McAfee™
Together is power.

OpenDXL

Security Connected Solutions using Open APIs

Andres More (andres_more2@mcafee.com)
Damian Quiroga (damian_quiroga2@mcafee.com)

Andres More

Andres is a software architect at McAfee and lecturer at a local university on InfoSec. Andres holds a degree in CS and a MsC in High Performance Computing.

Andres started in security when implementing Multi-Level Security at the Linux's kernel network stack and after that worked in several roles on products and services at Intel until McAfee’s spin out.

At the moment leads McAfee´s Threat Intelligence Exchange and MVision EDR technical roadmaps, also collaborating in multiple enterprise solutions covering both protection and detection scenarios.

Damian Quiroga

Damian is a software security architect at McAfee. He holds a degree in Telecommunications Engineering.

Damian worked in several roles on products and services at Intel until McAfee's spin out.

He is currently the Software Security Architect for the McAfee Threat Intelligence Exchange product, leading the implementation of SDL processes and handling vulnerabilities reports.

Abstract

OpenDXL is an **open API** to enable devices to share intelligence and orchestrate security operations in **near real-time**. This **security connected platform** provides a unified framework for hundreds of products and services, which partners and customers already adopted in the field.

Any organization can **improve its security posture** and minimize operational costs through the platform's capabilities. This data exchange framework is used to build collective threat intelligence to make endpoint, network, and cloud countermeasures **protect and detect as one**.

The tutorial includes a review of the **problem-space** and **use cases** to be supported, then matching against the proposed platform capabilities including open APIs. After reviewing concrete examples of real-world integrations by partners and customers, we will discuss how to develop by leveraging current APIs and creating new ones. To complete the tutorial students will be challenged with **hands-on exercises** reusing samples from OpenDXL to add security-related behaviors to the platform.

InfoSec Integration APIs

Corporate Environment

Business Needs

Workstations, Servers, Branches, BYOD, Remote Access, Services & Applications, Email, Web, IM, Voice & Video, VoIP, Telepresence Rooms, CCTV

Non Functional Needs

Security, Management & Monitoring, Resilience, Availability

Regulatory Frameworks

Data Protection Regulations (GDPR), PCI, FedRAMP

Toolbox

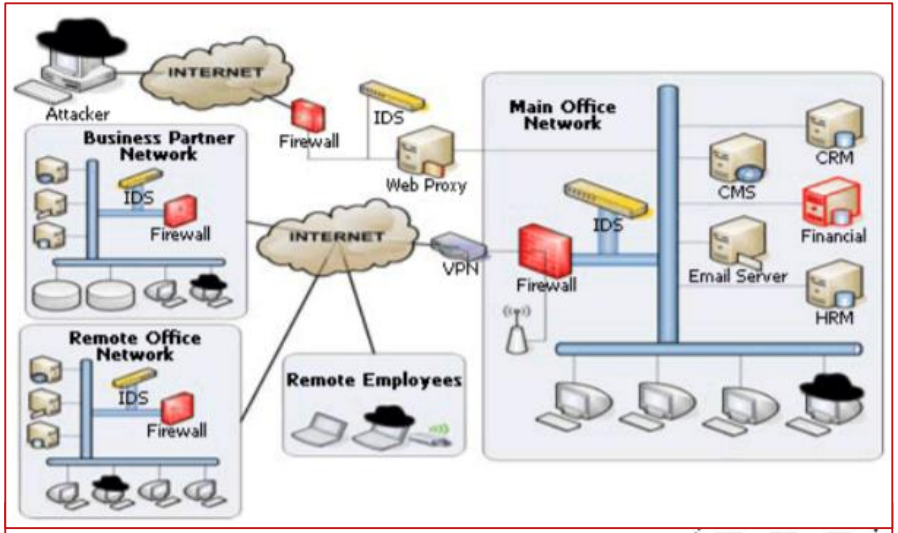
Firewalls: Packet & Stateful Filtering, App Layer.

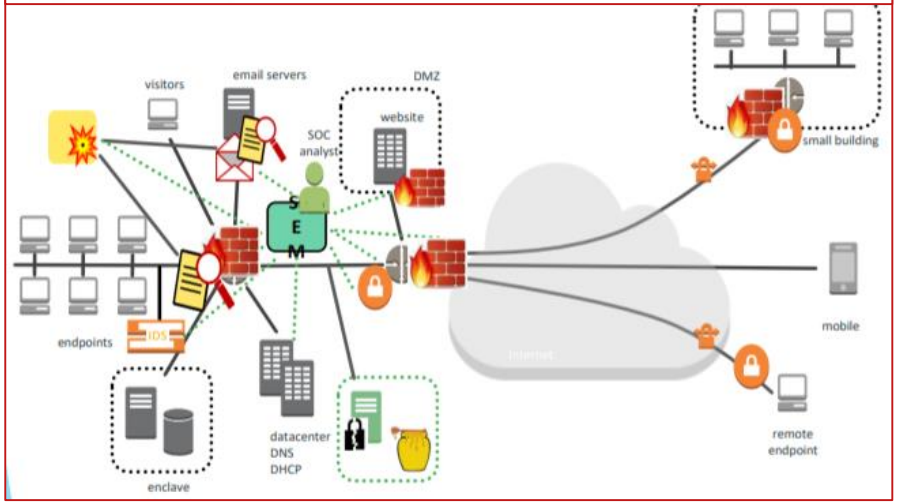
Gateways: Web & Email

IPSs & IDSs

VPN, DLP, EDR, SIEM

Sandboxing

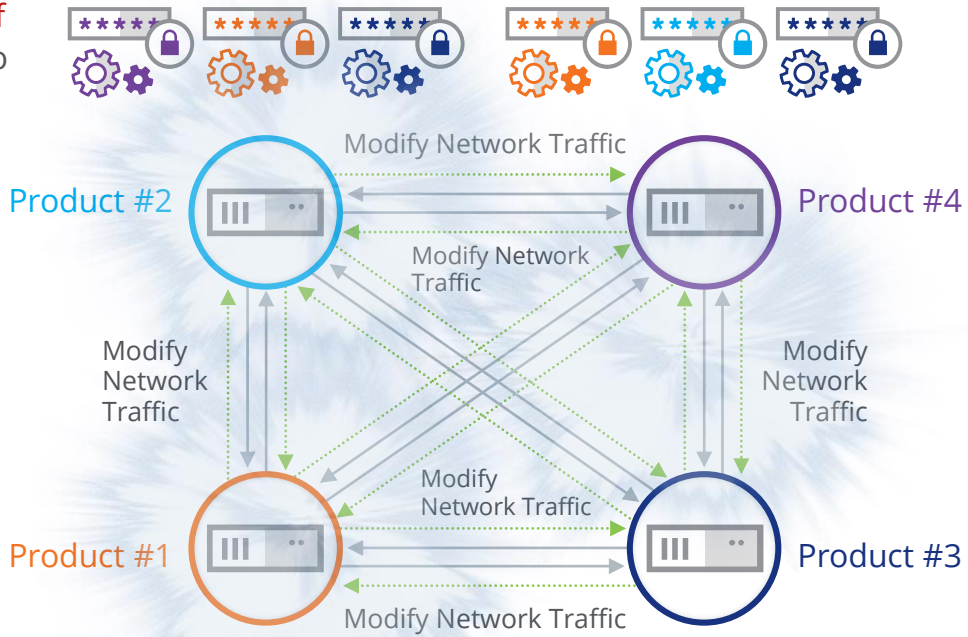




Integrating Products

A considerable amount of time and effort is taken to build out this complex system with multiple products.

When one product changes it's API (which happens), everything breaks and you are back to square one!



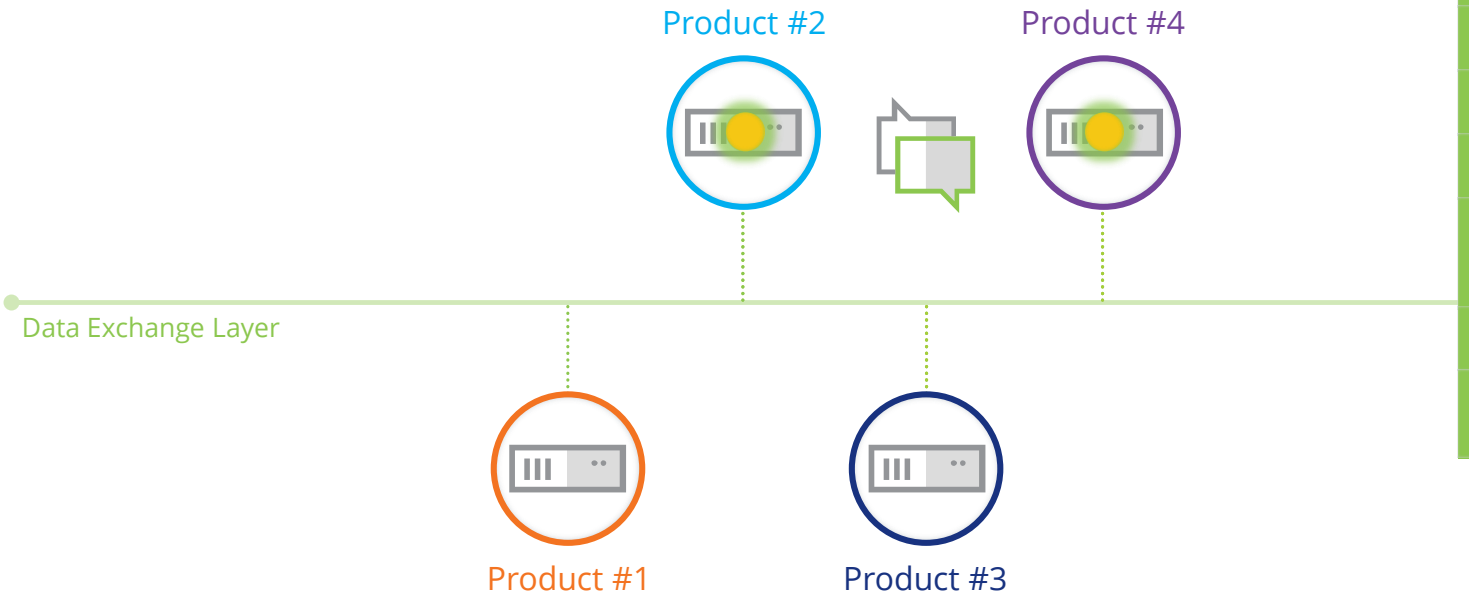
X	Distribute credentials
X	Integrate product APIs
X	Adjust network for ports/protocol
X	Poll for changes (simulate real-time events)

Credentials

API

Eliminating the Complexity

Multiple Communication modes



✓	All communication on single fabric and protocol
✓	Consistent API across products (DXL API)
✓	Decoupled (topic-based) communication
✓	Near real-time events (no polling)
✓	Centralized authentication and authorization model
✓	Persistent connections established from endpoints to fabric (Firewall friendly)
✓	One to many event-based communication
✓	One to one query response-based communication

Messaging Capabilities

What is it?

- Near real-time communication
  - Persistent connection
  - Firewall friendly
- Multiple messaging patterns
  - Event-based message broadcast (publish/subscribe)
  - Service-based request-response (question and answer)
    - Failover and load balancing
- Supported environments
  - SaaS
  - On-premise
  - Hosted (cloud)

Production Use Case Examples

- Real-time fabric-wide alerting
  - Object Reputation Changes
  - Sandboxing file reports
  - SIEM alerts
  - …
- Scalable fabric-wide services with failover
  - EDR data collection (traces, etc.)
  - Object Reputations
  - Adaptive Network Control
  - …
- Server-initiated real-time communication
  - Agent wake-up (on-premise and cloud)
  - EDR queries (cloud to endpoint)
  - Pro-active health check

…

Messaging Components

Broker

- Public/subscribe events
- Service management (REST-like over pub/sub)
- Directed messaging (specify brokers/clients)
- Console policy-driven
  - Bridging
  - Authentication/authorization
- Multi-tenancy
- WebSockets

Managed Clients

- C/C++
- Java

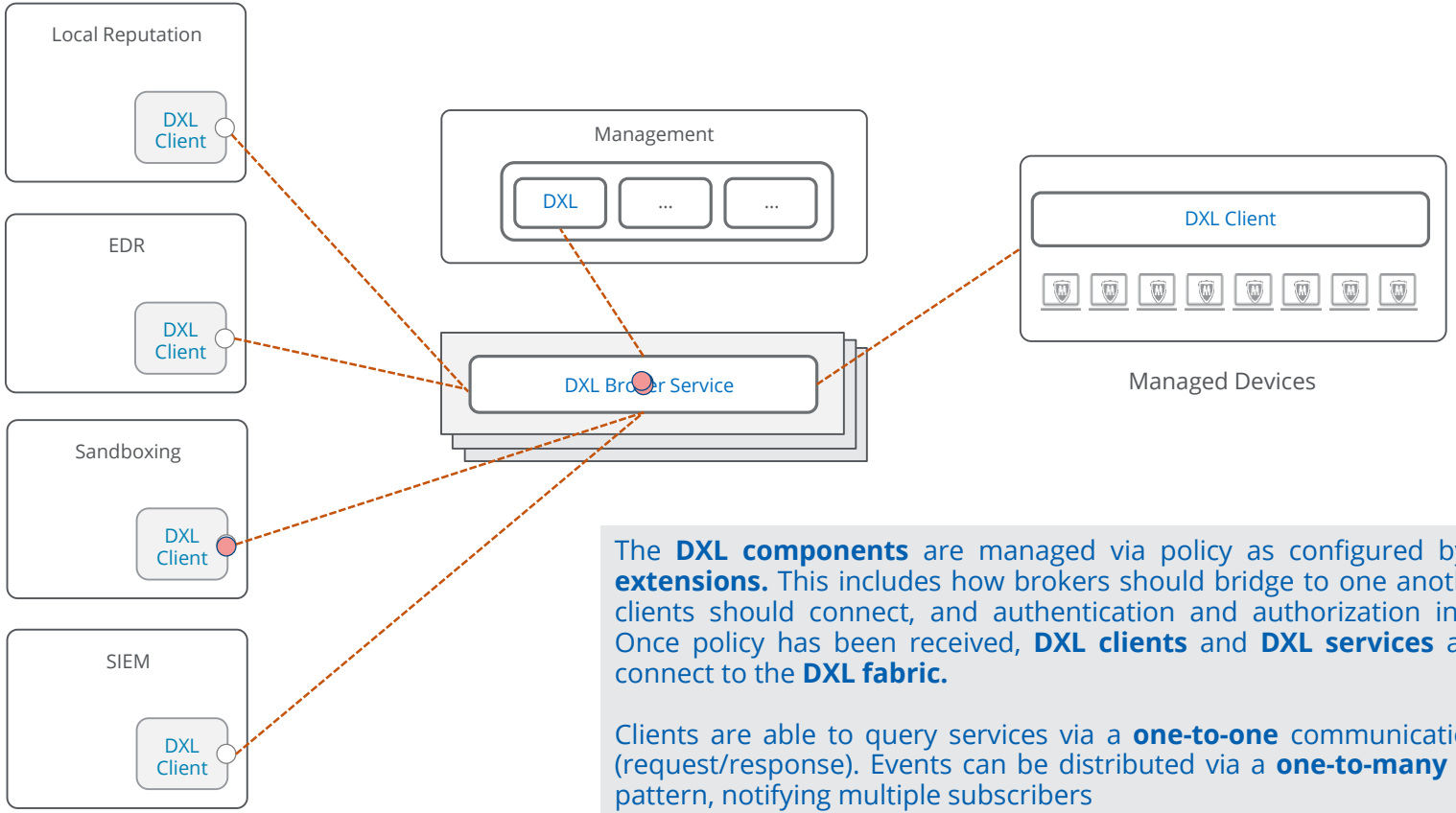
Open Clients

- Python
- Node
- Java
- Ruby

Management Extensions

- Management of brokers and clients
- Authentication and Authorization
- Policy delivery (to brokers and clients)
- Provisioning
  - Managed clients
  - Open clients
- Integration APIs
  - McAfee products
  - Partners
  - Customers
  - Community

Customer Environment



The **DXL components** are managed via policy as configured by the **DXL extensions.** This includes how brokers should bridge to one another, where clients should connect, and authentication and authorization information. Once policy has been received, **DXL clients** and **DXL services** are able to connect to the **DXL fabric.**

Clients are able to query services via a **one-to-one** communication pattern (request/response). Events can be distributed via a **one-to-many** messaging pattern, notifying multiple subscribers

OpenDXL

OpenDXL

What is it?

- Set of open source-related initiatives to drive the adoption of DXL-related technologies
- OpenDXL.com – Community hub
- +50 open source projects hosted on GitHub
  - OpenDXL Broker
  - OpenDXL Clients
  - OpenDXL Solutions
- Pre-built Docker images hosted in Docker Hub
- Distributed via standard repositories
  - PyPI (Python Package Index)
  - NPM (Node Package Manager)
  - Maven Central

OpenDXL (Cont'd)

Web Presence

- OpenDXL.com
- Source on GitHub
- Pre-built images on Docker Hub
- Testing with Travis CI

General Components

- Broker
- Console
- Environment
- Bootstrap

Clients

- Python
- Node
- Java
- Ruby

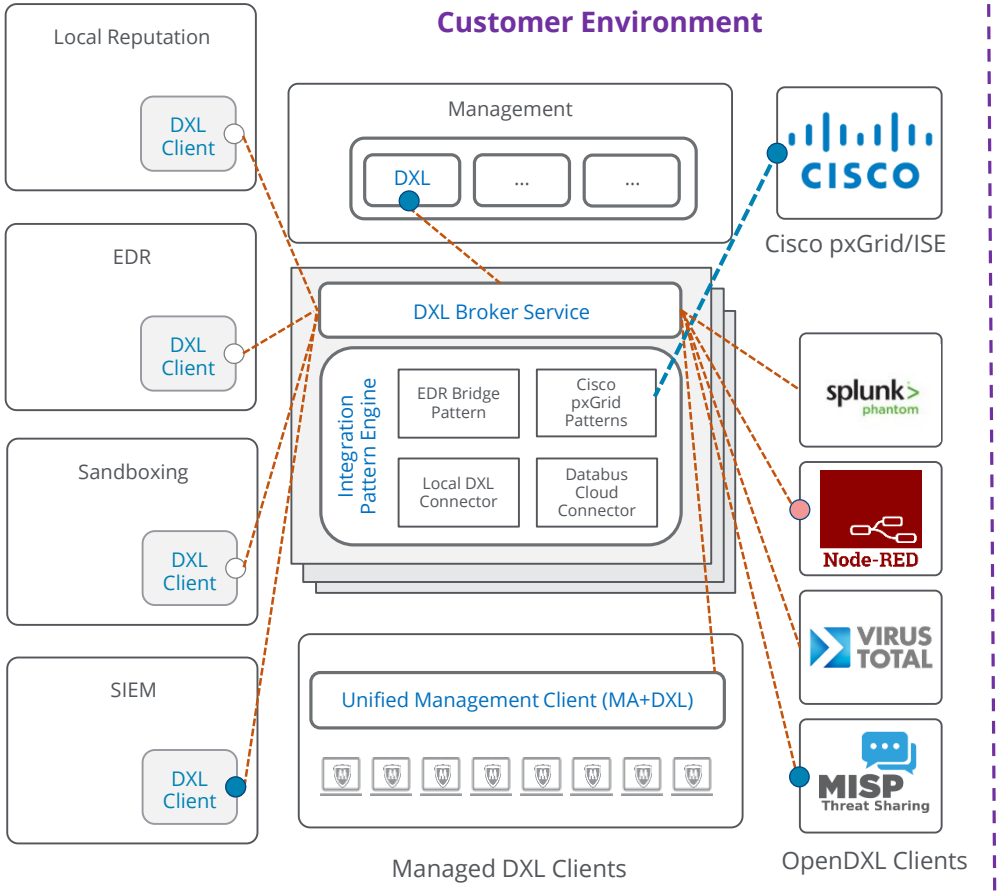
Services

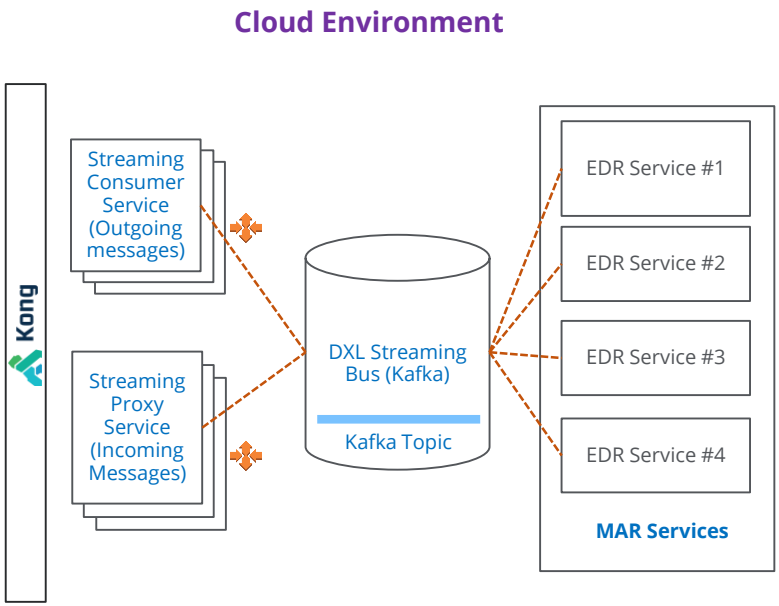
- Malware Information Sharing Platform (MISP)
- TheHive
- DomainTools
- VirusTotal
- Elasticsearch
- MaxMind (GeoIP)
- URLVoid
- Cuckoo Sandbox

Node-RED (Real-time flow engine)

- DXL-specific Docker Image
- Extensions
  - Core DXL
  - McAfee Active Reponse (MAR)
  - McAfee Threat Intelligence Exchange (TIE)
  - McAfee ePolicy Orchestrator (ePO)
  - Cisco pxGrid/ISE

OpenDXL Components





Solutions developed with **OpenDXL** clients are able to connect to both **McAfee-managed DXL fabrics** (on-premise and **MVISION ePO**) and **OpenDXL** fabrics. All of the clients in the connected **DXL messaging fabric** are able to communicate with each other. **Multiple messaging patterns are supported**. Request/response (one-to-one) and publish/subscribe (one-to-many)

Overview and History -> remove

Total downloads of OpenDXL components

Over 121k total downloads

Python Package Index (PyPI)

- **96,466** total downloads
- Top downloads:
  OpenDXL Python Client, OpenDXL Bootstrap, OpenDXL ePO Service

Docker Hub

- **15,916** total image pulls
- Top pulls: OpenDXL Broker, OpenDXL Environment, OpenDXL Console

Node Package Manager (NPM)

- **5,138** total downloads
- Top downloads: OpenDXL Node.js Client, OpenDXL Node.js Bootstrap

GitHub

- **3,614** packaged downloads (does not include clones)

Current as of Oct 02, 2018

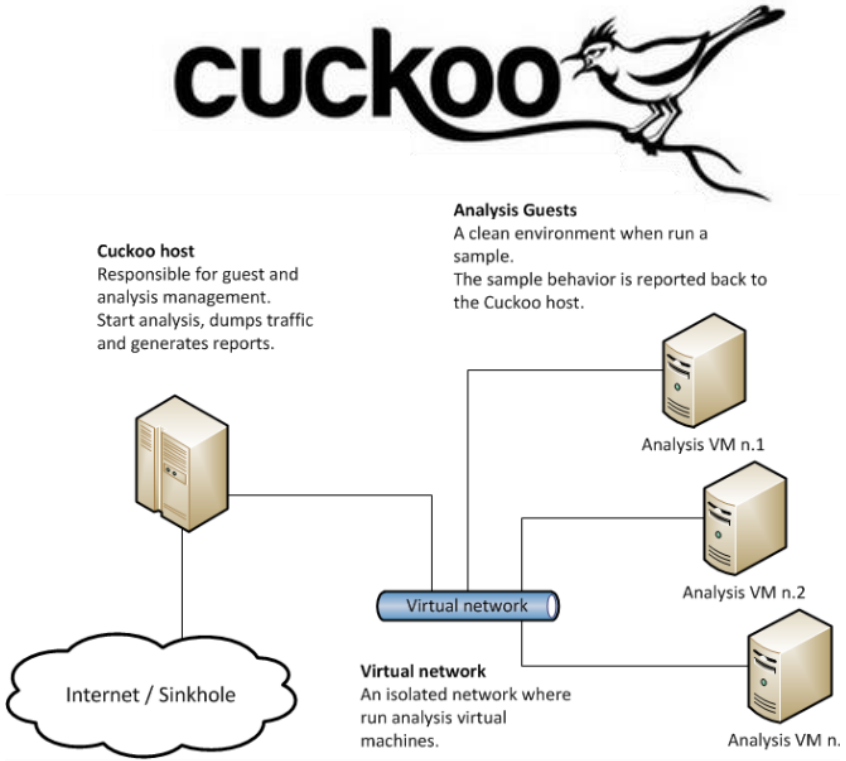
Images property of GitHub (github.com), Python Software Foundation (pypi.org), Docker Inc. (docker.com), and npm, inc. (npmjs.com)

Cuckoo OpenDXL Integration

Reports analysis results over DXL fabric

"Cuckoo Sandbox is the leading open source automated malware analysis system"

"...in a matter of minutes Cuckoo will provide a detailed report outlining the behavior of the file when executed inside a realistic but isolated environment"



Quotes and images property of Cuckoo Sandbox (cuckoosandbox.org)

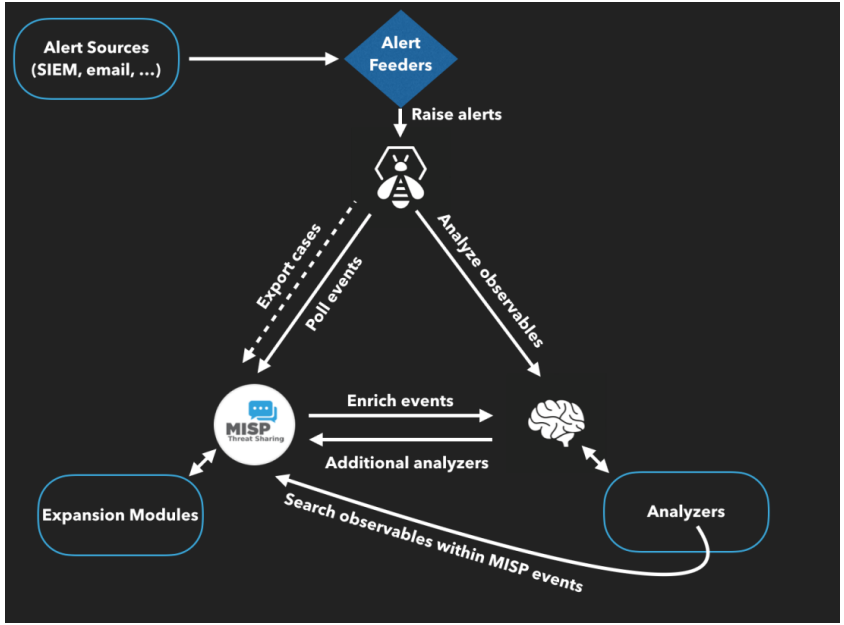
TheHive OpenDXL Integration

Create and search cases and alerts via DXL

"A scalable, open source and free Security Incident Response Platform, tightly integrated with MISP (Malware Information Sharing Platform)..."

"...designed to make life easier for SOCs, CSIRTs, CERTs and any information security practitioner dealing with security incidents that need to be investigated and acted upon swiftly."

"Add one, hundreds or thousands of observables to each case that you create or import them directly from a MISP event or any alert sent to the platform."



Quotes and images property of TheHive Project (thehive-project.org)

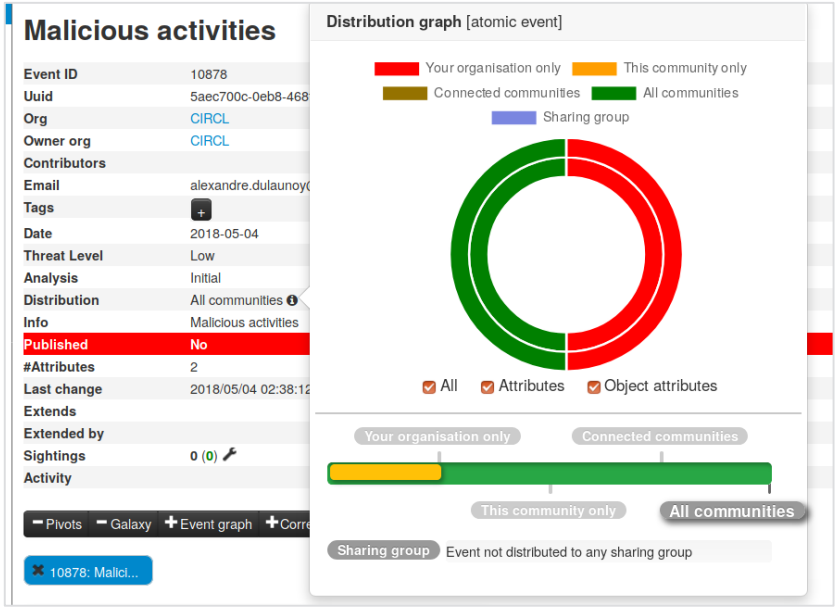
MISP OpenDXL Integration

Receive real-time notifications and manage events via DXL

“The MISP threat sharing platform is a free and open source software helping information sharing of threat intelligence including cyber security indicators.”

“Automatic correlation finding relationships between attributes and indicators from malware, attacks campaigns or analysis.”

“An intuitive user-interface for end-users to create, update and collaborate on events and attributes/indicators.”



Quotes and images property of MISP project (misp-project.org)

Node-RED Integrations

Open source orchestration with Node-RED and OpenDXL

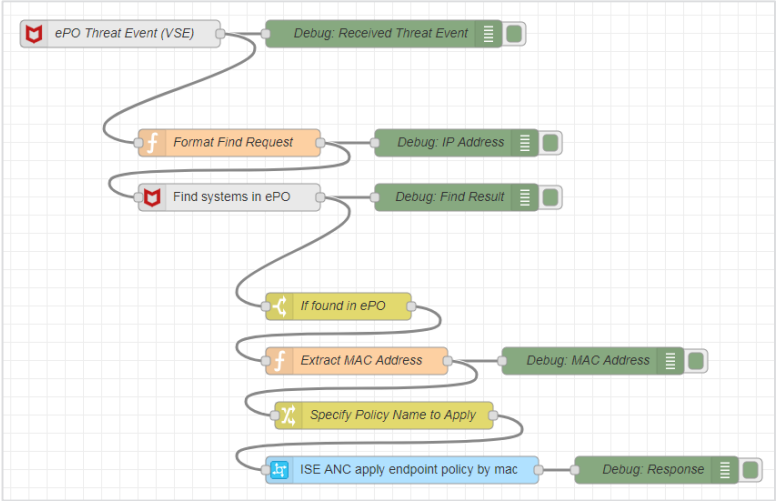
“Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways”

“… began as a proof-of-concept for visualizing and manipulating mappings between MQTT topics, quickly became a much more general tool that could be easily extended in any direction”

“… open-sourced in September 2013 and has been developed in the open ever since, culminating in it being one of the founding projects of the JS Foundation in October 2016”

“… consists of a Node.js-based runtime that you point a web browser at to access the flow editor”

Node-RED

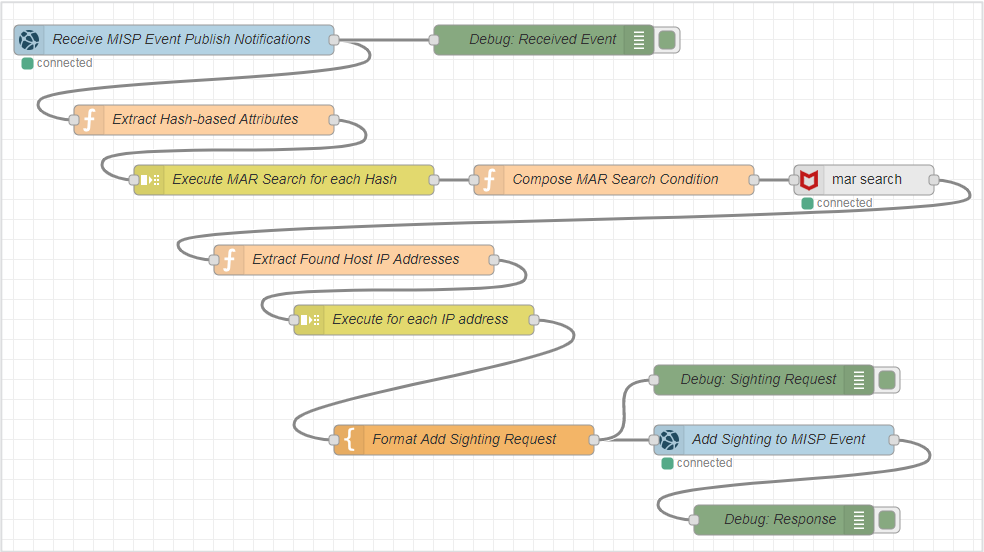


Quotes and images property of the JS Foundation (js.foundation)

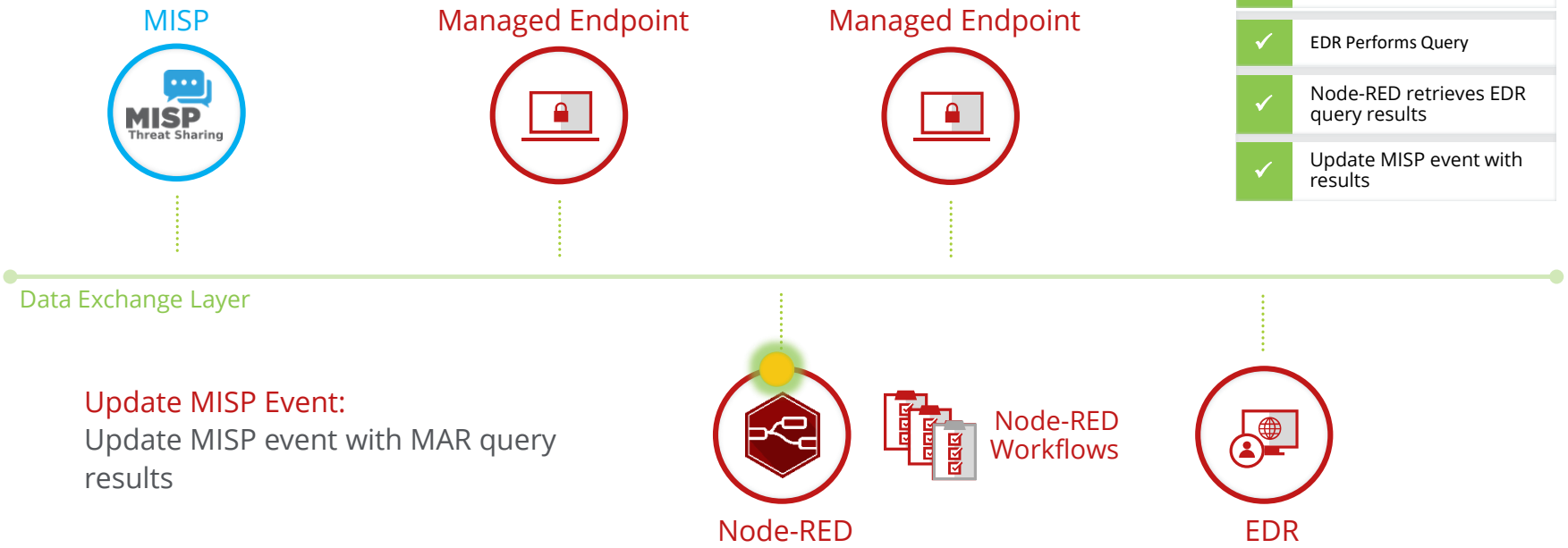
Scenario: Search for Infected Systems

As MISP events are published, search for systems that are infected

1. Problem to solve
   - Want to locate systems that have files that are associated with a MISP event
2. Products involved
   - MISP, EDR
3. What should happen
   - When a MISP event is received, search enterprise for infected systems. Update MISP event with results.
4. DXL Capabilities
   - Receive an event: Trigger workflow when a MISP event is published
   - Ask a question: Perform EDR query for systems with file(s)
   - Take action: Update MISP event with results



Search for Infected Systems



- ✓ Event published by MISP
- ✓ Node-RED performs EDR query
- ✓ EDR Performs Query
- ✓ Node-RED retrieves EDR query results
- ✓ Update MISP event with results

Update MISP Event:
Update MISP event with MAR query results

DXL TIE Reputation APIs

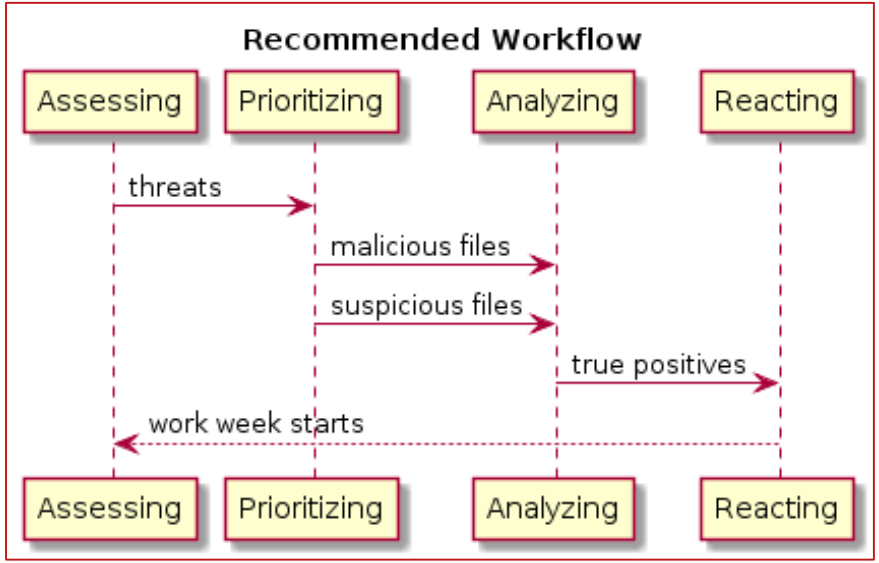
TIE Workflow

Capabilities

Assessing
Dashboards → TIE Server Threat Intelligence
Dashboards → TIE Server Files
Dashboards → TIE Server Certificates
Dashboards → Dashboard Actions → New
Queries and Reports → Reports → TIE Server Summary → Run/Edit

Prioritizing
TIE Reputations → File Search → Custom → Malicious Files
TIE Reputations → File Search → Custom → Add

Analyzing
TIE Reputations → Select Item → File Details & Additional Information
TIE Reputations → Select Item → Actions → Associated File Details
TIE Reputations → Select Item → Actions → File Parents
TIE Reputations → Select Item → Actions → Associated Certificate Details
TIE Reputations → Select Item → Actions → Where File has Run
System Tree → Select Item → Actions → TIE → Show Files used on system



Reacting
TIE Reputations → Select Item → Actions → File Most Likely Malicious or File Most Likely Trusted
System Tree → Select Item → Actions → System Health Indicator → Set Possibly Compromised

Article can be found here.

TIE Workflow

Playbooks

Playbook can be found here.

Threat Hunting using Threat Intelligence Exchange

Summary

This is a sample playbook that shows how to leverage `McAfee Threat Intelligence Exchange` (TIE) towards threat hunting. It details how to support workflows inside `McAfee ePolicy Orchestrator` (ePO) that starts from a file-related threat event or a dashboard, continues gathering contextual information thru pivoting and closes disseminating updated local reputation.

Events

The following alerts start the workflows over this playbook:

1. There are ePO Threat Events coming from TIE reporting block events.

   ▼ Implementation
   Search for `ePO Threat Events` coming from `TIE Module for VSE`, `TI Client for ENS 10.2` or `ATP for ENS 10.5` having `Block` or `Would Block` actions.

2. There are IOCs coming from Threat Intelligence feeds

   ▶ Implementation

3. There is suspicious activity identified in TIE reporting

   ▶ Implementation

Analysis

1. *There is common malware activity.*

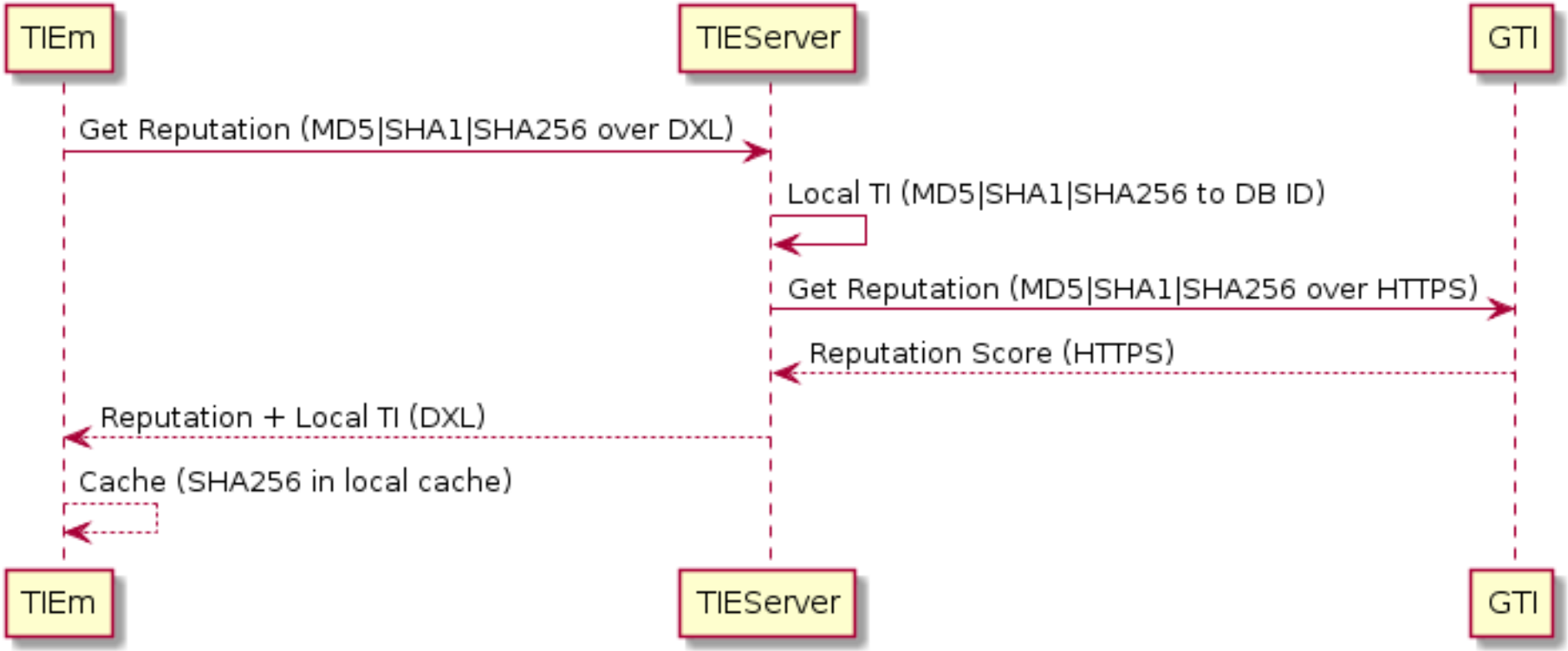
   i. Why is the file reputation suspicious?

      a. Which are the provider-specific reputation scores leading to suspicious?

         ▼ Implementation
         Review `Composite`, `Latest Local`, `Certificate Enterprise`, `Certificate GTI`, `GTI`, `ATD`, `CTD` and `MWG`

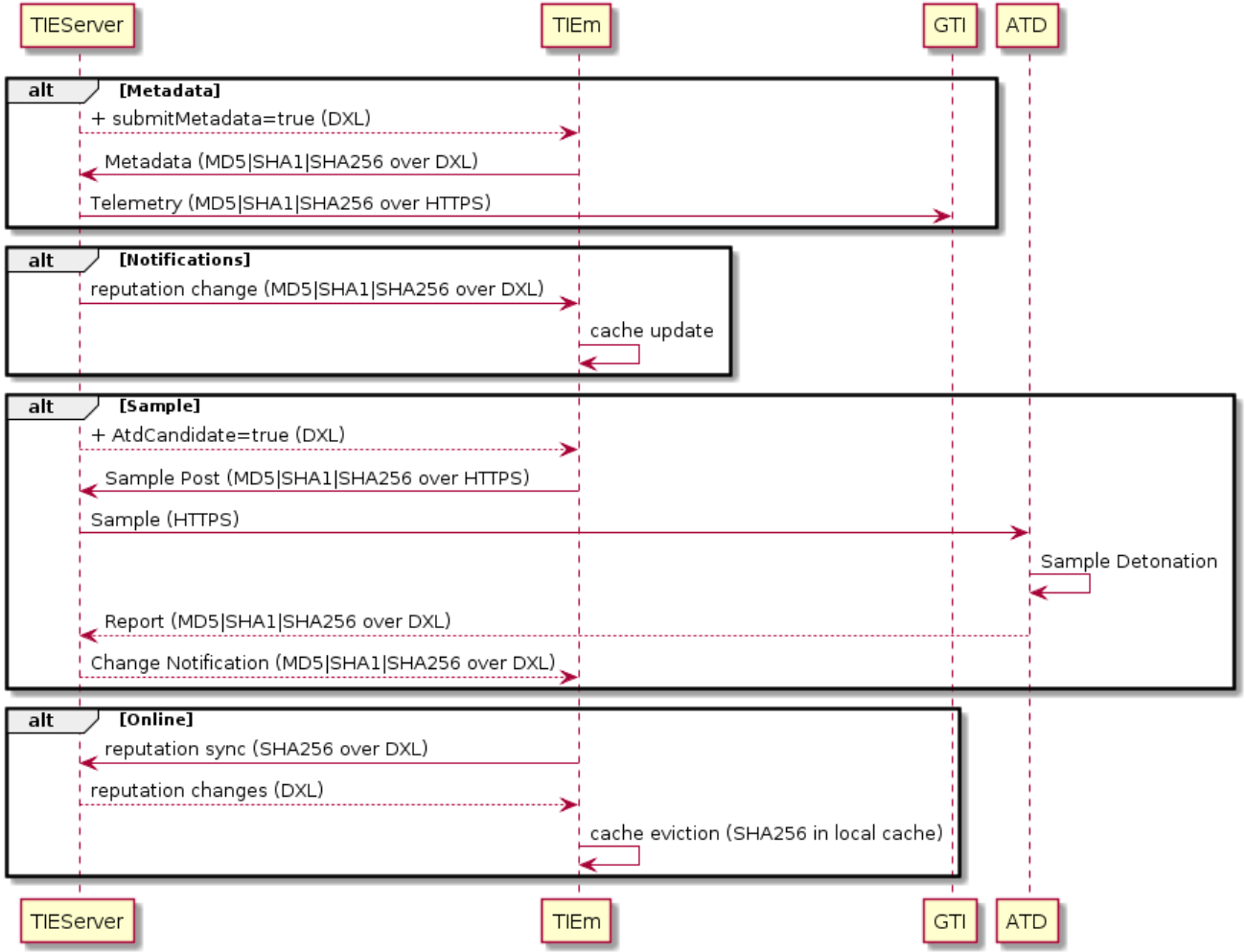
TIE SDK Overview

Basic API Flow



TIE SDK Overview

Extended API Flow



TIE SDK Sample

Client Module

GitHub Repository

Each module APIs are documented with the usual pythonic approach plus examples.

Attributes might be plain Python dictionaries which are then exchanged as JSON thru lower-level DXL APIs.

`set_file_reputation`(*trust_level*, *hashes*, *filename=''*, *comment=''*)

Sets the "Enterprise" reputation (*trust level*) of a specified file (as identified by hashes).

Note: Client Authorization

The OpenDXL Python client invoking this method must have permission to send messages to the `/mcafee/service/tie/file/reputation/set` topic which is part of the `TIE Server Set Enterprise Reputation` authorization group.

The following page provides an example of authorizing a Python client to send messages to an *authorization group*. While the example is based on McAfee Active Response (MAR), the instructions are the same with the exception of swapping the `TIE Server Set Enterprise Reputation` *authorization group* in place of `Active Response Server API`:

https://opendxl.github.io/opendxl-client-python/pydoc/marsendauth.html

Example Usage

```
# Set the Enterprise reputation (trust level) for notepad.exe to Known Trusted
tie_client.set_file_reputation(
    TrustLevel.KNOWN_TRUSTED, {
        HashType.MD5: "f2c7bb8acc97f92e987a2d4087d021b1",
        HashType.SHA1: "7eb0139d2175739b3ccb0d1110067820be6abd29",
        HashType.SHA256: "142e1d688ef0568370c37187fd9f2351d7ddeda574f8bfa9b0fa4ef42db85aa2"
    },
    filename="notepad.exe",
    comment="Reputation set via OpenDXL")
```

Parameters:

- **trust_level** -- The new *trust level* for the file. The list of standard *trust levels* can be found in the `dxltieclient.constants.TrustLevel` constants class.
- **hashes** -- A `dict` (dictionary) of hashes that identify the file to update the reputation for. The `key` in the dictionary is the *hash type* and the `value` is the *hex* representation of the hash value. See the `dxltieclient.constants.HashType` class for the list of *hash type* constants.
- **filename** -- A file name to associate with the file (optional)
- **comment** -- A comment to associate with the file (optional)

TIE SDK Overview

General Considerations

Caching!

Messaging

1. Requests should be limited to avoid saturation of the shared resource that is the DXL fabric.
2. The events provided should only be subscribed to by clients that will make use of the events.
3. Subscription should be kept to a minimum to prevent unnecessary DXL deliveries.

Basic principles with JSON API

1. Do not assume any given order of object attributes in a JSON message (per RFC4627).
2. Integrators must ignore any additional data in a JSON message which they were not expecting.

Reputation Change Events

Broadcast: Intended to be used by listeners which are interested in general updates.

Targeted: For non prevalent files will only direct these reputation change notifications to them if they have previously requested the reputation for the specific file/certificate that is affected. Once the file/certificate becomes prevalent the TIE service will send this event to all registered listeners.

TIE SDK Overview

General Practices

1. Consumer must provide as much metadata as available if requested to do so by the TIE server in the reputation response. Metadata is meaningful context used by the Security Administrator to make an informed decision about the reputation of an object.
2. Consumer must ignore any attributes or values provided in TIE server responses which the consumer is not expecting or capable to address. This is to ensure forwards compatibility of the TIE server API.
3. Consumer must properly handle error responses and timeouts.
4. Consumer must back off retries randomly to avoid congestion.

TIE SDK Overview

Test Scenarios - Caching

1. Consumer must cache reputations of files.
2. Consumer must cache reputations of certificates.
3. Consumer must invalidate cache entries after a configurable amount of days if the entry is not used.
4. Consumer must not refresh reputation proactively and fully rely on notifications.
5. Consumer must ask for certificate reputation when a file is signed.
6. Consumer must check for has-override flag and use file reputation although the file being signed.
7. Consumer should provide MD5, SHA1 and SHA256 file hashes when asking for reputations.
8. Consumer should avoid multiple requests against the same object while the response is still not received before timeout.

TIE SDK Overview

Test Scenarios - Notifications

1. Consumer must subscribe to file reputation change notifications to keep cached items refreshed.
2. Consumer must subscribe to certificate reputation change notifications to keeps cached items refreshed properly.
3. Consumer must refresh cache entries only already present in cache (note that after 5k endpoints the reputation change topic behaves as a broadcast).
4. Consumer must reuse data in the notification instead of querying again for reputations.
5. Consumer should promptly react after reputation change updates (e.g. a running matching binary is killed).

TIE SDK Overview

Test Scenarios – Cache Invalidation Updates

1. Consumer cache must be indexed for fast access, but it must nor rely on presence of a given hash type.
2. Consumer must ask for reputation updates after becoming online to invalidates cached entries.
3. Consumer must not refresh updated reputations after becoming online until they are needed.
4. Consumer should limit each reputation updates request. Note that it will default to 100 if omitted.
5. Consumer should preserve the serverTime of the response to the last reputation request issued and use it as sinceTime in the first reputation updates request after becoming online.
6. Consumer should ask rolling requests following last time-stamp in response until getting all available updates.

TIE SDK Overview

Test Scenarios – Sample Submission

1. Submitter must honor the ATD sandbox submission candidate flag in reputation response.
2. Submitter must use the provided TIE Master IP address.
3. Submitter must not send samples over a configurable file size.
4. Submitter must submit samples using HTTPS.
5. Submitter should retry sample submission in case of failure.
6. Submitter should validate certificate before submission using intermediate CA located in ePO.

TIE Traces

Messages

```
 INFO {2019-07-09 22:13:46,111} - traceId: {78ec1a66-1058-442f-89ef-219b13bdfc3c} -
request:/mcafee/service/tie/file/reputation : {"agentGuid":"{363e1c68-9dd3-11e9-
03a7005056b98114}","hashes":[{"type":"sha1","value":"EOaRNvkYMMiOTDLEjuoAX3Tv00U="},{"type":"sha256","v
alue":"ekY50D2+huvbvFzFTDyN3zAtTUdspwXhZLiE7Pa6bY0="},{"type":"md5","value":"1/5jxxHWpyUvRAQXKt9NCw=="}
]}
 INFO {2019-07-09 22:13:46,663} - traceId: {78ec1a66-1058-442f-89ef-219b13bdfc3c} -
response:/mcafee/service/tie/file/reputation:{"reputations":[{"providerId":1,"trustLevel":99,"createDat
e":1562710426,"attributes":{"2120340":"2134900864"}},{"providerId":3,"trustLevel":0,"createDate":156271
0426,"attributes":{"2101652":"0","2123156":"0","2098277":"0","2102165":"1562710426","2114965":"0","2111
893":"820","2139285":"144959617300955252"}}],"props":{"atdCandidate":1,"submitMetaData":1,"serverTim
e":1562710426,"masterServer":"HOSTNAME;IP"}}

-bash-4.1# head tieserver-traffic.csv
2019-06-29T00:32:00,269 2       /mcafee/service/tie/reputation/updates  {e3b4bb6c-784b-11e8-24af-
000000000000}  {4edceff8-a753-4543-9eb4-3ef9cec39fa7}:V1.38c.5d16b180.1
REQUEST qp=86735 tq=1
2019-06-29T00:32:27,029 2       /mcafee/service/tie/cert/reputation     {9a60f94c-7893-11e8-0b6d-
000000000000}  {9796b868-862e-4d59-bbbc-3f7dacfb174d}:V1.16c8.5d16b177.1
SVhHqTGHz7jHH4QMt7QUl62Vxk8=                                                    REQUEST
```

Format is TIMESTAMP, DELAY, API, DEVICE_ID, REQUEST_ID, PARAMETERS

TIE Traces

Logging Rotation

```
-bash-4.1# ll -h
total 391M
-rw-r--r-- 1 mfetie mfetie  15M Jul  9 21:06 tieserver-traffic.csv
-rw-r--r-- 1 mfetie mfetie  36M Jun 29 00:32 tieserver-traffic.csv-1.gz
-rw-r--r-- 1 mfetie mfetie  37M Mar 21 19:31 tieserver-traffic.csv-2.gz
-rwx------ 1 mfetie mfetie  36M Dec 14  2018 tieserver-traffic.csv-3.gz
-rwx------ 1 mfetie mfetie  38M Sep 28  2018 tieserver-traffic.csv-4.gz
-rwx------ 1 mfetie mfetie  39M Jul 13  2018 tieserver-traffic.csv-5.gz
-rwx------ 1 mfetie mfetie 192M Feb 27  2018 tieserver-traffic.csv-6.gz
-bash-4.1# zcat tieserver-traffic.csv-1.gz | wc -l
683631
```

OpenDXL - Hands-on Exercises

Getting started with the OpenDXL python client

Install the client package

```
# python3 -m pip install dxlclient
```

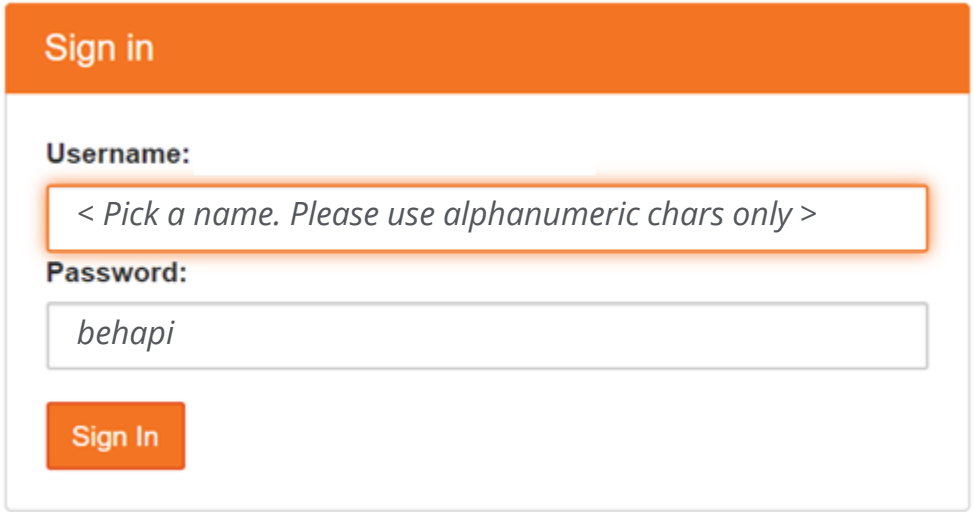
Provision the configuration files

```
$ dxlclient provisionconfig config_dir <server ip> client_name
```

https://github.com/opendxl/opendxl-client-python

Hands-on exercises

https://mcafee.behapi.net



“Make your own AV in 10 minutes” using OpenDXL and VirusTotal