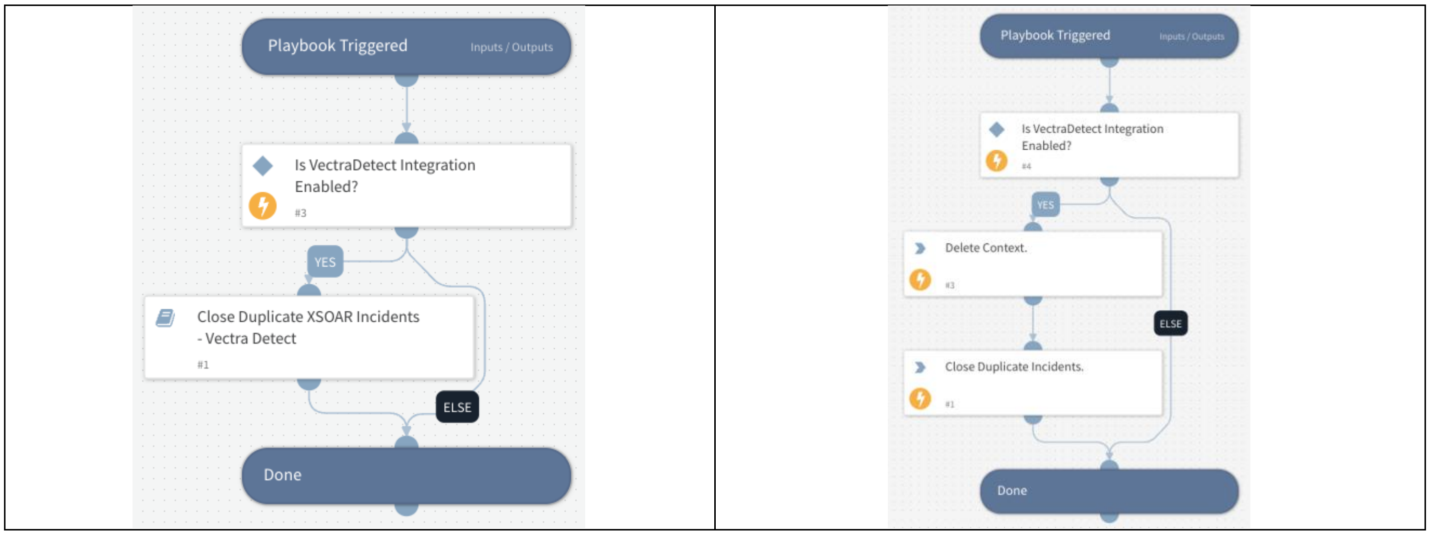
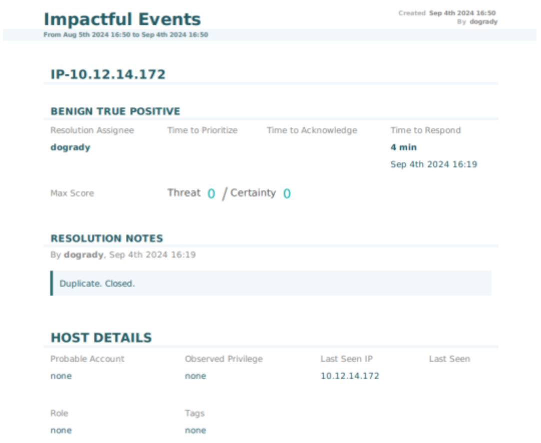

Playbook Triggered
Inputs / Outputs
Is VectraDetect Integration Enabled?
#3
YES
Close Duplicate XSOAR Incidents - Vectra Detect
#1
ELSE
Done
Playbook Triggered
Inputs / Outputs
Is VectraDetect Integration Enabled?
#4
YES
Delete Context.
#3
ELSE
Close Duplicate Incidents.
#1
Done

Figure 29 – Vectra Close Duplicate Incidents Playbook

When empty/duplicate incidents are close, the closure status is reflected in the Vectra Operational Metrics report with a resolution note of "Duplicate. Closed." As per Figure 30.



Impactful Events

Created **Sep 4th 2024 16:50**
By **dogrady**

From Aug 5th 2024 16:50 to Sep 4th 2024 16:50

IP-10.12.14.172

BENIGN TRUE POSITIVE

Resolution Assignee	Time to Prioritize	Time to Acknowledge	Time to Respond
dogrady			**4 min** Sep 4th 2024 16:19

Max Score Threat 0 / Certainty 0

RESOLUTION NOTES

By **dogrady**, Sep 4th 2024 16:19

Duplicate. Closed.

HOST DETAILS

Probable Account	Observed Privilege	Last Seen IP	Last Seen
none	none	10.12.14.172	
Role	Tags		
none	none		

Figure 30 – Vectra Operational Metrics Report

Troubleshooting

No incidents

If no events are displaying after configuration this could be the result of several things.

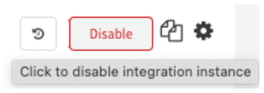
An incorrect API token could have been entered during configuration. Use the 'test connectivity' button under instance settings to validate.

The instance may not be configured to fetch incidents (set to 'Do not fetch').

The filters that have been configured restricts the data that is initially received and if the filters are too restrictive, it's possible there is no matching data. Modify the "Instance Settings" filters to poll all entities for prioritized and remove any other filters to isolate the issue.

Review the polling settings under "Incident Fetch Interval" to ensure that a polling interval or schedule has been set correctly.

The instance configuration itself may be disabled. When it's running it should show "Disable" as per the image below as clicking that icon will instruct the system to disable the configuration.



Playbook not running properly

When troubleshooting playbooks use "Playbook Debugger Panel" to isolate testing. In the debugger panel it's possible to select an existing incident to use as test data.

Worldwide Support Contact Information

- Support portal: https://support.vectra.ai/ (preferred contact method)
- Email: support@vectra.ai
- Additional information: https://www.vectra.ai/support