RawSec

WHIDS integration with MISP

Github / Twitter: 0xrawsec

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

RawSec

?I ma ohw

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

- Originally doing Incident Response, digital forensics, malware oriented digital forensics …
- Also Open-Source developer (in my free time) mainly Go, C, Python. At the origin of several projects:
  - Golang-evtx
  - Golang-misp
  - Gene
  - WHIDS

Doing other stuffs as well: software RE, bug hunting ...

What the hell is WHIDS?

Stands for: Windows Host IDS (even though it is more than just an IDS)

To be more accurate, it **combines** IDS features with detection based Incident Response Capabilities.

WHIDS strongly relies on the existence of **Microsoft Sysmon** since most of its nice features are built on to of Sysmon events

Features:

- **Correlate** Windows Event on host
- **Detect** in real time suspicious events (raw/correlated) based on user defined rules
- **React** to the detection:
  - Dump files
  - Dump process
  - Dump registry
- Can send all the information collected to a central point

RawSec

Why the hell? Then!

I want people who cannot afford expansive solutions (EDR, SIEM …) to have something:

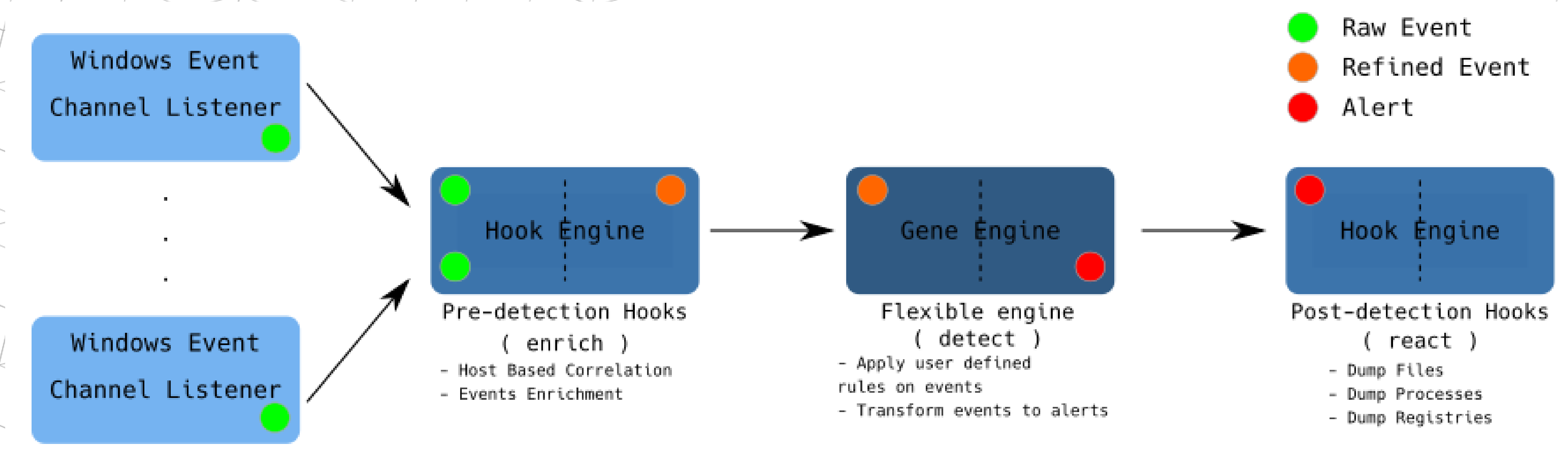
- They can craft detection rules specific to their environment

  Spoiler Alert: vendors often sell generic products, in the end not customizable as you would like it to be. May be it can be customized … but you will have to pay ☺

- That scales
- Which can also be plugged in with the other open source tools they are using

I also want to save time to analysts and allow them to have the data collected in real time

How WHIDS Engine Works



NB: you can listen on absolutely any Windows Event Log channel you want and create detection rules for those

Just an example of enrichment

Original Event

```
CommandLine: C:\\Windows\\system32\\svchost.exe -k appmodel -p -s
camsvc
Company: Microsoft Corporation
CurrentDirectory: C:\\Windows\\system32\\Description: Host Process
for Windows Services
FileVersion: 10.0.18362.1 (WinBuild.160101.0800)
Hashes:
SHA1=75C5A97F521F760E32A4A9639A653EED862E9C61,MD5=9520A99E77D6196D
0D09833146424113,SHA256=DD191A5B23DF92E12A8852291F9FB5ED594B76A28A
5A464418442584AFD1E048,IMPHASH=247B9220E5D9B720A82B2C8B5069AD69
Image: C:\\Windows\\System32\\svchost.exe
IntegrityLevel: System
LogonGuid: {515cd0d1-df83-5d00-0000-0020e7030000}
LogonId: 0x3e7
OriginalFileName: svchost.exe
ParentCommandLine: C:\\Windows\\system32\\services.exe
ParentImage: C:\\Windows\\System32\\services.exe
ParentProcessGuid: {515cd0d1-df83-5d00-0000-0010d6620000}
ParentProcessId: 608ProcessGuid: {515cd0d1-33b8-5d01-0000-
001024046a00}
ProcessId: 10244
Product: Microsoft┬« Windows┬« Operating System
RuleName:
TerminalSessionId: 0
User: NT AUTHORITY\\SYSTEM
UtcTime: 2019-06-12 20:17:44.014
```

Additional fields

```
# All the ancestors of the process
Ancestors:
System|C:\\Windows\\System32\\smss.exe|C:\\Windows\\System32\\smss
.exe|C:\\Windows\\System32\\wininit.exe|C:\\Windows\\System32\\ser
vices.exe|C:\\ProgramData\\Microsoft\\Windows
Defender\\Platform\\4.18.1907.4-
0\\MsMpEng.exe|C:\\ProgramData\\Microsoft\\Windows
Defender\\Platform\\4.18.1907.4-0\\MpCmdRun.exe

# Image Size
ImageSize: 885760

# Integrity Level of parent process
ParentIntegrityLevel: System
# Integrity metric (compared to disk image)
ProcessIntegrity: 0

# Integrity metric of parent process
ParentProcessIntegrity: 0

# Wether or not integrity computation timed out
IntegrityTimeout: false

# Name of the service(s) associated to the process
Services: N/A
```

What’s this Gene thing?

Gene is the detection engine of WHIDS so I need to explain you what it is.

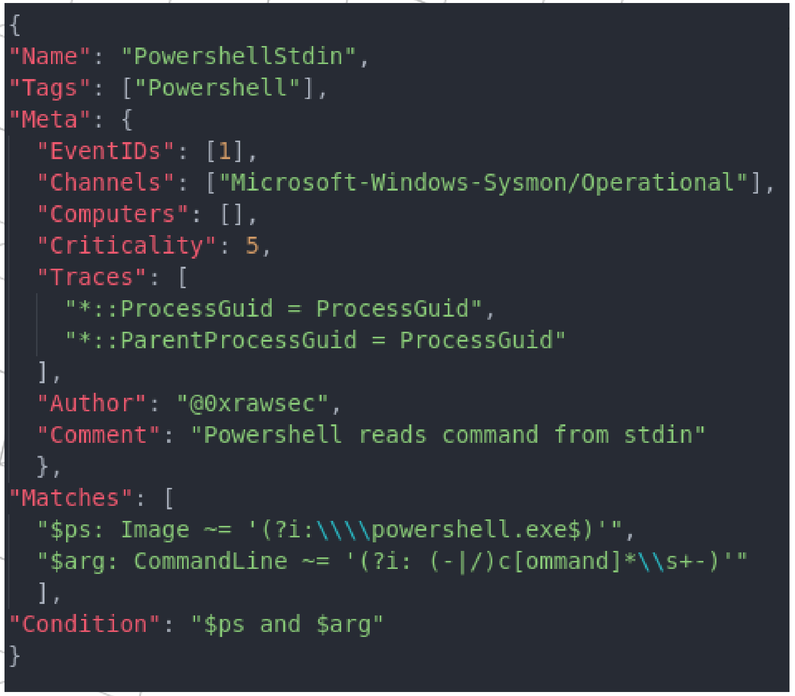
Gene is at the origin of everything…

- **What**: an engine and a rule format designed to detect patterns in Windows Event Logs. It was developed **prior to WHIDS** for Incident Response purposes.
- **Why**: any Windows Event can be considered as an **IOC** so it make sense to have a tool / rule format, to catch them

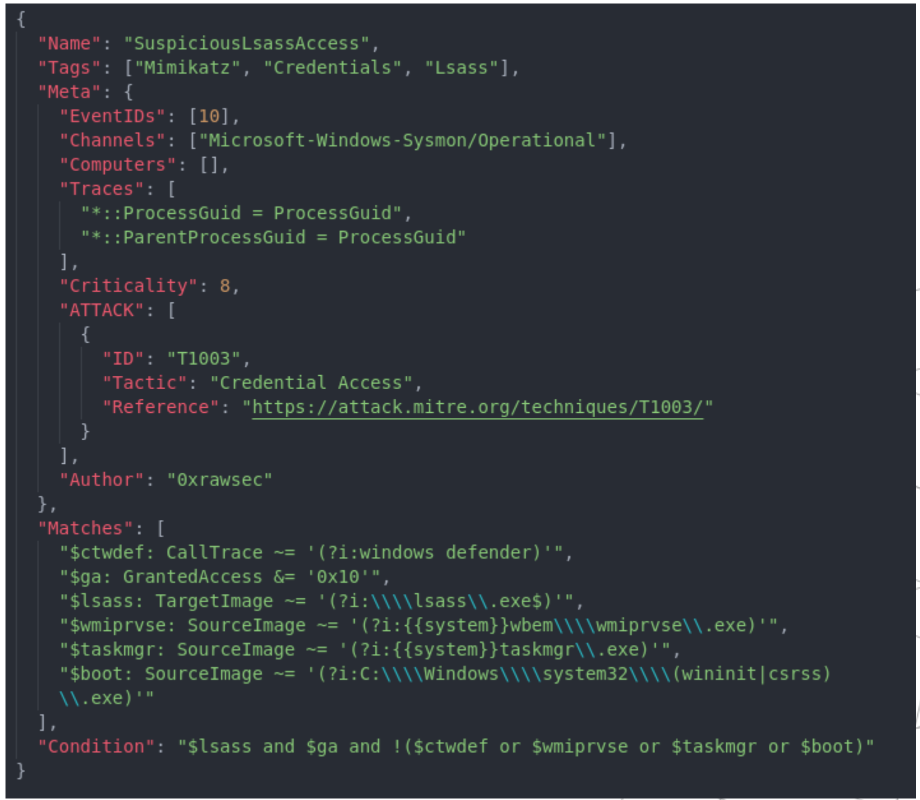
You can see it as a Yara engine but to match against Windows Event Logs

Give me an example !

We can do pretty complex stuff!



```
{
"Name": "PowershellStdin",
"Tags": ["Powershell"],
"Meta": {
  "EventIDs": [1],
  "Channels": ["Microsoft-Windows-Sysmon/Operational"],
  "Computers": [],
  "Criticality": 5,
  "Traces": [
    "*::ProcessGuid = ProcessGuid",
    "*::ParentProcessGuid = ProcessGuid"
  ],
  "Author": "@0xrawsec",
  "Comment": "Powershell reads command from stdin"
  },
"Matches": [
  "$ps: Image ~= '(?i:\\\\powershell.exe$)'",
  "$arg: CommandLine ~= '(?i: (-|/)c[ommand]*\\s+-)'"
  ],
"Condition": "$ps and $arg"
}
```



```
{
  "Name": "SuspiciousLsassAccess",
  "Tags": ["Mimikatz", "Credentials", "Lsass"],
  "Meta": {
    "EventIDs": [10],
    "Channels": ["Microsoft-Windows-Sysmon/Operational"],
    "Computers": [],
    "Traces": [
      "*::ProcessGuid = ProcessGuid",
      "*::ParentProcessGuid = ProcessGuid"
    ],
    "Criticality": 8,
    "ATTACK": [
      {
        "ID": "T1003",
        "Tactic": "Credential Access",
        "Reference": "https://attack.mitre.org/techniques/T1003/"
      }
    ],
    "Author": "0xrawsec"
  },
  "Matches": [
    "$ctwdef: CallTrace ~= '(?i:windows defender)'",
    "$ga: GrantedAccess &= '0x10'",
    "$lsass: TargetImage ~= '(?i:\\\\lsass\\.exe$)'",
    "$wmiprvse: SourceImage ~= '(?i:{{system}}wbem\\\\wmiprvse\\.exe)'",
    "$taskmgr: SourceImage ~= '(?i:{{system}}taskmgr\\.exe)'",
    "$boot: SourceImage ~= '(?i:C:\\\\Windows\\\\system32\\\\(wininit|csrss)
    \\.exe)'"
  ],
  "Condition": "$lsass and $ga and !($ctwdef or $wmiprvse or $taskmgr or $boot)"
}
```

Documentation: https://rawsec.lu/doc/gene/1.6

And when do you talk about MISP?

I recently plugged MISP and WHIDS together to benefit from IOCs present in MISP

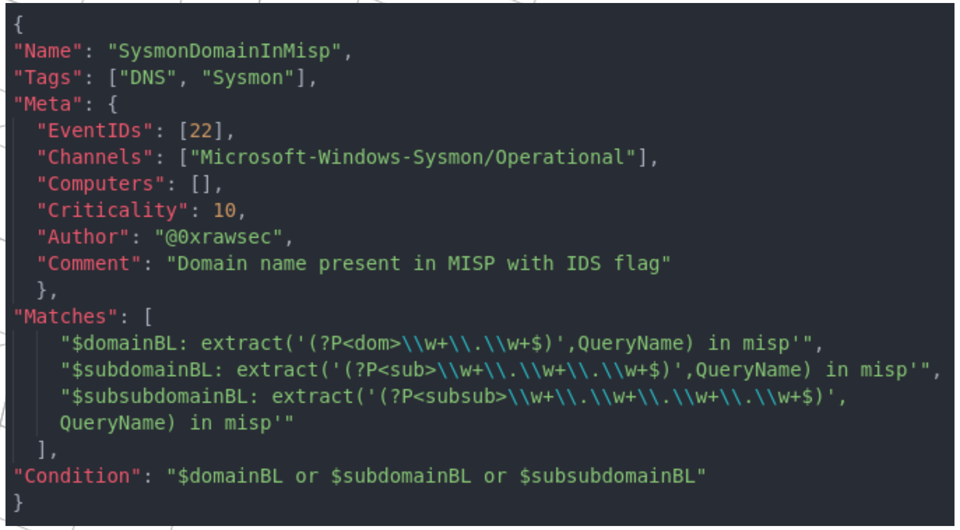
The challenges:

- **Performance**: IOCs usually come in mass, and you don't want your IOC checking process to be slow (especially in real time processing)
- **Scalability**: detection time should not be impacted while the number of IOCs increases
- **Flexibility**: make possible the match of only a **sub-part** of an event field
- Some IOCs need to be matched case **insensitively** (registry keys, paths …)

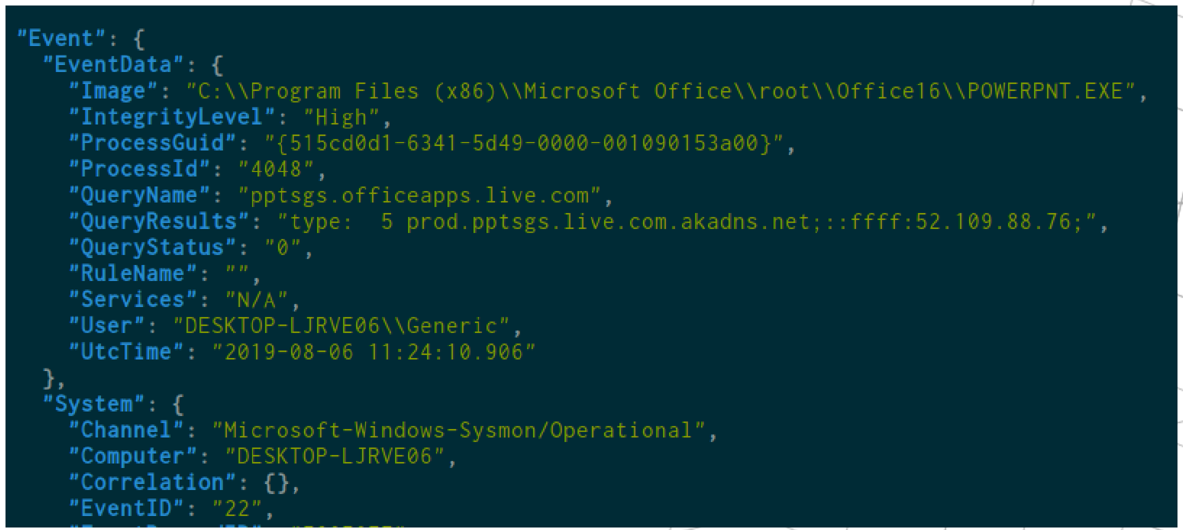
Gene to the rescue!

Hopefully Gene comes with a handy feature called **container match**: we can extract part of an event and check this against a container:

- Set data structure **O[1]** for lookup (**performance + scalability**)
- Store data **case insensitive**



```
{
"Name": "SysmonDomainInMisp",
"Tags": ["DNS", "Sysmon"],
"Meta": {
  "EventIDs": [22],
  "Channels": ["Microsoft-Windows-Sysmon/Operational"],
  "Computers": [],
  "Criticality": 10,
  "Author": "@0xrawsec",
  "Comment": "Domain name present in MISP with IDS flag"
  },
"Matches": [
    "$domainBL: extract('(?P<dom>\\w+\\.\\w+$)',QueryName) in misp'",
    "$subdomainBL: extract('(?P<sub>\\w+\\.\\w+\\.\\w+$)',QueryName) in misp'",
    "$subsubdomainBL: extract('(?P<subsub>\\w+\\.\\w+\\.\\w+\\.\\w+$)',
    QueryName) in misp'"
  ],
"Condition": "$domainBL or $subdomainBL or $subsubdomainBL"
}
```



```
"Event": {
  "EventData": {
    "Image": "C:\\Program Files (x86)\\Microsoft Office\\root\\Office16\\POWERPNT.EXE",
    "IntegrityLevel": "High",
    "ProcessGuid": "{515cd0d1-6341-5d49-0000-001090153a00}",
    "ProcessId": "4048",
    "QueryName": "pptsgs.officeapps.live.com",
    "QueryResults": "type:  5 prod.pptsgs.live.com.akadns.net;::ffff:52.109.88.76;",
    "QueryStatus": "0",
    "RuleName": "",
    "Services": "N/A",
    "User": "DESKTOP-LJRVE06\\Generic",
    "UtcTime": "2019-08-06 11:24:10.906"
  },
  "System": {
    "Channel": "Microsoft-Windows-Sysmon/Operational",
    "Computer": "DESKTOP-LJRVE06",
    "Correlation": {},
    "EventID": "22",
```

So what’s needed to be done?

A nice integration of this into WHIDS

- MISP integration is done only for WHIDS running **with a central manager** (not to have MISP API key exposed on the endpoint)
- MISP IOCs are pulled **periodically** on the manager and updated on the endpoints running WHIDS
- **Not all** MISP IOCs can be used, only the ones with **IDS flag** and belonging to those categories:
  - Md5 / sha1 /sha256
  - Hostname / domain
- **Registry keys**: not that easy to integrate since they can contain variable parts + not so many registry keys with IDS flag in MISP
- **Filename**: not the priority because it is not a strong IOC
- Supporting new IOC types is just a matter of adding a line of code and creating rules to match against events

Integration with MISP is available since WHIDS 1.6.2 (~~yesterday~~ today’s commit)

RawSec

Where is your demo ?

RawSec

Q&A

Thank you !

References:

- WHIDS: https://github.com/0xrawsec/whids
- Gene: https://github.com/0xrawsec/gene
- Gene rules: https://github.com/0xrawsec/gene-rules
- Gene Documentation: https://rawsec.lu/doc/gene/1.6

I give a training about WHIDS on Thursday so feel free to come