ANALISIS DE MALWARE

En este informe explicare el proceso paso por paso de como compile mas de 25000 reglas yara en un solo binario, como obtuve mas de 140 muestras de malware y como ejecute el compilado yara encontrando coincidencias con las muestras malware.

1) Obtención de Reglas Yara

El primer paso fue realizar una investigación para recopilar la mayor cantidad de reglas yara clonando repositorios y abarcando todas las fuentes que provean reglas de detección y clasificación de malware. Mi objetivo era alcanzar el mayor grado de probabilidad en detecciones

Repositorios GIT y Webs

- 0x101 Cyber Security
- Adlice
- AlienVault
- Avast
- BAE Systems
- Bayshore Networks, Inc.
- Binalyze
- BinaryAlert
- Blueliv
- Cisco Talos Intelligence Group
- Cloudina Security
- Cofense
- Conix
- CounterCraft
- Cuckoo Sandbox
- Cyber Triage
- Cybereason
- Digita Security
- Dragos Platform
- Dtex Systems
- ESET

- ESTsecurity
- Elastic Security
- Fidelis XPS
- FireEye, Inc.
- Forcepoint
- Fox-IT
- FSF
- Guidance Software
- Heroku
- Hornetsecurity
- ICS Defense
- InQuest
- Joe Security
- Kaspersky Lab
- KnowBe4
- Koodous
- Laika BOSS
- Lastline, Inc.
- libguestfs
- LimaCharlie
- Malpedia
- Malwation
- McAfee Advanced Threat Defense
- Metaflows
- NBS System
- ndaal
- NetLock
- Nextron Systems
- Nozomi Networks
- osquery
- Payload Security
- PhishMe
- Picus Security
- Radare2
- RedSocks Security
- ReversingLabs
- Scanii
- SecondWrite
- SonicWall
- SpamStopsHere
- Spyre
- stoQ

- SumoLogic
- Tanium
- Tenable Network Security
- The DigiTrust Group
- ThreatConnect
- ThreatStream, Inc.
- Thug
- Threat.Zone
- TouchWeb
- Trend Micro
- UnpacMe
- Uptycs Inc
- VirusTotal Intelligence
- VMRay
- Volexity
- We Watch Your Website
- x64dbg
- YALIH
- /opt/CAPEv2/data
- YARA RULES

Lamentablemente en el siguiente paso cometí el error de borrar los repositorios clonado con las carpetas originales, únicamente quedaron los ficheros .yar y .yara.

2) Filtración de ficheros .yar y .yara

Descubrí que ocupe 40 gigabytes entre todos los directorios que recopile, por lo que debía filtrar únicamente las reglas yara.

Entendí que no podría hacer el trabajo que propuse en tiempo y forma, por lo que decidí crear varios códigos Python para automatizar el proceso completo. Estos códigos estan almacenados en la carpeta “Codigos”

El primer código Python que utilice es “copiar-archivos.py” que iteraba dentro de las carpetas buscando únicamente los archivos de tipo .yar y .yara, luego los copiaba y pegaba en otro directorio.

De esta forma logre filtrar lo que necesitaba de lo que me era innecesario

3) Depurar Directorio de Reglas

En el proceso encontre reglas que contenían errores, por lo que construi el código “extraer-utiles.py” que compila reglas una a una, copia y pega las reglas sin errores en otro directorio y elimina los compilados resultantes.

Así seleccione las reglas efectivas que permitían compilarse en un solo binario.

4) Compilación

Al momento tenia mas de 40 gigabytes en reglas yara listas para compilar en un solo directorio, debía crear un script que lograra compilar todas al mismo tiempo. Busque en el repositorio GIT de Keepcoding en la sección Malware provisto por el profesor Adrian de Análisis de Malware un código de compilación.

Adapte el compilador Python a mis necesidades y logre compilar todas las reglas en un binario llamado “rules_compiled”

5) Adquisición de Muestras de Malware

Investigando para obtener muestras de Malware encontré un repositorio creado por una comunidad de Analistas de Malware, su nombre es “TheZoo”. En el repositorio hay una base de datos de numerosas muestras de virus en archivos .zip.

Clone el repositorio https://github.com/ytisf/theZoo

6) Obtención de Binarios

Dentro del repositorio “TheZoo” se encontraban 640 elementos, la mayoría ficheros .zip con clave “infected”. Por lo que cree el código Python “descomprimir.py” que itera sobre todos los directorios y subdirectorios de la carpeta, y descomprime en otra carpeta los .zip con la clave “infected”

7) Discriminación de muestras

Los archivos descomprimidos aparecían en diferentes formatos, difícil era detectar cuales eran las muestras. Por lo que decidí crear un código Python que discrimine los binarios de malware de lo innecesario. El código “importando-muestras.py” itero en cada elemento del directorio y ejecuto el binario en busca de que haya coincidencias o no, si daba error pasaba por alto el archivo, sino copiaba el archivo y lo pegaba en otro directorio donde estarán las muestras finales listas para analizar.

8) Compilado Yara vs Muestras

El paso final fue crear un código Python que buscara coincidencias con cada muestra iterando en la carpeta donde estas se encontraban. Si encuentra coincidencias, guardara en el directorio “Matches” un fichero .txt con el nombre de la muestra y dentro los resultados de la comparación. En caso contrario guardara en el directorio “Non-Matches” un fichero .txt con el nombre de la muestra.

El resultado fue que las 25000 reglas yara compiladas encontraron coincidencias con el 100% de las muestras de malware