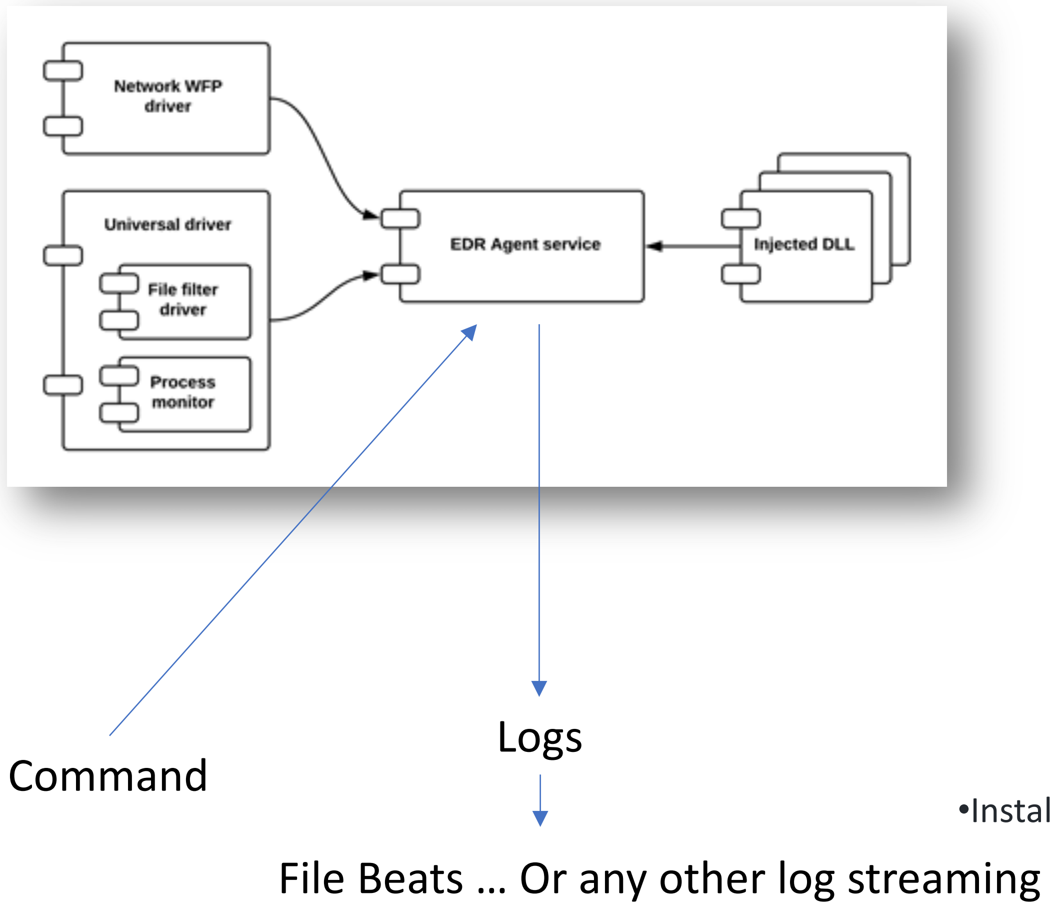
Endpoint Detection and Response (EDR) Ontology Considerations - Track 2

Update 3/23/2022

Public Domain Dedication

OSS EDR Candidate 1: Comodo



Command

Logs

File Beats … Or any other log streaming

The Open EDR consists of the following components:

- Runtime components
  - Core Library – the basic framework;
  - Service – service application;
  - Process Monitor – components for per-process monitoring;
    - Injected DLL – the library which is injected into different processes and hooks API calls;
    - Loader for Injected DLL – the driver component which loads injected DLL into each new process
    - Controller for Injected DLL – service component for interaction with Injected DLL;
  - System Monitor – the genetic container for different kernel-mode components;
  - File-system mini-filter – the kernel component that hooks I/O requests file system;
  - Low-level process monitoring component – monitors processes creation/deletion using system callbacks
  - Low-level registry monitoring component – monitors registry access using system callbacks
  - Self-protection provider – prevents EDR components and configuration from unauthorized changes
  - Network monitor – network filter for monitoring the network activity;
- Installer

OSS EDR Candidate 1: Comodo

… The API is provided by the autogenerated documentation.
The API of components and implementation details (including code samples) are described in the source code (as comments). The automatic documentation generator uses these sources for generation documents. These documents can be found in appropriate API documents.

…

Generic high-level interaction diagram for runtime components

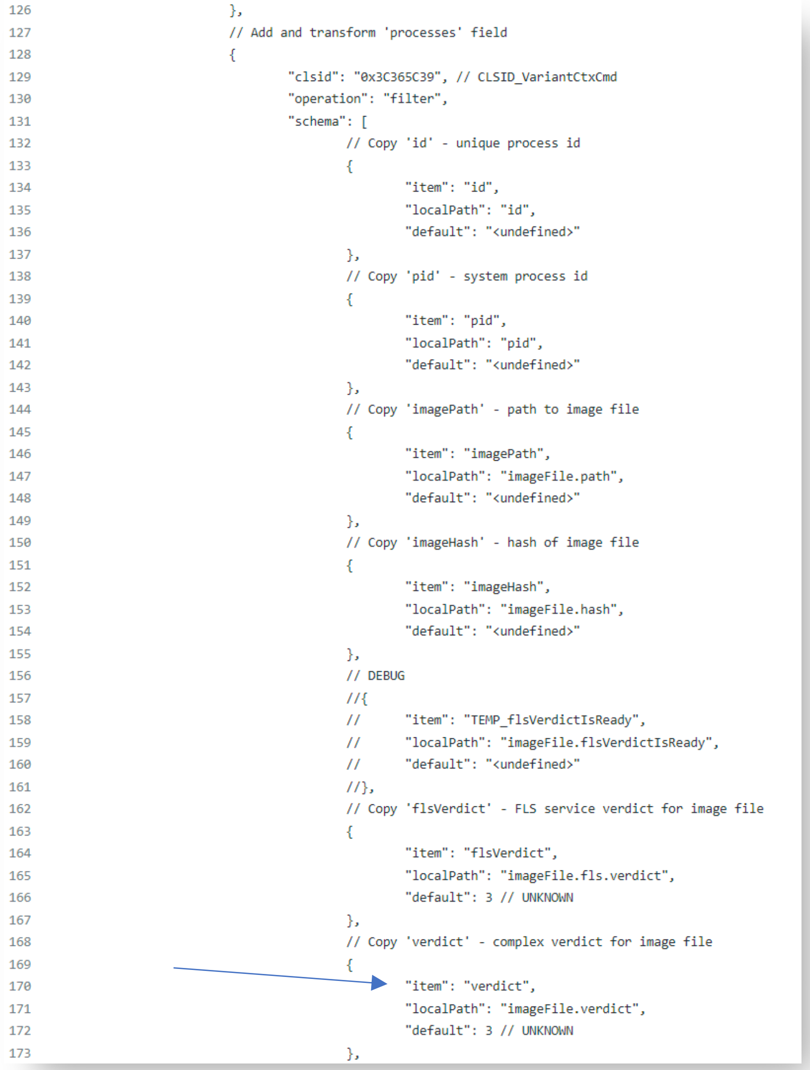
 * The service initializes and uses other components for collecting data, providing the response and for other functions.

https://github.com/ComodoSecurity/openedr

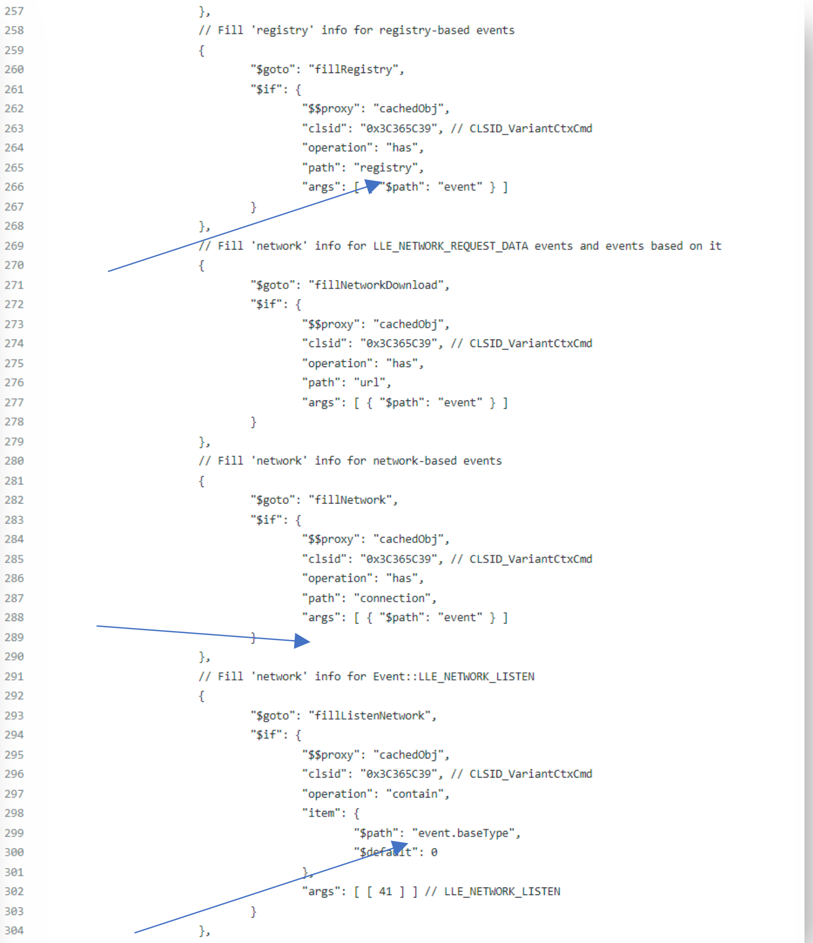
So …

https://github.com/ComodoSecurity/openedr/blob/main/edrav2/iprj/edrdata/qsc/transform_for_cloud.qsc

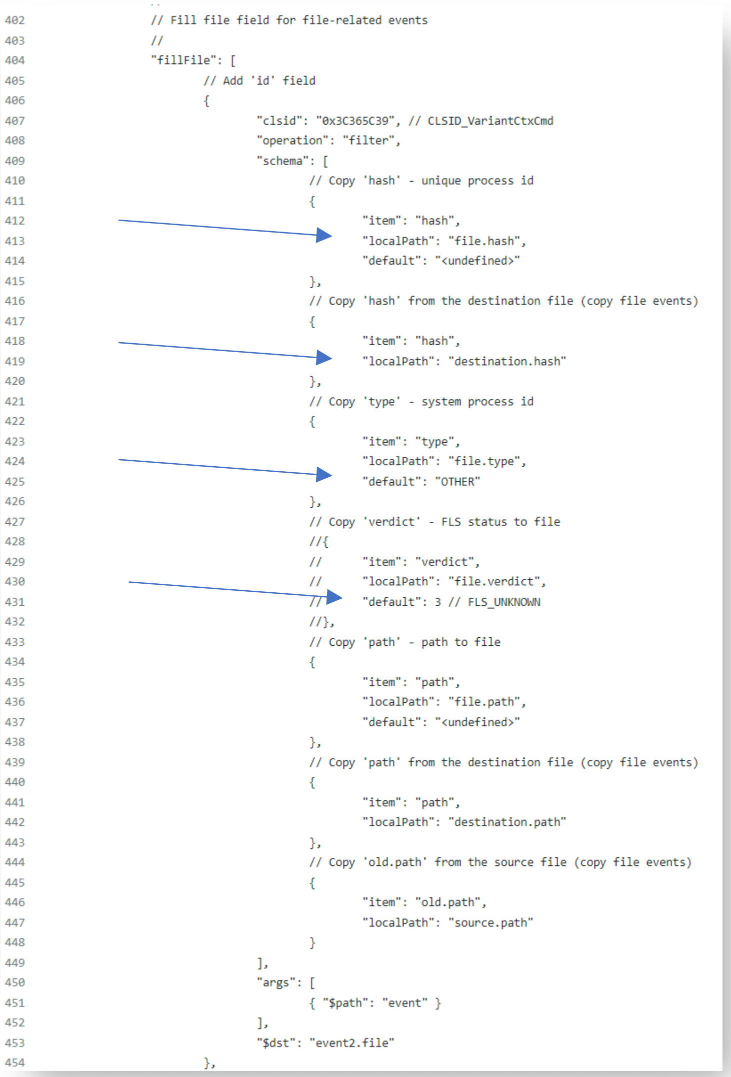
COSS EDR Candidate 1: Comodo idiosyncrasies (prelim)



```
},
// Add and transform 'processes' field
{
    "clsid": "0x3C365C39", // CLSID_VariantCtxCmd
    "operation": "filter",
    "schema": [
        // Copy 'id' - unique process id
        {
            "item": "id",
            "localPath": "id",
            "default": "<undefined>"
        },
        // Copy 'pid' - system process id
        {
            "item": "pid",
            "localPath": "pid",
            "default": "<undefined>"
        },
        // Copy 'imagePath' - path to image file
        {
            "item": "imagePath",
            "localPath": "imageFile.path",
            "default": "<undefined>"
        },
        // Copy 'imageHash' - hash of image file
        {
            "item": "imageHash",
            "localPath": "imageFile.hash",
            "default": "<undefined>"
        },
        // DEBUG
        //{
        //      "item": "TEMP_flsVerdictIsReady",
        //      "localPath": "imageFile.flsVerdictIsReady",
        //      "default": "<undefined>"
        //},
        // Copy 'flsVerdict' - FLS service verdict for image file
        {
            "item": "flsVerdict",
            "localPath": "imageFile.fls.verdict",
            "default": 3 // UNKNOWN
        },
        // Copy 'verdict' - complex verdict for image file
        {
            "item": "verdict",
            "localPath": "imageFile.verdict",
            "default": 3 // UNKNOWN
        },
```



```
},
// Fill 'registry' info for registry-based events
{
    "$goto": "fillRegistry",
    "$if": {
        "$$proxy": "cachedObj",
        "clsid": "0x3C365C39", // CLSID_VariantCtxCmd
        "operation": "has",
        "path": "registry",
        "args": [ { "$path": "event" } ]
    }
},
// Fill 'network' info for LLE_NETWORK_REQUEST_DATA events and events based on it
{
    "$goto": "fillNetworkDownload",
    "$if": {
        "$$proxy": "cachedObj",
        "clsid": "0x3C365C39", // CLSID_VariantCtxCmd
        "operation": "has",
        "path": "url",
        "args": [ { "$path": "event" } ]
    }
},
// Fill 'network' info for network-based events
{
    "$goto": "fillNetwork",
    "$if": {
        "$$proxy": "cachedObj",
        "clsid": "0x3C365C39", // CLSID_VariantCtxCmd
        "operation": "has",
        "path": "connection",
        "args": [ { "$path": "event" } ]
    }
},
// Fill 'network' info for Event::LLE_NETWORK_LISTEN
{
    "$goto": "fillListenNetwork",
    "$if": {
        "$$proxy": "cachedObj",
        "clsid": "0x3C365C39", // CLSID_VariantCtxCmd
        "operation": "contain",
        "item": {
            "$path": "event.baseType",
            "$default": 0
        },
        "args": [ [ 41 ] ] // LLE_NETWORK_LISTEN
    }
},
```



```
// Fill file field for file-related events
//
"fillFile": [
    // Add 'id' field
    {
        "clsid": "0x3C365C39", // CLSID_VariantCtxCmd
        "operation": "filter",
        "schema": [
            // Copy 'hash' - unique process id
            {
                "item": "hash",
                "localPath": "file.hash",
                "default": "<undefined>"
            },
            // Copy 'hash' from the destination file (copy file events)
            {
                "item": "hash",
                "localPath": "destination.hash"
            },
            // Copy 'type' - system process id
            {
                "item": "type",
                "localPath": "file.type",
                "default": "OTHER"
            },
            // Copy 'verdict' - FLS status to file
            //{
            //      "item": "verdict",
            //      "localPath": "file.verdict",
            //      "default": 3 // FLS_UNKNOWN
            //},
            // Copy 'path' - path to file
            {
                "item": "path",
                "localPath": "file.path",
                "default": "<undefined>"
            },
            // Copy 'path' from the destination file (copy file events)
            {
                "item": "path",
                "localPath": "destination.path"
            },
            // Copy 'old.path' from the source file (copy file events)
            {
                "item": "old.path",
                "localPath": "source.path"
            }
        ],
        "args": [
            { "$path": "event" }
        ],
        "$dst": "event2.file"
    },
```

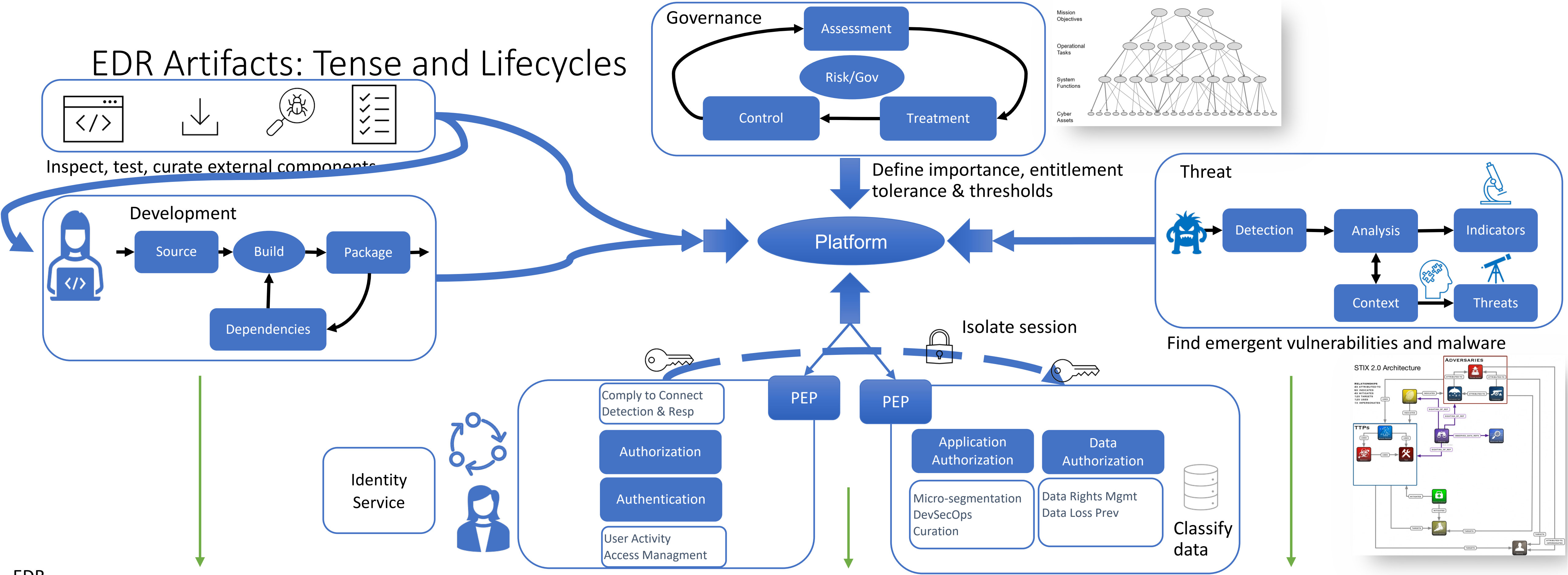
Per Last Meeting

- Parking interactions with external groups until we are ready to form a project/sub-project.
- Proceeding with analysis for normalization – cataloging the “gotchas” across *DR tooling (tooling capable of supporting *DR (EDR, NDR, XDR, …) required functionality (emergent threat detection & response, intelligence ingestion, mal/anomalous detection, hunting, analysis, response…) :
  - Comodo – widely used
  - GRR - cloud scale
  - BlueSpawn – academic
  - Question 1: Others?
  - Question 2: How to include non-OSS products (proprietary integrations (commercial API), normalized mappings (ATT&CK), or only at the “indicator sharing” level (STIX)…)

Track 2 - Objective 1 - Status

- Indication and Behavior Normalization
- Indications and Behaviors are potentially invariant attributes across *DR tooling, so are important in normalizing across different *DR implementations.
- As Indications and Behaviors get exposed, across the security, dev and system disciplines, some implicit context around indications and behavior, need to be made more explicit.

EDR Artifacts: Tense and Lifecycles



Inspect, test, curate external components

Development

Source

Build

Package

Dependencies

Governance

Assessment

Risk/Gov

Control

Treatment

Define importance, entitlement tolerance & thresholds

Platform

Threat

Detection

Analysis

Indicators

Context

Threats

Find emergent vulnerabilities and malware

Isolate session

PEP

PEP

Comply to Connect Detection & Resp

Authorization

Authentication

User Activity Access Managment

Identity Service

Application Authorization

Data Authorization

Micro-segmentation DevSecOps Curation

Data Rights Mgmt Data Loss Prev

Classify data

EDR Artifacts

Dev Context & Telemetry

- IDs: SW Component ID: SWID, CycloneDX, SPDX, GUID, “string”…
- **Intended** SW Component Configuration: Settings, Privilege, Dependencies, secrets, obj hashes, policy …
- **Expected** SW Component Configuration (Test): DLLs Used,
- **Intended** SW Component Behavior: OpenAPI, RAML, … (L7)
- **Expected** SW Component Behavior (Test): SysCall/ Res Profile, Memory, Network, Data, CPU

Operational Telemetry & Context

- IDs: Instance MAC, IP, SysSID, GUIDs, … (stack)
- Provisioning Decisions (operationalization)
- Provisioned Configuration (deployment baseline)
- Mapping from SW Component Manifest -> Instance:
- **Observed** SW Component Configuration (Instances):
- **Observed** SW Component Behavior (Instances):
- **Observed** Telemetry Inconsistency:

Curated Intelligence

- IDs: Relevant Component ID Types specified, Malicious IDs identified, …
  - “strings”, SWID, SPDX, CycloneDX, (SCAP: CPE, CVE, …)
- **Relevant** Indicators (Instances):
- **Relevant** Behaviors (Instances):
- **Relevant** Inconsistencies (anomaly): (telemetry)
- Mitigation **Verification**: State & behavior restriction
- Remediation **Verification**: Sustained resilience to repeat exploit
- Cleanup considerations

Ontology Consideration 1: Tense

- “Tense” is conventionally implicit in EDR ecosystem, and inconsistently selected & represented across EDR tools/svcs.
- EDR relevant artifacts are produced by different processes, at different times with different implications
    - Intended – By design or decision. May be coverage tested. Ex. Supported API
    - Expected – Observed under test. Cannot be coverage tested. Good automate-able baseline. Test Platform sensitive. Ex. SysCall pattern demonstrated during coverage testing.
    - Observed – Runtime or forensic telemetry. Relevant by difference from Intended, Expected or by association with Vuln* or Mal* via Intelligence feed.
    - Verification – What should I see post mitigation/remediation?
- Maybe be the same attribute across all tenses: Intended, Expected, Relevant, Verification – Registry setting, DLL Hash
- But, may be different. May only be indirectly associated: Patch level vs. Patch Level Indication
- Recommendation: make provision to capture Intended, Expected, Observed, Verification, … on Configuration and Behavior
- * Process , Ontologies (upper vs lower) - Patrick