III Bootcamp Full Stack Cibersecurity

Módulo 7 - Malware Analysis

Caso Práctico

Reglas Yara

Marcos Alonso González

alonsogonzalezmarcos@gmail.com

https://github.com/magalorn

7 de mayo de 2022

Índice

Creación de script Python

Se ha creado un script con Python que reúne varios repositorios GitHub con ficheros que incluyen reglas Yara, y las compila en un solo archivo. Después, podremos hacer análisis con ficheros de malware para comprobar si matchea con alguna de las reglas incluidas en el archivo compilado.

Almacenado en el fichero `yara_compile.py` adjunto a este informe.

1. Preparación

Se describen a continuación los pasos a seguir para el proceso de matcheo de reglas contra archivos de malware:

1. En primer lugar, hay que asegurarse de utilizar una máquina virtual donde almacenaremos las muestras de malware a matchear. En este caso, se utiliza una maquina ubuntu 20.04.

2. Importar todos los módulos necesarios para el script, en este caso:

```
import yara
import os
import git
from git import Repo
import glob
import shutil
import fnmatch
```

2. Clonar y actualizar repositorios de reglas Yara

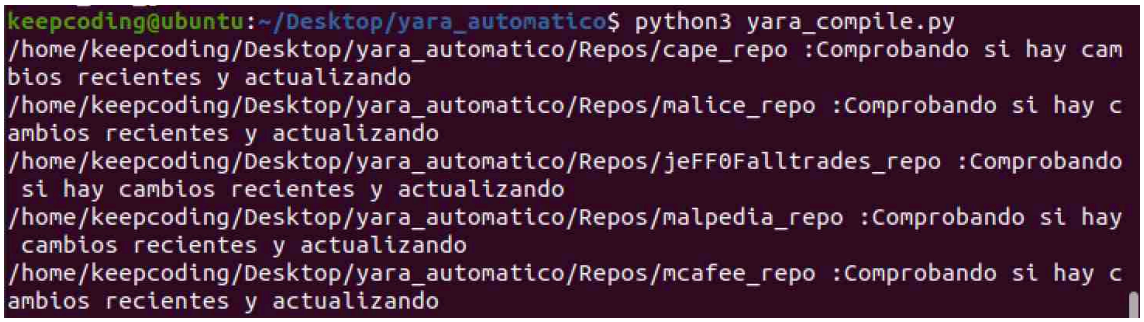
Los pasos que sigue el script para clonar y actualizar los repositorios de reglas son:

1. Seleccionar varios repositorios GitHub que contengan reglas Yara. En este caso se han seleccionado:

CAPE Rules: https://github.com/kevoreilly/CAPEv2.git
Malice.IO YARA Plugin Rules: https://github.com/malice-plugins/yara.git
jeFF0Falltrades Rules: https://github.com/jeFF0Falltrades/YARA-Signatures.git
Malpedia Auto Generated Rules Repo: https://github.com/malpedia/signator-rules.git
McAfee Advanced Threat Research Yara-Rules: https://github.com/advanced-threat-research/Yara-Rules.git

2. Actualizar o clonar repositorios de reglas Yara:

- Crear carpetas de los repositorios
- Clonar o actualizar repositorios: Si se incluye un nuevo repositorio, la función correspondiente lo clona y almacena en el directorio indicado. Si el repositorio ya está clonado, comprueba y actualiza, en su caso, el repositorio.



3. Se crea una carpeta donde almacenar todas las reglas Yara de todos los repositorios

4. Se copian todas las reglas Yara de todos los repositorios a la carpeta creada y convertir los ficheros con extension `.yara` en `.yar`

3. Compilar reglas

1. Seguidamente el script compila todas las reglas mediante una función en un solo fichero dentro de la carpeta donde se han almacenado previamente todos los ficheros `.yar`.

4. Matchear reglas Yara con muestras de malware

1. En un directorio, almacenamos las muestras de malware

2. Dentro del script, con el método `yara.match`, podemos recorrer todo el directorio de malware y matchear todas las muestras contra el archivo de reglas compiladas.

3. Si tenemos éxito, en la terminal se imprime la ruta del fichero y una tupla con las reglas yara con las que ha matcheado, es decir, aquellas reglas que han detectado la muestra como malware, tal como ocurre por ejemplo con Dharma.exe, el malware que hemos elegido para analizar.

```
/home/keepcoding/Desktop/yara_automatico/malware/Dharma.exe : [PEiD_00497_dUP_v2
_x_Patcher_____www_diablo2oo2_cjb_net_, PEiD_01070_Microsoft_Visual_C___6_0___8_
0_, PEiD_01091_Microsoft_Visual_C___8_, Contains_PE_File, maldoc_function_prolog
_signature, maldoc_structured_exception_handling, maldoc_find_kernel32_base_meth
od_1, maldoc_suspicious_strings, _dUP_v2x_Patcher__wwwdiablo2oo2cjbnet_]
/home/keepcoding/Desktop/yara_automatico/malware/RevengeRAT.exe : [PEiD_00014__N
ET_executable____Microsoft_, PEiD_00015__NET_executable_, PEiD_00497_dUP_v2_x_Pa
tcher_____www_diablo2oo2_cjb_net_, PEiD_01060_Microsoft_Visual_C____Basic__NET_,
 PEiD_01061_Microsoft_Visual_C__v7_0___Basic__NET_, PEiD_01128_Microsoft_Visual_
Studio__NET_, Contains_PE_File, maldoc_suspicious_strings, _dUP_v2x_Patcher__www
diablo2oo2cjbnet_, _NET_executable_, _Microsoft_Visual_C_v70__Basic_NET_, _Micro
soft_Visual_C__Basic_NET_]
```