

SOC

Le centre d'opération de cyberdéfense est là pour vous protéger !

- [\[SOC\] Outils de cyber défense](#)
- [\[SOC\] Afficher l'empreinte d'un fichier](#)
- [\[SOC\] YARA](#)
- [\[SOC\] Dettect](#)

[SOC] Outils de cyber défense

Introduction

Cette page répertorie une base d'outils de cyber défense utile à un SOC ou même à un administrateur aguéri et conscient des enjeux cyber.



Scanner d'URL

Une multitude d'outils existent pour scanner les URL grâce à leur réputation.

Voici une liste d'outils :

- [VirusTotal](#)

- [MetaDefender Cloud - OPSWAT](#)
- [URLscan.io](#)
- [URLhaus](#)
- [Whols](#)
- [Cisco Talos Intelligence](#)
- [ThreatMiner](#)
- [Brightcloud](#)

Scanner d'adresse IP

Voici une liste d'outils en ligne pour analyser la réputation des adresses IP :

- [AbuseIPDB](#)
- [MetaDefender Cloud - OPSWAT](#)
- [Cisco Talos Intelligence](#)
- [Feodo Tracker](#)
- [Threatbook](#)
- [Pulsedive](#)
- [Shodan](#)
- [XForcelBM](#)
- [Alienvault](#)
- [GreyNoise](#)

Scanner de fichier et de hash

Voici une liste d'outils en ligne pour analyser des fichiers et des hashes de fichiers :

- [VirusTotal](#)
- [MetaDefender Cloud - OPSWAT](#)
- [MalwareBazaar](#)
- [Malshare](#)
- [Cisco Talos Intelligence](#)
- [HybridAnalysis](#)

Règles de détection

- [SOC Prime Threat Detection Marketplace](#)

Blacklist d'empreintes de certificats SSL

- [SSL Blacklist](#)

Scanner de mail

Pour se protéger du phishing, il peut être utile d'utiliser des outils qui vont décortiquer le mail afin de potentiellement trouver des IOC.

- [PhishTool](#)

Sandbox

Les sandboxes permettent de tester le comportement de fichier ou de site web afin de détecter un comportement malveillant dans un environnement protégé.

- [Browserling](#) : Accéder à un site web dans une sandbox.
- [AnyRun](#) : Lancer un exécutable Windows dans une sandbox.
- [Wannabrowser](#) : Accéder à un site web dans une sandbox.

Matrice MITRE ATT&CK

Framework permettant notamment de lister les TTPs et les APT.

- [MITRE ATT&CK](#)

Liste d'IOC multiples

- ThreatFox : MISP events, Suricata IDS Ruleset, Domain Host files, DNS Response Policy Zone, JSON files and CSV files.

PS - Obtenir le hash d'un fichier

```
Get-FileHash <FILE> -Algorithm MD5
```

[SOC] Afficher l'empreinte d'un fichier

Introduction

Dans le cadre du travail d'analyste, il peut être intéressant d'obtenir l'empreinte d'un fichier sous ses différentes formes (**MD5**, **SHA-1** ou **SHA-256**).

Ce tutoriel traitera la méthodologie à suivre sous Linux exclusivement.



Manuel

MD5

```
md5sum <FILE>
```

SHA-1

```
sha1sum <FILE>
```

SHA-256

```
sha256sum <FILE>
```

Changer l'empreinte d'un fichier

Il est possible de changer l'empreinte d'un fichier sans (presque) altérer son contenu en ajoutant un **null byte** à la fin de celui-ci :

```
echo -n -e "\x00" >> file.txt
```

Cette technique est très pratique pour échapper à la détection basée sur la signature mais permet aussi d'envoyer des échantillons de malware sur des plateformes publiques sans que celui-ci soit retrouvé par l'éditeur.

[SOC] YARA

Introduction

Le **YARA** est un langage pour écrire des règles de détection de malware.

Il s'agit d'un langage simple et descriptif qui est adopté par le grand public.

Il est découpé par section qui ont chacune leur utilité.



Source

- [TryHackMe - Yara](#)
- [Cuckoosandbox - Sandbox pour tester vos règles Yara](#)
- [PEfile - Scan les exécutables Windows PE](#)

Annexes

- [Github - Awesome Yara](#)
- [Valhalla - Base de règles Yara Opensource](#)

Anatomie d'une règle

ANATOMY OF A YARA RULE



Yara is a tool used to identify file, based on **textual or binary pattern**.



A rule consists of a **set of strings and conditions** that determine its logic.



Rules can be compiled with "yacc" to **increase the speed** of multiple Yara scans.

1

IMPORT MODULE

Yara modules allow you to extend its functionality. The PE module can be used to match specific data from a PE:

- `pe.number_of_exports`
- `pe.sections[0].name`
- `pe.imphash()`
- `pe.imports("kernel32.dll")`
- `pe.is_dll()`

List of modules: `pe`, `elf`, `hash`, `math`, `cuckoo`, `dotnet`, `time`

2

RULE NAME

The rule name identifies your Yara rule. It is recommended to add a meaningful name. There are different types of rules:

- Global rules: applies for all your rules in the file.
- Private rules: can be called in a condition of a rule but not reported.
- Rule tags: used to filter yara's output.

3

METADATA

Rules can also have a metadata section where you can put additional information about your rule.

- Author
- Date
- Description
- Etc...

4

STRINGS

The field strings is used to define the strings that should match your rule. It exists 3 type of strings:

- Text strings
- Hexadecimal strings
- Regex

5

CONDITION

Conditions are Boolean expressions used to match the defined pattern.

- Boolean operators:
 - `and`, `or`, `not`
 - `<`, `>`, `==`, `<`, `>`, `!=`
- Arithmetic operators:
 - `+`, `-`, `*`, `/`, `%`
- Bitwise operators:
 - `&`, `|`, `<<`, `>>`, `^`, `~`
- Counting strings:
 - `#string0 == 5`
- Strings offset:
 - `$string1 at 100`

```
import "pe"

rule demo_rule : Tag1 Demo
{
    meta:
        author = "Thomas Roccia"
        description = "demo"
        hash = ""

    strings:
        $string0 = "hello" nocase wide
        $string1 = "world" fullword ascii
        $hex1 = { 01 23 45 ?? 89 ab cd ef }
        $re1 = /md5: [0-9a-zA-Z]{32}/

    condition:
        uint16(0) == 0x5A4D and filesize < 2000KB
        or pe.number_of_sections == 1 and
        any of ($string*) and (not $hex1 or $re1)
}
```

TEXT STRINGS

Text strings can be used with modifiers:

- `nocase`: case insensitive
- `wide`: encoded strings with 2 bytes per character
- `fullword`: non alphanumeric
- `xor(0x01-0xFF)`: look for xor encryption
- `base64`: base64 encoding

HEXADECIMAL

Hex strings can be used to match piece of code:

- Wild-cards: `{ 00 ?2 A? }`
- Jump: `{ 3B [2-4] B4 }`
- Alternatives: `{ F4 (B4 | 56) }`

REGEX

Regular expression can also be used and defined as text strings but enclosed in forward slash.

ADVANCED CONDITION

- Accessing data at a given position: `uint16(0) == 0x5A4D`
- Check the size of the file: `filesize < 2000KB`
- Set of strings: `any of ($string0, $hex1)`
- Same condition to many strings: `for all of them : (# > 3)`
- Scan entry point: `$value at pe.entry_point`
- Match length: `!re1[1] == 32`
- Search within a range of offsets: `$value in (0..100)`

 @FR0GGER_
THOMAS ROCCIA

Manuel

Installation

```
apt install -y yara
```

Meta

Cette section permet de donner des informations complémentaires qui ne seront pas interprétés comme le ferait un commentaire dans du code.

Par exemple on peut utiliser le mot-clé **desc**, pour donner une description à notre règle afin qu'elle soit plus explicite pour les utilisateurs.

Strings

Cette section permet de détecter des chaînes de caractères présente dans les fichiers.

Voici un exemple d'utilisation :

```
rule helloworld_checker{
  strings:
    $hello_world = "Hello World!"

  condition:
    $hello_world
}
```

On peut aussi détecter des chaînes multiples :

```
rule helloworld_checker{
  strings:
    $hello_world = "Hello World!"
    $hello_world_lowercase = "hello world"
    $hello_world_uppercase = "HELLO WORLD"

  condition:
    any of them
}
```

Opérateurs

Comme dans les langages de programmation traditionnels, on peut utiliser des opérateurs pour nos conditions :

```
rule helloworld_checker{
  strings:
    $hello_world = "Hello World!"

  condition:
    #hello_world <= 10
}
```

| Opérateurs | Descriptions |
|------------|--------------------|
| <= | Plus petit ou égal |
| >= | Plus grand ou égal |
| != | Différent de |

Combinaisons

On peut utiliser les mot-clés suivants pour combiner nos conditions :

| Mot-clés | Descriptions |
|----------|---|
| and | Les deux conditions doivent être valides |
| or | Au moins l'une des deux conditions doit être valide |
| not | Inverse la condition (true devient false et false devient true) |

Voici un exemple pour vérifier si la chaîne est présente et si la taille du fichier est inférieure à 10KB :

```
rule helloworld_checker{
  strings:
    $hello_world = "Hello World!"

  condition:
    $hello_world and filesize < 10KB
}
```

Lancer le scan

```
yara <RULE>.yar <FILE_TO_SCAN>
```

Si la règle match, la commande renverra le nom de la règle qui a matchée ainsi que le nom du fichier qui a matché.

Scanners d'IOC basés sur Yara

- [Loki](#)
- [THOR](#)
- [Fenrir](#)
- [YAYA](#)

Loki

Mettre à jour la base de signature

```
python loki.py --update
```

Lancer un scan d'un dossier

```
python loki.py -p <DIR>
```

YarGen

Cet outil permet de créer une règle Yara à partir d'un ou plusieurs fichiers connus pour être malveillants.

Il va se baser sur les chaînes de caractères et les informations pour générer la règle qui va détecter le ou les fichiers.

Téléchargement

- [Github - YarGen](#)

Mettre à jour l'outil

```
python3 yarGen.py --update
```

Cela va mettre à jour la base avec les chaînes et les opcodes.

Créer une règle

```
python3 yarGen.py -m <FILE_PATH> --excludegood -o <OUTPUT.yar>
```

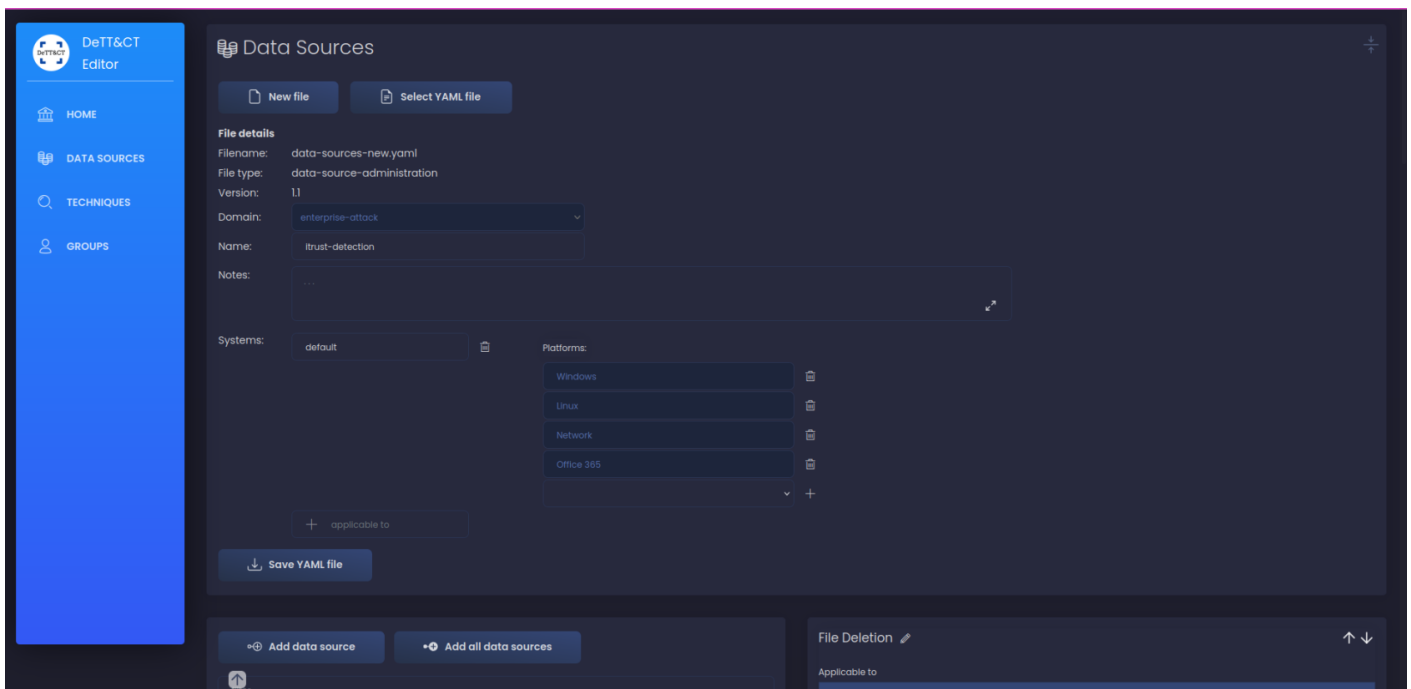
Bien que l'outil soit fonctionnel, il est recommandé d'éditer la règle pour supprimer les chaînes qui pourraient lever des faux-positifs.

[SOC] Dettect

Introduction

Le projet Dettect a pour objectif d'identifier les TTPs couvertes (et non-couvertes) par vos règles de détection.

Le projet vous aidera à générer un fichier avec vos datasources couvertes et à convertir ce fichier en un fichier importable dans le MITRE Navigator afin d'afficher les TTPs.



DeTT&CT

Voici le lien du projet :

- <https://github.com/rabobank-cdc/DeTTECT>

Lancez le conteneur :

```
docker run -p 8080:8080 -v $(pwd)/output:/opt/DeTTECT/output -v $(pwd)/input:/opt/DeTTECT/input --name  
dettect -it rabobankcdc/dettect:latest /bin/bash
```

Puis lancez le serveur web en écoute :

```
python3 detect.py e
```

Vous pouvez ouvrir un navigateur web et vous rendre sur <http://localhost:8080> .

- Après avoir créer vos datasources et télécharger votre configuration, déplacez-le dans le dossier input du projet.
- Ensuite ouvrez un shell dans le conteneur :

```
docker exec -it detect bash
```

Puis convertissez pour obtenir un fichier de configuration importable dans le MITRE Navigator :

```
python3 detect.py ds -fd input/data-sources-new.yaml -l
```

Et importez dans le MITRE Navigator :

| | | | | | | | | | | | | | |
|---|-----------------------------------|------------------------------------|---------------------------------------|---|--------------------------------------|-------------------------------|---------------------------------------|--------------------------------------|---------------------------------------|--|-------------------------------|--|--|
| ATT&CKcon 5.0 returns October 22-23, 2024 in McLean, VA. Register here today! | | | | | | | | | | | | MITRE ATT&CK® | |
| Data sources itrust-detection x + | | | | | | | | | | | | Selection Controls Layer Controls Technique Controls | |
| | | | | | | | | | | | | Q X 🔒 ⚙️ | |
| Initial Access 10 techniques | Execution 11 techniques | Persistence 19 techniques | Privilege Escalation 14 techniques | Defense Evasion 38 techniques | Credential Access 17 techniques | Discovery 29 techniques | Lateral Movement 9 techniques | Collection 17 techniques | Command and Control 18 techniques | Exfiltration 9 techniques | Impact 14 techniques | | |
| Content Injection | Command and Scripting Interpreter | Account Manipulation | Abuse Elevation Control Mechanism | Abuse Elevation Control Mechanism | Adversary-in-the-Middle | Account Discovery | Exploitation of Remote Services | Adversary-in-the-Middle | Application Layer Protocol | Automated Exfiltration | Account Access Removal | | |
| Drive-by Compromise | AutoHotKey & AutoIT | Additional Cloud Roles | Bypass User Account Control | Bypass User Account Control | ARP Cache Poisoning | Cloud Account | Internal Spearphishing | ARP Cache Poisoning | File Transfer Protocols | Traffic Duplication | Data Destruction | | |
| Exploit Public Facing Application | Cloud API | Device Registration | Setuid and Setgid | Setuid and Setgid | DHCP Spoofing | Domain Account | Lateral Tool Transfer | DHCP Spoofing | Mail Protocols | Data Transfer Size Limits | Data Encrypted for Impact | | |
| External Remote Services | JavaScript | SSH Authorized Keys | Sudo and Sudo Caching | Sudo and Sudo Caching | LLMNR/NBT-NS Poisoning and SMB Relay | Email Account | Remote Service Session Hijacking | LLMNR/NBT-NS Poisoning and SMB Relay | Web Protocols | Exfiltration Over Alternative Protocol | Data Manipulation | | |
| Hardware Additions | Network Device CLI | BITS Jobs | Temporary Elevated Cloud Access | Temporary Elevated Cloud Access | Brute Force | Local Account | RDP Hijacking | Archive Collected Data | Communication Through Removable Media | Exfiltration Over Symmetric Encrypted Non-C2 Protocol | Runtime Data Manipulation | | |
| Phishing | PowerShell | Boot or Logon Autostart Execution | Access Token Manipulation | Access Token Manipulation | Credential Stuffing | Application Window Discovery | SSH Hijacking | Archive via Custom Method | Content Injection | Exfiltration Over Asymmetric Encrypted Non-C2 Protocol | Stored Data Manipulation | | |
| Spearphishing Attachment | Python | Active Setup | Create Process with Token | Create Process with Token | Password Cracking | Browser Information Discovery | Cloud Services | Archive via Library | Data Encoding | Exfiltration Over Symmetric Encrypted Non-C2 Protocol | Transmitted Data Manipulation | | |
| Spearphishing Link | Unix Shell | Authentication Package | Make and Impersonate Token | Make and Impersonate Token | Password Guessing | Cloud Service Dashboard | Distributed Component Object Model | Audio Capture | Non-Standard Encoding | Exfiltration Over Encrypted Non-C2 Protocol | Defacement | | |
| Spearphishing via Service | Visual Basic | Kernel Modules and Extensions | Parent PID Spoofing | SID-History Injection | Password Spraying | Device Driver Discovery | Remote Desktop Protocol | Automated Collection | Standard Encoding | Exfiltration Over Unencrypted Non-C2 Protocol | External Defacement | | |
| Spearphishing Voice | Windows Command Shell | LSASS Driver | SID-History Injection | Token Impersonation/Theft | Credentials from Password Stores | Domain Trust Discovery | SMB/Windows Admin Shares | Clipboard Data | Data Obfuscation | Exfiltration Over C2 Channel | Disk Wipe | | |
| Replication Through Removable Media | Exploitation for Client Execution | Port Monitors | Token Impersonation/Theft | Debugger Evasion | Credentials from Web Browsers | Securityd Memory | SSH | Steganography | Junk Data | Exfiltration Over Other Network Medium | Disk Structure Wipe | | |
| Supply Chain Compromise | Inter-Process Communication | Print Processors | Account Manipulation | Deobfuscate/Decode Files or Information | Password Managers | Windows Credential Manager | VNC | Dynamic Resolution | Protocol Impersonation | Exfiltration Over Physical Medium | Endpoint Denial of Service | | |
| Compromise Hardware Supply Chain | Component Object Model | Registry Run Keys / Startup Folder | Additional Cloud Roles | Domain or Tenant Policy Modification | Windows Credential Manager | Log Enumeration | Windows Remote Management | DNS Calculation | Steganography | Exfiltration Over Bluetooth | Application Exhaustion Flood | | |
| Compromise Software Dependencies and Development Tools | Dynamic Data Exchange | Security Support Provider | Additional Email Delegate Permissions | Group Policy Modification | Exploitation for Credential Access | Network Service Discovery | Replication Through Removable Media | Domain Generation Algorithms | Dynamic Resolution | Exfiltration Over USB | OS Exhaustion Flood | | |
| Compromise Software Supply Chain | Native API | Shortcut Modification | Device Registration | Trust Modification | Forced Authentication | Network Share Discovery | Software Deployment Tools | Fast Flux DNS | Encrypted Channel | Exfiltration Over Web Service | Service Exhaustion Flood | | |
| Trusted Relationship | Scheduled Task/Job | Time Providers | SSH Authorized Keys | Execution Guardrails | SAML Tokens | Network Sniffing | Taint Shared Content | Asymmetric Cryptography | Data from Configuration Repository | Exfiltration Over Webhook | | | |
| Valid Accounts | At | XDG Autostart Entries | Boot or Logon Autostart Execution | Environmental Keying | Web Cookies | Password Policy Discovery | Use Alternate Authentication Material | Sharepoint | Data from Information Repositories | | | | |
| Cloud Accounts | Cron | Logon Script (Windows) | Active Setup | Exploitation for Defense Evasion | Input Capture | Peripheral Device Discovery | Application Access Token | | Sharepoint | | | | |
| Default Accounts | Scheduled Task | Network Logon Script | Kernel Modules and Extensions | File and Directory Permissions Modification | Credential API Hooking | | | | Sharepoint | | | | |
| | System Timers | RC Scripts | LSASS Driver | Linux and Mac File and Directory | | | | | Data from Local | | | | |
| | Serverless Execution | | | | | | | | | | | | |