RawSec

WHIDS an Open-Source EDR for Windows

Github / Twitter: 0xrawsec

Project: https://github.com/0xrawsec/whids

About Me

First Name: Quentin **Last Name**: JEROME **Age**: 32

Freelance Security Consultant working in Luxembourg, running for my own company

- Originally doing Incident Response, digital forensics, malware oriented digital forensics, endpoint's based Threat Hunting …
- Open-Source developer (in my free time) mainly Go, C, Python. At the origin of several projects: Gene, WHIDS, golang-evtx, golang-misp, golang-etw …

Why do I do that ?: for pure fun, to bring Open Source alternative, to help people, ~~to make money~~

Motivations

Problems:

- SIEM are good but they can analyze only a limited number of events. So defenders have to make a smart signal/noise ratio while minimizing blind spots.
- In traditional IR approaches there are sometimes day(s) between detection and artifact collection which leaves room for changes
- No EDR on the Open-Source market (**4 years ago** when I started this project)

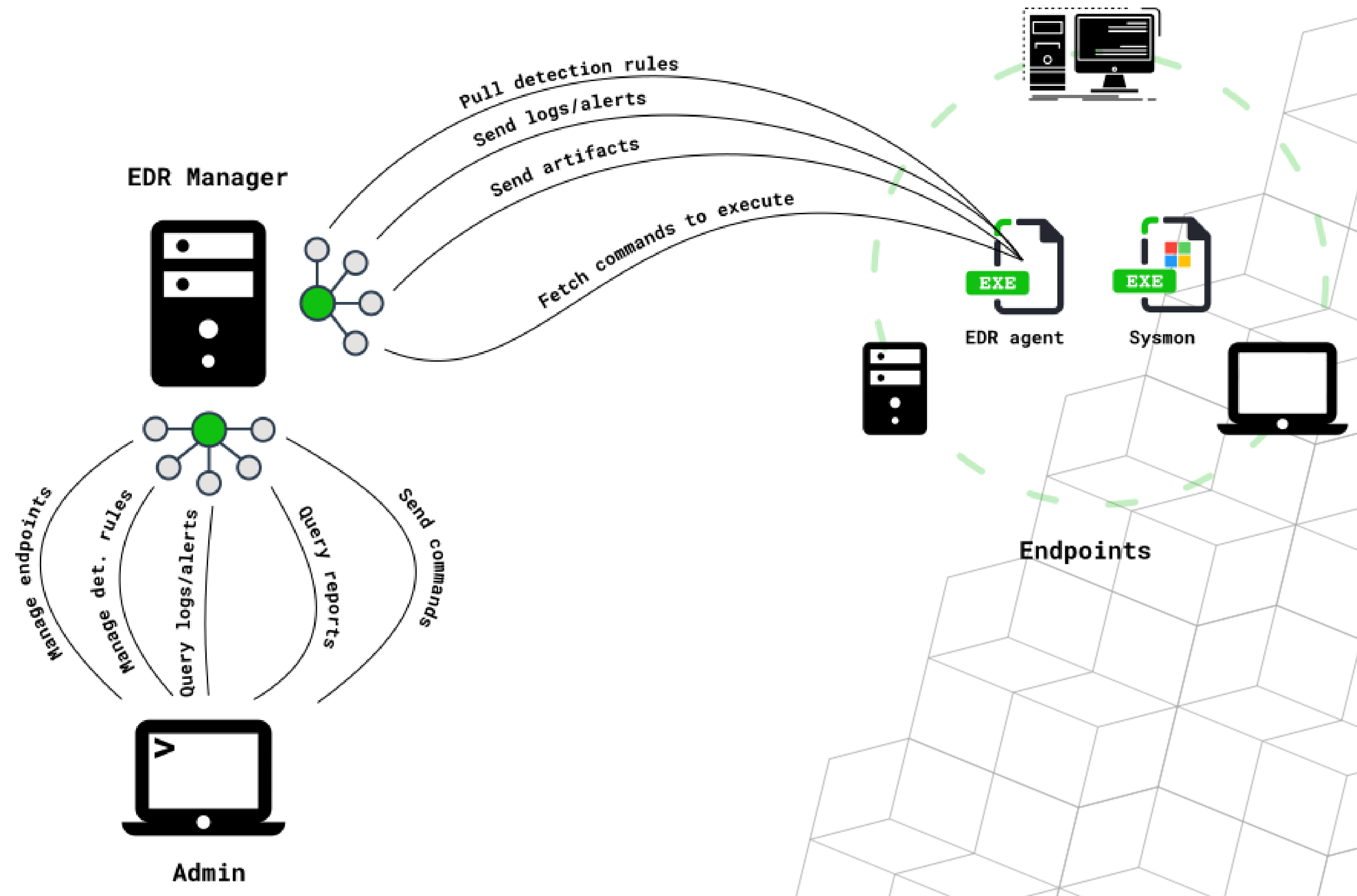
Vision:

- Provide a robust tool for SMBs and people who cannot afford buying an expansive solution
- Making the tool highly customizable for **more control** (understand what got detected and why)
- Bring a new dimension to Incident Response by collecting artifacts in near RT
- Make the whole solution pluggable with any other Open-Source tool

Architecture

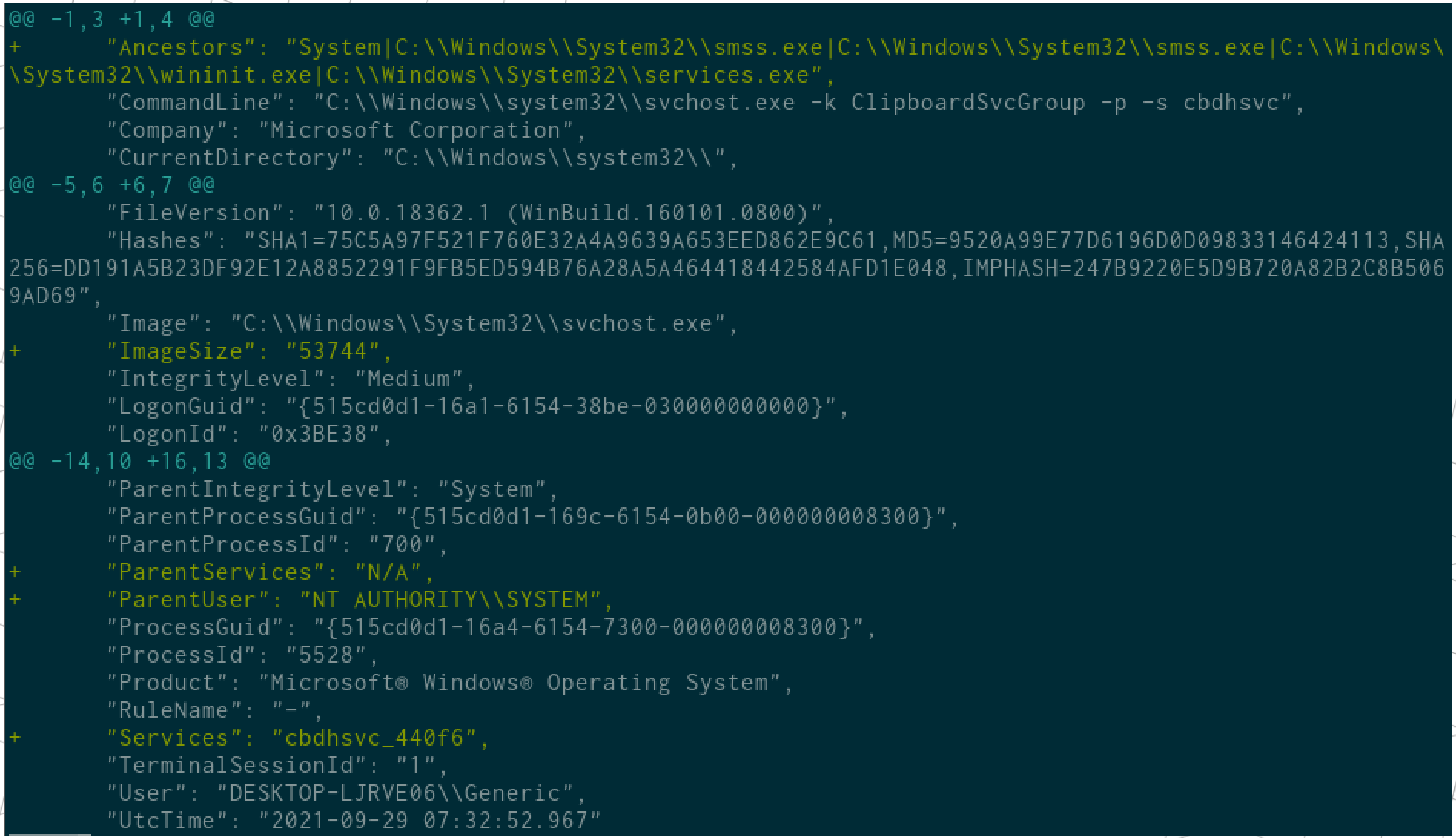
Agent
- **Correlate** events (**ETW**) on host
- **Detect** in real time suspicious events (raw/correlated) based on user defined rules
- **React** to detection in RT: dump artifacts (files, process, registries), blacklist process, kill process

Manager
- **Central** manager to administrate endpoints
- **Collect** logs, and artifacts
- **HTTP API** for administrators and plugins



RawSec

Correlate & enrich to better detect



```
@@ -1,3 +1,4 @@
+       "Ancestors": "System|C:\\Windows\\System32\\smss.exe|C:\\Windows\\System32\\smss.exe|C:\\Windows\
\System32\\wininit.exe|C:\\Windows\\System32\\services.exe",
        "CommandLine": "C:\\Windows\\system32\\svchost.exe -k ClipboardSvcGroup -p -s cbdhsvc",
        "Company": "Microsoft Corporation",
        "CurrentDirectory": "C:\\Windows\\system32\\",
@@ -5,6 +6,7 @@
        "FileVersion": "10.0.18362.1 (WinBuild.160101.0800)",
        "Hashes": "SHA1=75C5A97F521F760E32A4A9639A653EED862E9C61,MD5=9520A99E77D6196D0D09833146424113,SHA
256=DD191A5B23DF92E12A8852291F9FB5ED594B76A28A5A464418442584AFD1E048,IMPHASH=247B9220E5D9B720A82B2C8B506
9AD69",
        "Image": "C:\\Windows\\System32\\svchost.exe",
+       "ImageSize": "53744",
        "IntegrityLevel": "Medium",
        "LogonGuid": "{515cd0d1-16a1-6154-38be-030000000000}",
        "LogonId": "0x3BE38",
@@ -14,10 +16,13 @@
        "ParentIntegrityLevel": "System",
        "ParentProcessGuid": "{515cd0d1-169c-6154-0b00-000000008300}",
        "ParentProcessId": "700",
+       "ParentServices": "N/A",
+       "ParentUser": "NT AUTHORITY\\SYSTEM",
        "ProcessGuid": "{515cd0d1-16a4-6154-7300-000000008300}",
        "ProcessId": "5528",
        "Product": "Microsoft® Windows® Operating System",
        "RuleName": "-",
+       "Services": "cbdhsvc_440f6",
        "TerminalSessionId": "1",
        "User": "DESKTOP-LJRVE06\\Generic",
        "UtcTime": "2021-09-29 07:32:52.967"
```

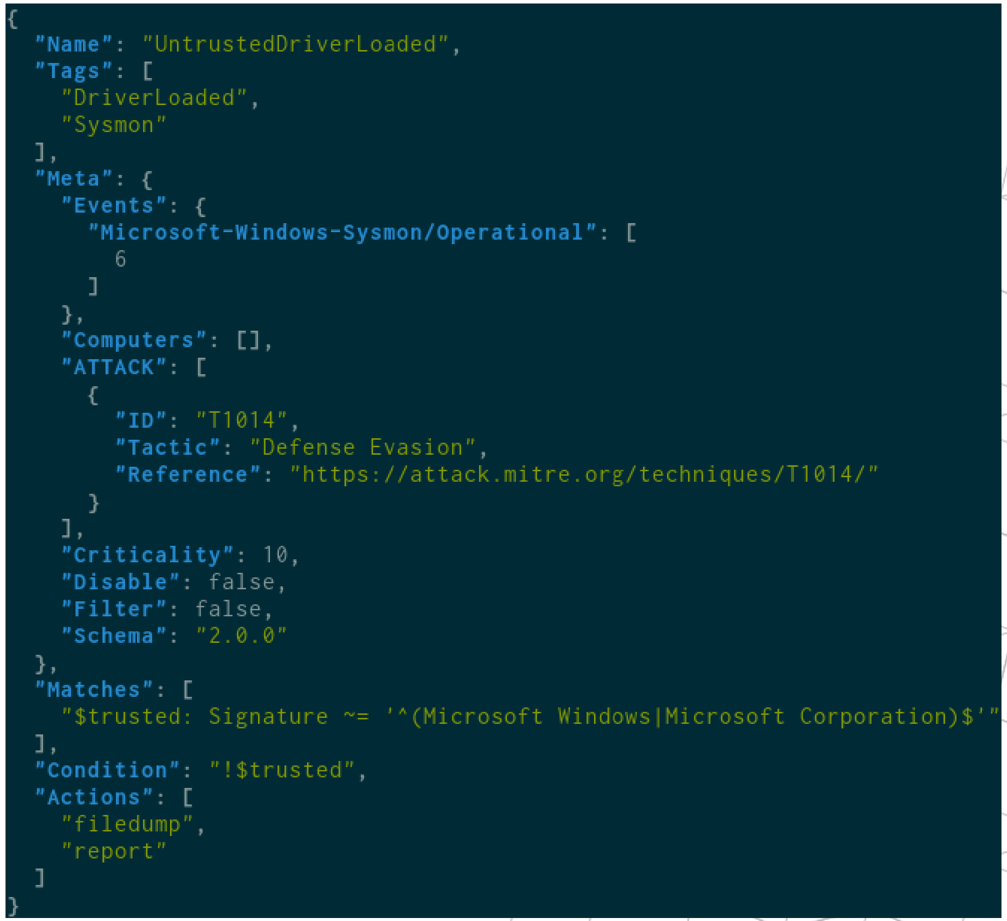
Detect & react

Detect: **Condition** section of the rule (combining **Matches** into a logical expression)

React: **Actions** section specifying the actions the **EDR** must take on detection

NB: Rules can also be used to **filter-in** logs

Rule format: https://rawsec.lu/doc/gene/2.0



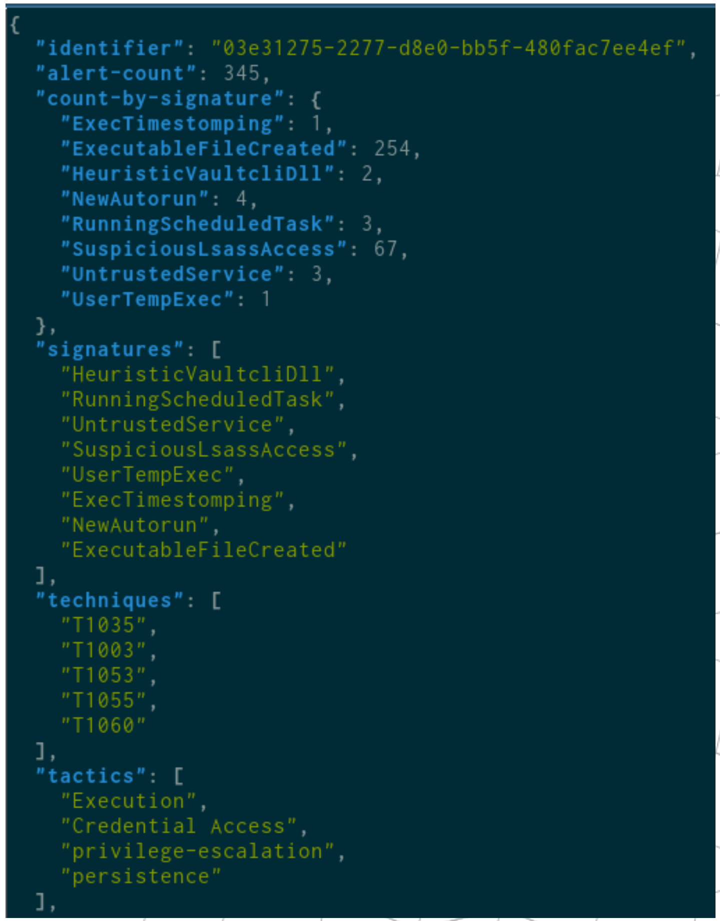
```
{
  "Name": "UntrustedDriverLoaded",
  "Tags": [
    "DriverLoaded",
    "Sysmon"
  ],
  "Meta": {
    "Events": {
      "Microsoft-Windows-Sysmon/Operational": [
        6
      ]
    },
    "Computers": [],
    "ATTACK": [
      {
        "ID": "T1014",
        "Tactic": "Defense Evasion",
        "Reference": "https://attack.mitre.org/techniques/T1014/"
      }
    ],
    "Criticality": 10,
    "Disable": false,
    "Filter": false,
    "Schema": "2.0.0"
  },
  "Matches": [
    "$trusted: Signature ~= '^(Microsoft Windows|Microsoft Corporation)$'"
  ],
  "Condition": "!$trusted",
  "Actions": [
    "filedump",
    "report"
  ]
}
```

ATT&CK Integration

1. **ATT&CK patterns** are defined at rule level
2. EDR manager provides **detection reports** showing ATT&CK info per endpoint within a time range
3. Reports are **rankable** -> can be used to **prioritize** investigations

Once investigated a **detection report** is **archived** and can be searched back

Detection reports offers a view on a per endpoint basis ≠ per alert view (traditional approach)



```
{
  "identifier": "03e31275-2277-d8e0-bb5f-480fac7ee4ef",
  "alert-count": 345,
  "count-by-signature": {
    "ExecTimestomping": 1,
    "ExecutableFileCreated": 254,
    "HeuristicVaultcliDll": 2,
    "NewAutorun": 4,
    "RunningScheduledTask": 3,
    "SuspiciousLsassAccess": 67,
    "UntrustedService": 3,
    "UserTempExec": 1
  },
  "signatures": [
    "HeuristicVaultcliDll",
    "RunningScheduledTask",
    "UntrustedService",
    "SuspiciousLsassAccess",
    "UserTempExec",
    "ExecTimestomping",
    "NewAutorun",
    "ExecutableFileCreated"
  ],
  "techniques": [
    "T1035",
    "T1003",
    "T1053",
    "T1055",
    "T1060"
  ],
  "tactics": [
    "Execution",
    "Credential Access",
    "privilege-escalation",
    "persistence"
  ],
```

Faster response thanks to IR reports

Bring contextual information

- Solve **90% of incidents** without further data acquisition. Incident Handlers can focus on the data **rather than focusing on how to get the data**
- Towards **automation driven IR**. Reports are in a standard format and contains loads of information (baseline reports -> find uncommon patterns).

Two ways to generate reports

1. **Automatic**: detections can trigger **reporting** actions (on top of already existing **artifacts dumping** actions)
2. **On-demand**: Commands can be executed on endpoints **from the manager**

What a **report** contains ?

- **processes** running, **drivers/modules** loaded, **network connections & DNS resolutions**, **last files** opened … -> **instant** to generate (all in memory)
- can include the output of **any tool** (osquery, autoruns …) you like

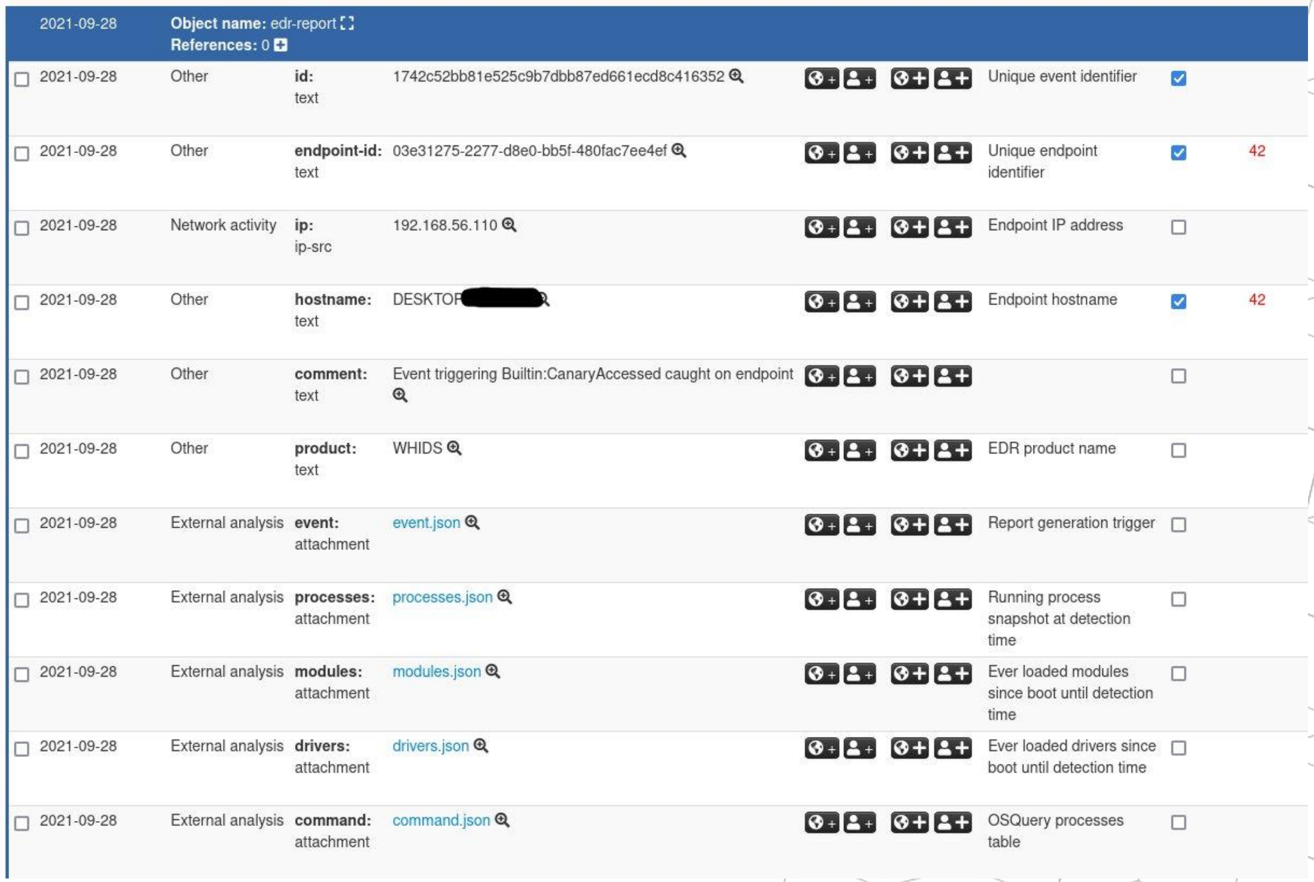
Pluggability

Work in progress

- **PyWHIDS**: library to use manager’s API from Python

Plugin examples

- **reporting.py** plugin to push to MISP IR reports generated by WHIDS
- **sightings.py** plugin to add MISP sightings as events are received by the EDR



2021-09-28 **Object name:** edr-report
References: 0

☐	2021-09-28	Other	**id:** text	1742c52bb81e525c9b7dbb87ed661ecd8c416352	Unique event identifier	✓	
☐	2021-09-28	Other	**endpoint-id:** text	03e31275-2277-d8e0-bb5f-480fac7ee4ef	Unique endpoint identifier	✓	42
☐	2021-09-28	Network activity	**ip:** ip-src	192.168.56.110	Endpoint IP address	☐	
☐	2021-09-28	Other	**hostname:** text	DESKTOP[redacted]	Endpoint hostname	✓	42
☐	2021-09-28	Other	**comment:** text	Event triggering Builtin:CanaryAccessed caught on endpoint		☐	
☐	2021-09-28	Other	**product:** text	WHIDS	EDR product name	☐	
☐	2021-09-28	External analysis	**event:** attachment	event.json	Report generation trigger	☐	
☐	2021-09-28	External analysis	**processes:** attachment	processes.json	Running process snapshot at detection time	☐	
☐	2021-09-28	External analysis	**modules:** attachment	modules.json	Ever loaded modules since boot until detection time	☐	
☐	2021-09-28	External analysis	**drivers:** attachment	drivers.json	Ever loaded drivers since boot until detection time	☐	
☐	2021-09-28	External analysis	**command:** attachment	command.json	OSQuery processes table	☐	

Latest News

- **PyWHIDS**: python library to interface with WHIDS (work in progress) -> used by **sightings.py** and **reporting.py**
- Uses **ETW logs** as event source -> more logs, less resources and higher throughput
- Improved admin API on manager’s side
- API enabling event streaming through Websocket
  - Pretty cool feature to implement any plugin needing to receive logs in real time
- New commands supported by agent (hash file, un/contain host, osquery, etc.)
- Completely new way of indexing logs on manager making event retrieval very fast
- Use of an ORM like framework (homemade ☺) for manager’s data persistence

RawSec

Future Work

1. Make a new release
   - re-work some old API endpoints for better integration with ORM framework
   - decouple MISP from WHIDS for IoC management -> go for a Python plugin approach
   - make everything manageable through HTTP API
2. Improve PyWHIDS library and plugins
3. Build new detection rules
4. Explore portability to Linux thanks to Sysmon for Linux ☺
5. May be a GUI one day !

Thank you all !

Contact via Twitter/Github **@0xrawsec**

- Feel free to open issues, ask questions, give feedbacks/suggestions …

References:

- WHIDS: https://github.com/0xrawsec/whids
- PyWHIDS: https://github.com/0xrawsec/pywhids
- Golang-etw: https://github.com/0xrawsec/golang-etw
- Gene: https://github.com/0xrawsec/gene
- Gene rules: https://github.com/0xrawsec/gene-rules
- Gene Documentation: https://rawsec.lu/doc/gene/2.0