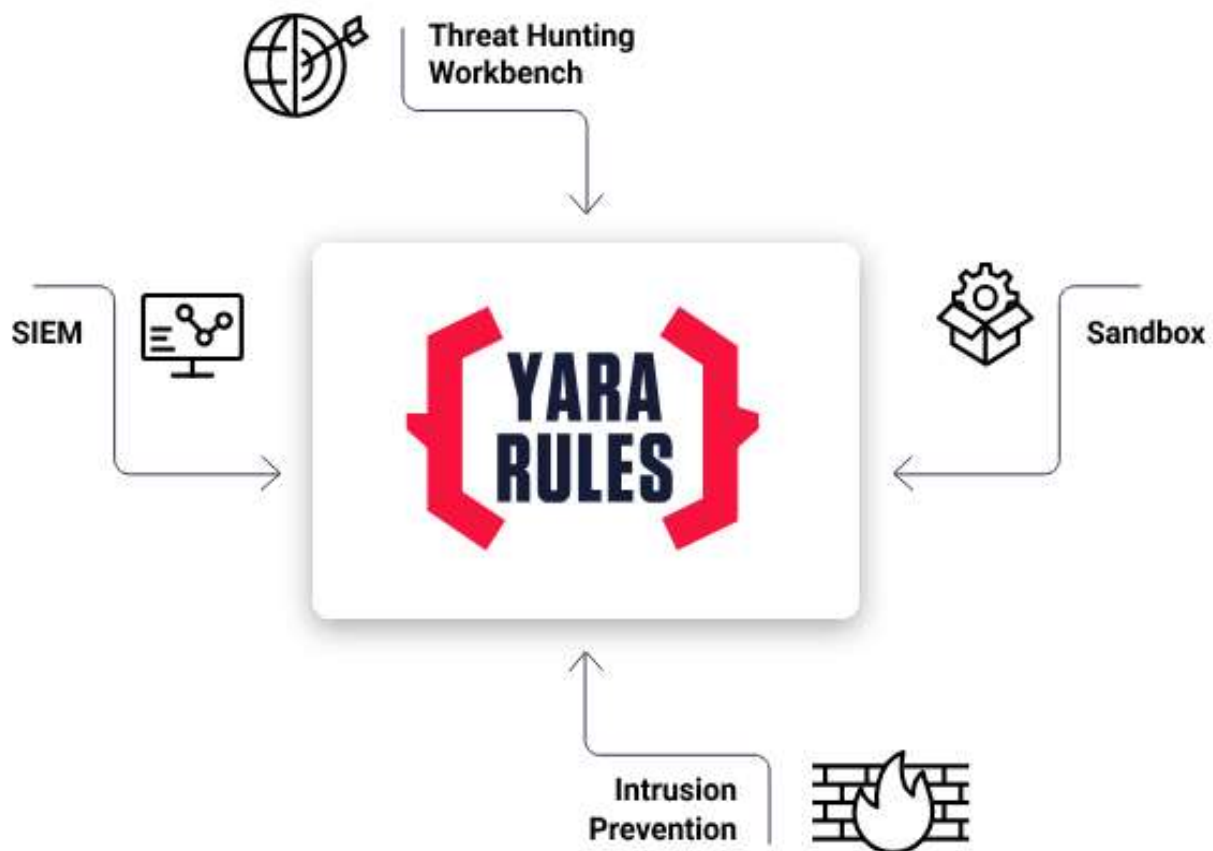


MÓDULO ANÁLISIS DE MALWARE REGLAS YARA



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

ÍNDICE

- Introducción
- Recopilación y Script
- Compilar Reglas Yara
- Descarga Malware
- Lanzar compilado
- Repositorios GitHub

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?

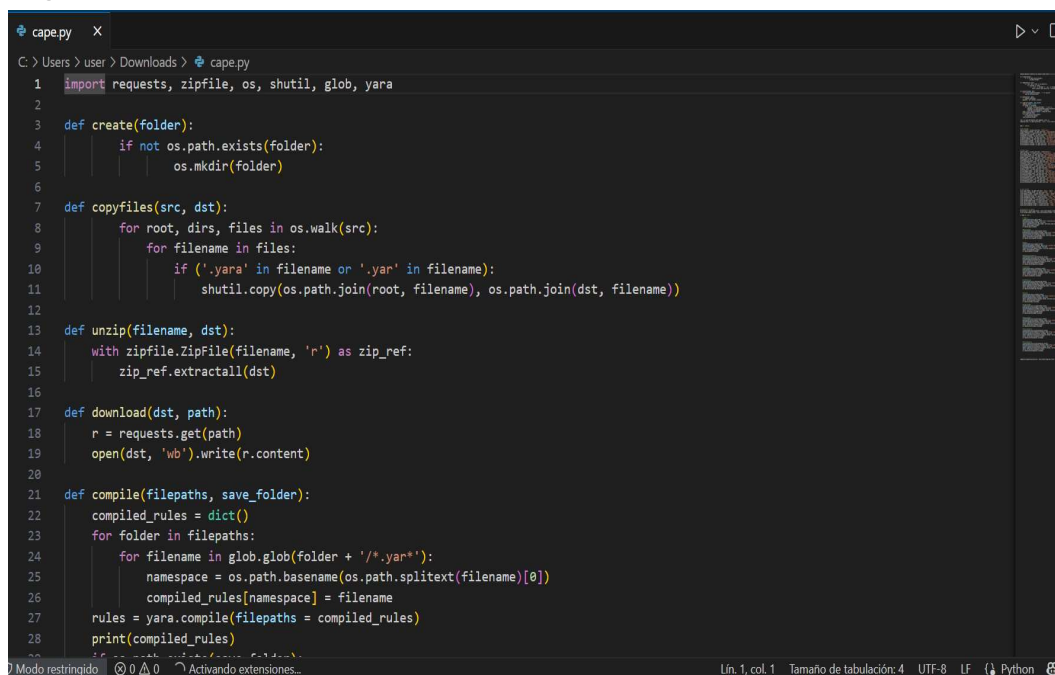
Se usan para identificar y clasificar malware al comparar esos patrones con archivos sospechosos. Ayudan a los analistas de seguridad a encontrar y manejar amenazas rápidamente.

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.



```
1 import requests, zipfile, os, shutil, glob, yara
2
3 def create(folder):
4     if not os.path.exists(folder):
5         os.mkdir(folder)
6
7 def copyfiles(src, dst):
8     for root, dirs, files in os.walk(src):
9         for filename in files:
10             if ('.yara' in filename or '.yar' in filename):
11                 shutil.copy(os.path.join(root, filename), os.path.join(dst, filename))
12
13 def unzip(filename, dst):
14     with zipfile.ZipFile(filename, 'r') as zip_ref:
15         zip_ref.extractall(dst)
16
17 def download(dst, path):
18     r = requests.get(path)
19     open(dst, 'wb').write(r.content)
20
21 def compile(filepaths, save_folder):
22     compiled_rules = dict()
23     for folder in filepaths:
24         for filename in glob.glob(folder + '/*.yar*'):
25             namespace = os.path.basename(os.path.splitext(filename)[0])
26             compiled_rules[namespace] = filename
27     rules = yara.compile(filepaths = compiled_rules)
28     print(compiled_rules)
```

```

# 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')
yara_rules2025_filename = os.path.join(root, 'Yara_Rules_2025-main.zip')
yara_rulesmain_filename = os.path.join(root, 'yara_rules-main.zip')
yara_rulesmaster_filename = os.path.join(root, 'yara-rules-master.zip')
Yara_rulesmaster_filename = os.path.join(root, 'Yara-rules-master.zip')

```

```

# 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')
yara_rules2025_folder = os.path.join(root, (variable) root: str ), 'Yara_Rules_2025-main', 'Rules_for_January')
yara_rulesmain_folder = os.path.join(root, 'yara_rules-main')
yara_rulesmaster_folder = os.path.join(root, 'yara-rules-master')
Yara_rulesmaster_folder = os.path.join(root, 'Yara-rules-master')
yara_Yara_rulesmaster_folder = os.path.join(root, 'Yara-rules-master', 'rules')

```

```

# 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_yara_rulesmain_folder = os.path.join(root, 'rules', 'YaraRulesmain')
local_yara_rulesmaster_folder = os.path.join(root, 'rules', 'YaraRulesmaster')
local_YaraRulesmaster_folder = os.path.join(root, 'rules', 'YaraRulesmaster')

```

```

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=(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 > ...
127 #RedTeamCountermeasures
128 create(folder=local_countermeasures_folder)
129 download(dst=countermeasures_filename, path='https://github.com/mandiant/red_team_tool_countermeasures/archive/refs/heads/master.zip')
130 unzip(filename=countermeasures_filename, dst=root)
131 shutil.copytree(src=yara_countermeasures_folder, dst=local_countermeasures_folder, dirs_exist_ok=True)
132 shutil.rmtree(countermeasures_folder)
133 os.remove(countermeasures_filename)
134
135 #rulesmaster
136 create(folder=local_rulesmaster_folder)
137 download(dst=rulesmaster_filename, path='https://github.com/Yara-Rules/rules/archive/refs/heads/master.zip')
138 unzip(filename=rulesmaster_filename, dst=root)
139 shutil.copytree(src=yara_rulesmaster_folder, dst=local_rulesmaster_folder, dirs_exist_ok=True)
140 shutil.rmtree(rulesmaster_folder)
141 os.remove(rulesmaster_filename)
142
143
144 #ThreatHunting
145 create(folder=local_ThreatHunting_folder)
146 download(dst=ThreatHunting_filename, path='(variable) root: str :ht/ThreatHunting-Keywords-yara-rules/archive/refs/heads/main.zip')
147 unzip(filename=ThreatHunting_filename, dst=root)
148 shutil.copytree(src=yara_ThreatHunting_folder, dst=local_ThreatHunting_folder, dirs_exist_ok=True)
149 shutil.rmtree(ThreatHunting_folder)
150 os.remove(ThreatHunting_filename)
151
152
```

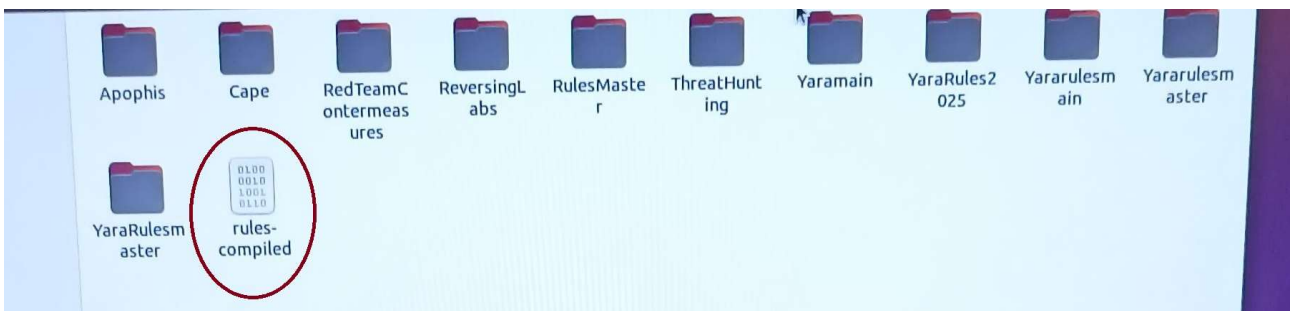

Compilar Reglas Yara

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)

```
keepcoding@ubuntu:~/Desktop$ python3.8 cape.py  
Traceback (most recent call last):
```



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.

The screenshot displays the AnyRun Public Submissions interface. The main area shows a list of submissions, each with a Windows logo, version (Windows 10 Professional 64 bit), date (02 May 2025, 19:22), a checkmark, a file icon, and a status (No threats detected). The right-hand panel contains a FILTER section with OBJECT (Hash, Runtype, Extension, Country) and VERDICT (Malicious, # Tag) dropdowns. Below the VERDICT section is a CONTEXT section with File hash, Domain, IP address, MITRE ATT&CK™ technique ID, and Suricata SID. At the bottom of the right-hand panel is a DATE section with a From dropdown. A sidebar on the left contains navigation options: New analysis, Reports (highlighted with a red box), Teamwork, History, TI, 10 32 bit, Profile, Notifications, and Pricing. The bottom of the main area shows a pagination bar with 1 OF 246643.

Windows 10 Professional 64 bit	02 May 2025, 19:22	✓	No threats detected	https://landingpage.premiummemberservices.com/
Windows 10 Professional 64 bit	02 May 2025, 19:22	✓	No threats detected	https://link.quorumtracking.com/f/a/TinAx1o2cajEuX2lguWY
Windows 10 Professional 64 bit	02 May 2025, 19:22	✓	No threats detected	https://download.cyberlearn.academy/download/download_f
Windows 10 Professional 64 bit	02 May 2025, 19:21	✓	No threats detected	https://33.172.153.160.host.secureserver.net/TYxXEBWH/Uc
Windows 10 Professional 64 bit	02 May 2025, 19:21	✓	No threats detected	https://weaver.dicoeur.top/0w/ow/auth/logon.aspx?replaceC
Windows 10 Professional 64 bit	02 May 2025, 19:21	✓	Malicious activity	BA46275B-3579-49BE-AF4D-3C5D34555A48.1_originalmail...

• 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.

```
python3.8: can't open file 'malware.py': [Errno 2] No such file or directory
keepcoding@ubuntu:~/Downloads/malware1$ python3.8 malware.py

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

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

A screenshot of a code editor window titled 'malware.py' with the path '~Downloads/malware1'. The code is as follows:

```
1 import yara
2 import os
3
4 # Ruta donde está el malware y las reglas YARA compiladas
5 MALWARE_DIR = os.path.expanduser("~/Downloads/malware1")
6 RULES_PATH = os.path.join(MALWARE_DIR, "rules-compiled")
7
8 def load_yara_rules():
9     try:
10         rules = yara.load(filepath=RULES_PATH)
11         print("[+] Reglas YARA cargadas correctamente.")
12         return rules
13     except Exception as e:
14         print(f"[-] Error cargando reglas YARA: {e}")
15         return None
16
17 def scan_malware(rules):
18     if not rules:
19         return
20
21     print(f"\n[+] Escaneando archivos en: {MALWARE_DIR}")
22
23     for filename in os.listdir(MALWARE_DIR):
24         file_path = os.path.join(MALWARE_DIR, filename)
25
26         # Ignorar el archivo de reglas compiladas
27         if filename == "rules-compiled":
28             continue
29
30         if os.path.isfile(file_path):
31             try:
32                 matches = rules.match(filepath=file_path)
33                 if matches:
34                     print(f"\n🔥 **Malware detectado en {filename}**")
35                     for match in matches:
36                         print(f"    - Regla: {match.rule}")
37                         if match.meta.get("description"):
38                             print(f"    - Descripción: {match.meta.get('description')}")
```

```
[+] 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.
repcoding@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