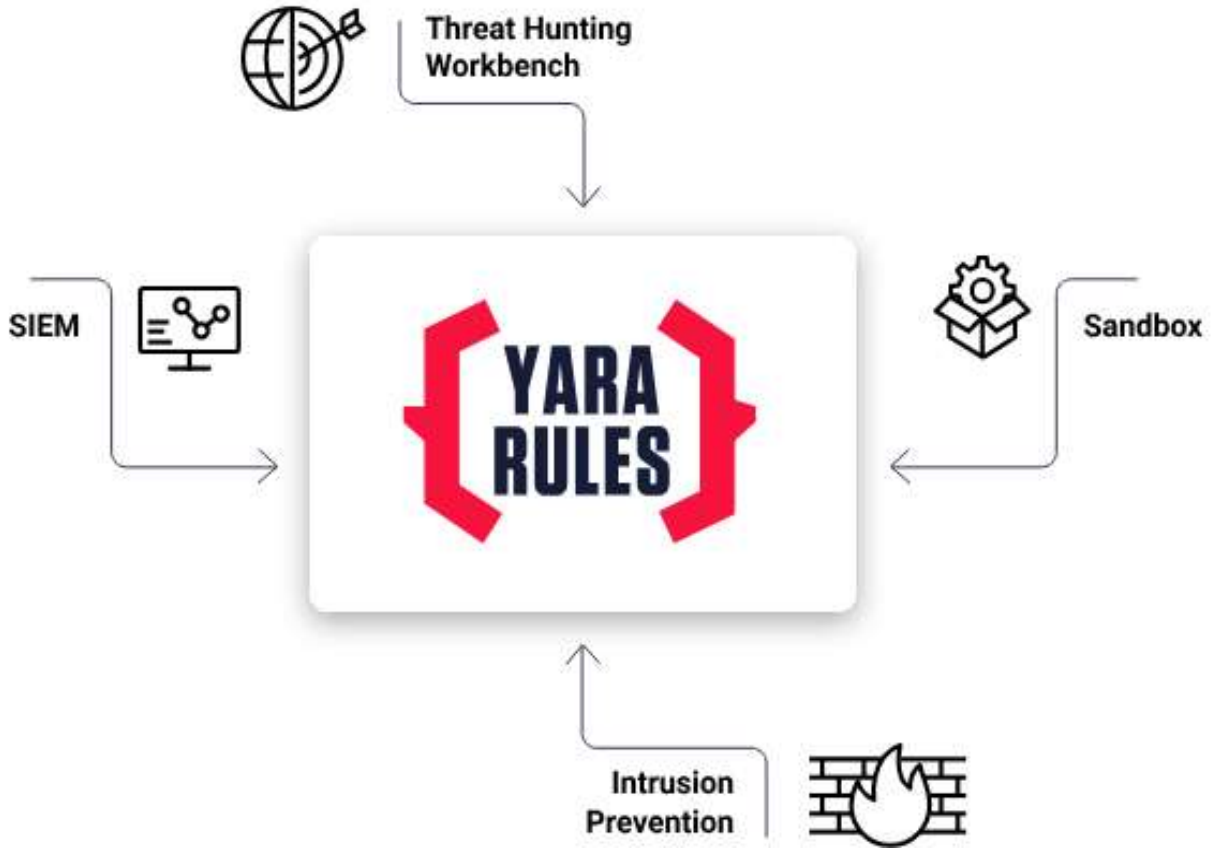
MÓDULO ANÁLISIS DE MALWARE
REGLAS YARA



Empresa: Keepcoding
Alumna: Raquel Suárez
Fecha:02-05-2025
Profesor: Adrian Rodriguez

ÍNDICE

Introducción

Las Reglas Yara Son patrones que se crean para buscar características específicas en archivos, como texto o secuencias de bytes.

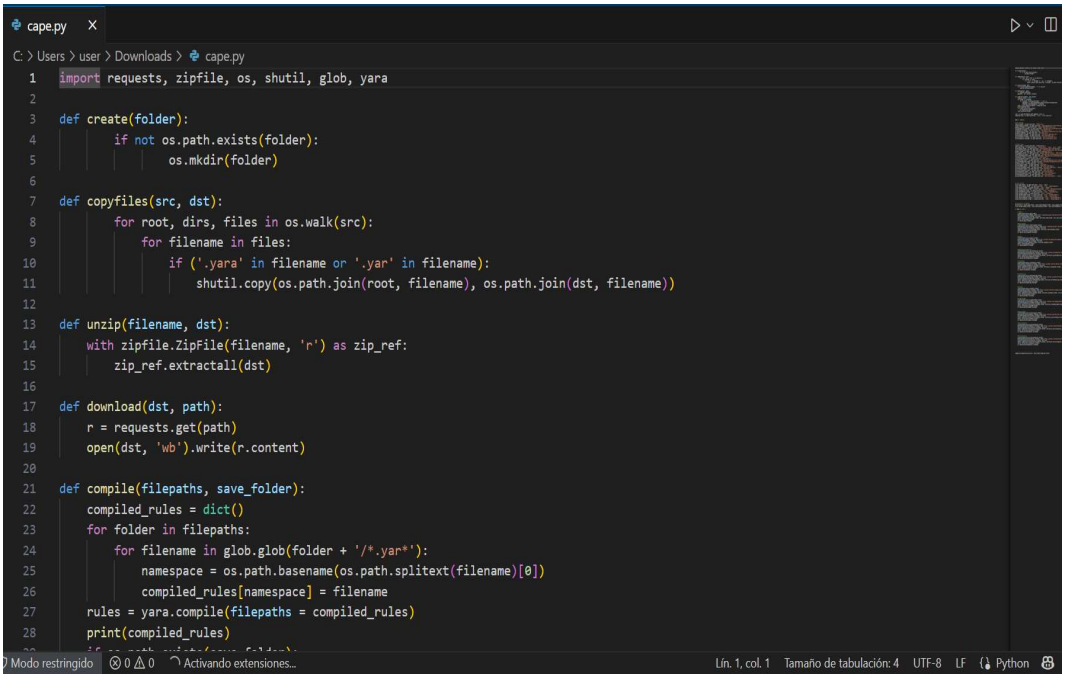
¿Para qué se utilizan?

Se usan para identificar y clasificar malware al comparar esos patrones con archivos sospechosos. Ayudan a los analistas de seguridad a encontrar y manejar amenazas rápidamente.

En resumen, **YARA** permite crear "filtros" que ayudan a detectar archivos maliciosos de forma eficiente.

Recopilación y Script Reglas Yara

Para la recopilación de Reglas Yara hemos utilizado diferentes repositorios de GitHub mencionados al final de esta memoria.
Una vez descargados dichos repositorios hemos creado un Script con todas las reglas recopiladas.



```python
# Zip filename
cape_filename = os.path.join(root, 'CAPEv2.zip')
reversinglabs_filename = os.path.join(root, 'reversinglabs-yara-rules-develop.zip')
apophis_filename = os.path.join(root, 'apophis-YARA-Rules-main.zip')
countermeasures_filename = os.path.join(root, 'red_team_tool_countermeasures-master.zip')
rulesmaster_filename = os.path.join(root, 'rules-master.zip')
ThreatHunting_filename = os.path.join(root, 'ThreatHunting-Keywords-yara-rules-main.zip')
yaramain_filename = os.path.join(root, 'yara-main.zip')
yararules2025_filename = os.path.join(root, 'Yara_Rules_2025-main.zip')
yararulesmain_filename = os.path.join(root, 'yara_rules-main.zip')
yararulesmaster_filename = os.path.join(root, 'yara-rules-master.zip')
Yararulesmaster_filename = os.path.join(root, 'Yara-rules-master.zip')
```

```python
# Folder unzip
capev2_folder = os.path.join(root, 'CAPEv2-master')
yara_cape_folder = os.path.join(root, 'CAPEv2-master', 'data', 'yara', 'CAPE')
reversinglab_folder = os.path.join(root, 'reversinglabs-yara-rules-develop')
yara_reversinglab_folder = os.path.join(root, 'reversinglabs-yara-rules-develop', 'yara')
apophis_folder = os.path.join(root, 'apophis-YARA-Rules-main')
yara_apophis_folder = os.path.join(root, 'apophis-YARA-Rules-main', 'YARA-rules')
countermeasures_folder = os.path.join(root, 'red_team_tool_countermeasures-master')
yara_countermeasures_folder = os.path.join(root, 'red_team_tool_countermeasures-master', 'rules')
rulesmaster_folder = os.path.join(root, 'rules-master')
yara_rulesmaster_folder = os.path.join(root, 'rules-master')
ThreatHunting_folder = os.path.join(root, 'ThreatHunting-Keywords-yara-rules-main')
yara_ThreatHunting_folder = os.path.join(root, 'ThreatHunting-Keywords-yara-rules-main')
yaramain_folder = os.path.join(root, 'yara-main')
yara_yaramain_folder = os.path.join(root, 'yara-main')
yararules2025_folder = os.path.join(root, (variable) root: str )
yara_yararules2025_folder = os.path.join(root, 'Yara_Rules_2025-main', 'Rules_for_January')
yararulesmain_folder = os.path.join(root, 'yara_rules-main')
yara_yararulesmain_folder = os.path.join(root, 'yara_rules-main')
yararulesmaster_folder = os.path.join(root, 'yara-rules-master')
yara_yararulesmaster_folder = os.path.join(root, 'yara-rules-master')
Yararulesmaster_folder = os.path.join(root, 'Yara-rules-master')
yara_Yararulesmaster_folder = os.path.join(root, 'Yara-rules-master', 'rules')
```

```python
# Local folders
local_cape_folder = os.path.join(root, 'rules', 'Cape')
local_reversinglabs_folder = os.path.join(root, 'rules', 'ReversingLabs')
local_apophis_folder = os.path.join(root, 'rules', 'Apophis')
local_countermeasures_folder = os.path.join(root, 'rules', 'RedTeamContermeasures')
local_rulesmaster_folder = os.path.join(root, 'rules', 'RulesMaster')
local_ThreatHunting_folder = os.path.join(root, 'rules', 'ThreatHunting')
local_yaramain_folder = os.path.join(root, 'rules', 'Yaramain')
local_YaraRules2025_folder = os.path.join(root, 'rules', 'YaraRules2025')
local_yararulesmain_folder = os.path.join(root, 'rules', 'Yararulesmain')
local_yararulesmaster_folder = os.path.join(root, 'rules', 'Yararulesmaster')
local_Yararulesmaster_folder = os.path.join(root, 'rules', 'YaraRulesmaster')
```

```python
if mode == 'full':

    # CAPEv2
    create(folder=local_cape_folder)
    download(dst=cape_filename, path='https://codeload.github.com/kevoreilly/CAPEv2/zip/refs/heads/master')
    unzip(filename=cape_filename, dst=root)
    shutil.copytree(src=yara_cape_folder, dst=local_cape_folder, dirs_exist_ok=True)
    shutil.rmtree(capev2_folder)
    os.remove(cape_filename)

    #ReversingLabs
    create(folder=local_reversinglabs_folder)
    download(dst=reversinglabs_filename, path='https://codeload.github.com/reversinglabs/reversinglabs-yara-rules/zip/refs/heads/develop')
    unzip(filename=reversinglabs_filename, dst=r (variable) local_reversinglabs_folder: str
    copyfiles(src=yara_reversinglab_folder, dst=local_reversinglabs_folder)
    shutil.rmtree(reversinglab_folder)
    os.remove(reversinglabs_filename)


    #Apophis
    create(folder=local_apophis_folder)
    download(dst=apophis_filename, path='https://github.com/apophis133/apophis-YARA-Rules/archive/refs/heads/main.zip')
    unzip(filename=apophis_filename, dst=root)
    copyfiles(src=yara_apophis_folder, dst=local_apophis_folder)
    shutil.rmtree(apophis_folder)
    os.remove(apophis_filename)
```

C: > Users > user > Downloads > cape.py > ...

```python
    #RedTeamCountermeasures
    create(folder=local_countermeasures_folder)
    download(dst=countermeasures_filename, path='https://github.com/mandiant/red_team_tool_countermeasures/archive/refs/heads/master.zip')
    unzip(filename=countermeasures_filename, dst=root)
    shutil.copytree(src=yara_countermeasures_folder, dst=local_countermeasures_folder, dirs_exist_ok=True)
    shutil.rmtree(countermeasures_folder)
    os.remove(countermeasures_filename)

    #rulesmaster
    create(folder=local_rulesmaster_folder)
    download(dst=rulesmaster_filename, path='https://github.com/Yara-Rules/rules/archive/refs/heads/master.zip')
    unzip(filename=rulesmaster_filename, dst=root)
    shutil.copytree(src=yara_rulesmaster_folder, dst=local_rulesmaster_folder, dirs_exist_ok=True)
    shutil.rmtree(rulesmaster_folder)
    os.remove(rulesmaster_filename)


    #ThreatHunting
    create(folder=local_ThreatHunting_folder)
    download(dst=ThreatHunting_filename, path='  (variable) root: str  cht/ThreatHunting-Keywords-yara-rules/archive/refs/heads/main.zip')
    unzip(filename=ThreatHunting_filename, dst=root)
    shutil.copytree(src=yara_ThreatHunting_folder, dst=local_ThreatHunting_folder, dirs_exist_ok=True)
    shutil.rmtree(ThreatHunting_folder)
    os.remove(ThreatHunting_filename)
```

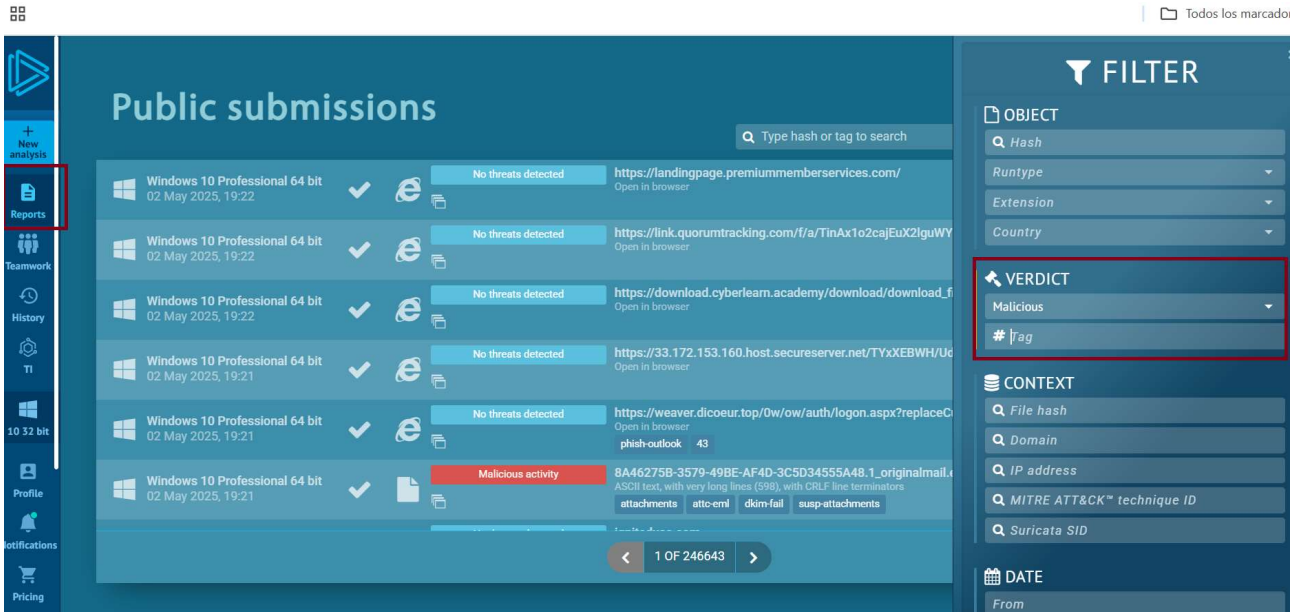
Compilar Reglas Yara

Para esta parte de la práctica lo que hemos hecho una vez hecho el Script , es utilizar el comando **“python3.8 cape.py” (Cape.py es el Script de las Yara)**

Una vez las reglas se compilan aparecen en nuestra carpeta creada llamada **“rules”** cada uno de los repositorios usados con su carpeta y dentro de ellas todos los archivos.yar/.yara y el archivo correspondiente al compilado. **“rules-compiled”**

Descarga Malware

Una vez terminado el compilado de las Reglas Yara nos toca bajar el Malware. Para ello hemos mirado diferentes herramientas, como VirusShare, JoeSandBox pero hemos utilizado la herramienta AnyRun , para descargar diferentes tipos.



• Lanzar compilado

Para poder lanzar el compilado sobre los malware descargados , creamos un script que llamamos en este caso malware.py y lo lanzamos con el comando

"python3.8 malware.py"

Tenia 32 malware y me ha cogido los 32. Adjunto video por correo electrónico para que se vea , que aquí solo 3 capturas.

```
keepcoding@ubuntu:~/Downloads/malware1$ python3.8 malware.py

🔍 Iniciando escaneo YARA...
[+] Reglas YARA cargadas correctamente.

[+] Escaneando archivos en: /home/keepcoding/Downloads/malware1
```

malware.py
~/Downloads/malware1

```python
import yara
import os

# Ruta donde está el malware y las reglas YARA compiladas
MALWARE_DIR = os.path.expanduser("~/Downloads/malware1")
RULES_PATH = os.path.join(MALWARE_DIR, "rules-compiled")

def load_yara_rules():
    try:
        rules = yara.load(filepath=RULES_PATH)
        print("[+] Reglas YARA cargadas correctamente.")
        return rules
    except Exception as e:
        print(f"[-] Error cargando reglas YARA: {e}")
        return None

def scan_malware(rules):
    if not rules:
        return

    print(f"\n[+] Escaneando archivos en: {MALWARE_DIR}")

    for filename in os.listdir(MALWARE_DIR):
        file_path = os.path.join(MALWARE_DIR, filename)

        # Ignorar el archivo de reglas compiladas
        if filename == "rules-compiled":
            continue

        if os.path.isfile(file_path):
            try:
                matches = rules.match(filepath=file_path)
                if matches:
                    print(f"\n🔥 **Malware detectado en {filename}**")
                    for match in matches:
                        print(f"  - Regla: {match.rule}")
                        if match.meta.get("description"):
```

Python Tab Width: 8 Ln 1, Col 1 INS

```
[+] Escaneando archivos en: /home/keepcoding/Downloads/malware1

🔥 **Malware detectado en 539656b4e63ebaaa0a33ae6a0a5156f8.exe.bin**
  - Regla: maldoc_find_kernel32_base_method_1
  - Regla: maldoc_getEIP_method_1
  - Regla: IsPE32
  - Regla: IsWindowsGUI
  - Regla: FSG_v110_Eng_dulekxt_
  - Regla: SEH_Save
  - Regla: SEH_Init
  - Regla: maldoc_find_kernel32_base_method_1
  - Regla: maldoc_getEIP_method_1
  - Regla: DebuggerPattern__RDTSC
  - Regla: DebuggerPattern__CPUID
  - Regla: DebuggerPattern__SEH_Saves
  - Regla: DebuggerPattern__SEH_Inits

🔥 **Malware detectado en e8a091a84dd2ea7ee429135ff48e9f48f7787637ccb79f6c3eb42f34588bc684.exe.bin**
  - Regla: maldoc_find_kernel32_base_method_1
  - Regla: IsPE32
  - Regla: IsWindowsGUI
  - Regla: IsPacked
    Descripción: Entropy Check
  - Regla: HasRichSignature
    Descripción: Rich Signature Check
  - Regla: win_files_operation
    Descripción: Affect private profile
  - Regla: SEH_Save
  - Regla: SEH_Init
  - Regla: Check_OutputDebugStringA_iat
```

```
🔥 **Malware detectado en HEUR-Trojan-Ransom.Win32.Generic-0545f842ca2eb77bcac0fd17d6d0a8c607d7dbc8669709f3096e5c1828e1c049.exe**
  - Regla: Lockbit
    Descripción: Lockbit Payload
  - Regla: Win32_Ransomware_LockBit
    Descripción: Yara rule that detects LockBit ransomware.
  - Regla: SHA512_Constants
    Descripción: Look for SHA384/SHA512 constants
  - Regla: SHA2_BLAKE2_IVs
    Descripción: Look for SHA2/BLAKE2/Argon2 IVs
  - Regla: RijnDael_AES_CHAR
    Descripción: RijnDael AES (check2) [char]
  - Regla: maldoc_find_kernel32_base_method_1
  - Regla: IsPE32
  - Regla: IsWindowsGUI
  - Regla: HasRichSignature
    Descripción: Rich Signature Check
  - Regla: Microsoft_Visual_Cpp_v50v60_MFC
  - Regla: Borland_Delphi_30_additional
  - Regla: Borland_Delphi_30_
  - Regla: Borland_Delphi_v40_v50
  - Regla: Borland_Delphi_v30
  - Regla: Borland_Delphi_DLL
  - Regla: maldoc_function_prolog_signature
  - Regla: maldoc_find_kernel32_base_method_1
  - Regla: DebuggerPattern__CPUID
  - Regla: SHA512_Constants
    Descripción: Look for SHA384/SHA512 constants
```

```
- Regla: Chacha_256_constant
  Descripción: Look for 256-bit key Chacha stream cipher constant
- Regla: SHA3_constants
  Descripción: SHA-3 (Keccak) round constants
- Regla: maldoc_find_kernel32_base_method_1
- Regla: IsPE32
- Regla: IsWindowsGUI
- Regla: HasRichSignature
  Descripción: Rich Signature Check
- Regla: maldoc_function_prolog_signature
- Regla: maldoc_find_kernel32_base_method_1
- Regla: maldoc_suspicious_strings
- Regla: DebuggerPattern__RDTSC
- Regla: DebuggerPattern__CPUID

🔥 **Malware detectado en 787c063e49255e491cf9424cdb48759c.exe.bin**
- Regla: IsPE32
- Regla: IsWindowsGUI
- Regla: IsPacked
  Descripción: Entropy Check
- Regla: FSG_v110_Eng_dulekxt_
- Regla: vmdetect
  Descripción: Possibly employs anti-virtualization techniques
- Regla: DebuggerPattern__RDTSC
- Regla: DebuggerPattern__CPUID
- Regla: vmdetect
  Descripción: Possibly employs anti-virtualization techniques
✅ DarkComet.exe - No se detectaron amenazas.

🏁 Escaneo completado.
eepcoding@ubuntu:~/Downloads/malware1$
```

Repositorios utilizados de GitHub para Reglas Yara

1. https://github.com/DarkenCode/yara-rules.git
2. https://github.com/shsameer786/Yara_Rules_2025.git
3. https://github.com/reversinglabs/reversinglabs-yara-rules.git
4. https://github.com/apophis133/apophis-YARA-Rules.git
5. https://github.com/mthcht/ThreatHunting-Keywords-yara-rules.git
6. https://github.com/Yara-Rules/rules.git
7. https://github.com/kevoreilly/CAPEv2.git
8. https://github.com/bartblaze/Yara-rules.git
9. https://github.com/securitymagic/yara.git
10. https://github.com/mandiant/red_team_tool_countermeasures.git
11. https://github.com/f0wl/yara_rules.git