Universidad Zaragoza
1542

Trabajo Fin de Grado

SISTEMA AUTOMATIZADO DE DETECCIÓN DE *MALWARE* CON LA *SANDBOX CAPE*

CAPE AUTOMATED MALWARE DETECTION SYSTEM

Autor

Mario Romeo Lázaro

Director

Álvaro Alesanco Iglesias

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2023

AGRADECIMIENTOS

En primer lugar, quiero agradecer a mi familia y a mis amigos todo el apoyo, paciencia y confianza que han tenido a lo largo de estos años de carrera.

También, quiero agradecer en especial a todas las personas que han contribuido y apoyado en la realización de este proyecto. Tanto a mi tutor, Álvaro, por aconsejarme en el enfoque del proyecto, en momentos de colapso, como a Lorenzo y Andriy por responder a mis preguntas sobre *CAPEv2*.

Además, quiero agradecer a toda la gente que he conocido a raíz de la carrera, la compañía en las buenas y, sobre todo, en las malas.

Por último, agradezco a todas las personas que, para bien o para mal, han influido en mi manera de ser hasta hoy. Ya que gracias a ellos, soy capaz de intentar, con todas mis energías, todo lo que me propongo.

Lista de acrónimos

- CEO: *Chief Executive Officer*
- CAAS: *Cybercrime as a Service*
- Anti-VM: *Anti-Virtual Machine*
- CAPE: *Configuration And Payload Extraction*
- LTS: *Long-Term Support*
- MAC: *Media Access Control*
- IDS: *Intrusion Detection System*
- DLL: *Dynamic Link Library*
- KVM: *Kernel-based Virtual Machine*
- API: *Application Programming Interface*
- VNC: *Virtual Network Computing*
- VPN: *Virtual Private Network*
- TOR: *The Onion Router*
- SFTP: *Secure File Transfer Protocol*
- IP: *Internet Protocol*
- IOC: *Indicator of Compromise*
- MD5: *Message Digest Algorithm 5*
- CMD: *Command Prompt*
- WEB: *World Wide Web*
- URL: *Uniform Resource Locator*
- HTML: *HyperText Markup Language*
- TCP: *Transmission Control Protocol*
- HTTP: *HyperText Transfer Protocol*

Lista de Figuras

Sistema automatizado de detección de malware con la *sandbox CAPE*

RESUMEN

El proyecto comienza con la investigación de los fundamentos en el ámbito de la seguridad en redes y sistemas. Se realiza un análisis detallado de los conceptos y principios clave del *malware* y de su análisis, con el objetivo de comprender mejor las amenazas existentes y las posibles soluciones. Además, se lleva a cabo un estudio de la situación actual en este campo, examinando las tendencias y los avances tecnológicos relevantes.

Tras este estudio, se procede a la selección y puesta en marcha conjunta de diferentes herramientas especializadas, para crear un entorno de *sandboxing*. Este entorno proporciona la capacidad de analizar y detectar posibles amenazas, en los ficheros descargados por los sistemas de una red, sin comprometer la integridad de los sistemas y la red en su conjunto.

Finalmente, se llevan a cabo pruebas y se evalúan los resultados obtenidos en el entorno desarrollado. Estas pruebas permiten evaluar la eficacia y el rendimiento de las herramientas seleccionadas, para detectar y prevenir ataques cibernéticos.

CAPE Automated Malware Detection System

ABSTRACT

The project begins by exploring the foundations of network and system security. A thorough analysis of key concepts and principles related to malware and its analysis is conducted to gain a deeper understanding of existing threats and potential solutions. Additionally, the current landscape in this field is examined, taking into account relevant trends and technological advancements.

Based on this research, a selection of specialized tools is implemented to create a sandboxing environment. This environment empowers the analysis and detection of potential threats within files downloaded by network systems, ensuring the integrity of both the systems and the network as a whole.

Finally, testing is carried out to evaluate the effectiveness and performance of the chosen tools in detecting and preventing cyberattacks. These tests provide valuable insights into the capabilities and limitations of the selected tools in real-world scenarios.

Índice

Capítulo 1

Introducción

1.1. Motivación

Durante las últimas décadas, debido al exponencial avance que experimenta la tecnología, un porcentaje de las actividades delictivas han sufrido, parcialmente, una transición del plano físico al plano virtual. El aumento de las cifras tiene varias justificaciones, entre las cuales se encuentra que ya no es necesario ser un experto en sistemas, para llevar a cabo un ciberataque. Dado que, desde hace unos años, las organizaciones criminales, ofrecen lo que se denomina como *Cybercrime-As-A-Service (CAAS)* [4], provocando que la realización de ciberdelitos esté al alcance de una gran parte de la población. Esto se ve reflejado, por ejemplo, en el aumento de intentos de intrusión en sistemas informáticos que, a pesar de que 2021, fue el año con más ciberataques registrados en la historia, según los datos proporcionados por la empresa *SonicWall* en su reporte anual [5], han aumentado en un 19 % respecto a 2021, sumando un total de 5500 millones de intentos.

La transición del crimen a un plano virtual, también se puede observar fácilmente en los nuevos métodos utilizados para atacar a entidades, que trabajan a diario con grandes cantidades de dinero o con información crítica, como pueden ser las empresas o las instituciones públicas. Por ejemplo, durante la pandemia causada por la *COVID-19*, se puso de manifiesto, en el sector sanitario, la gravedad que podía llegar a tener el secuestro de ordenadores en los hospitales [6]. En lo referido al sector privado, son comunes las estafas mediante la suplantación de identidad, en las que el atacante simula ser el *CEO* de una compañía e indica al personal que está a su cargo la realización de transferencias a cuentas [7] ubicadas en paraísos fiscales o en países en los que este tipo de delitos no se persiguen con la rigurosidad pertinente.

Esta transición genera nuevas necesidades, como pueden ser formar especialistas y crear soluciones capaces de combatir las amenazas que surgen a raíz del progreso tecnológico.

Una de estas amenazas es el código malicioso, también conocido en el ámbito de la ciberseguridad como *malware*. Siendo este el encargado de infectar los sistemas, habitualmente a través de un conjunto de instrucciones, que suelen estar enmascaradas sobre un ejecutable o un documento con apariencia no sospechosa, que aprovechan vulnerabilidades del software del sistema para comenzar su infección. Como se comentará en la sección 2.1 hay distintos tipos de *malware*, aumentando así la dificultad a la hora de abordar el problema de su detección y análisis. En el capítulo 2 se expanden los principales conceptos relativos al *malware*.

Este proyecto está centrado en la automatización de la detección y el análisis de código malicioso, pero para plantear el problema a resolver, como se explicará en la sección 4.1, hay que tener una visión global de la cuestión que se aborda. Por este motivo, se introducen los conceptos básicos del estado actual del *malware* y se desarrollan las características del entorno de trabajo que se va a emplear en el proyecto.

1.2. Objetivos

El objetivo principal de este proyecto es el diseño e implementación de un sistema de *sandboxing*, para minimizar el riesgo de infección en una red. Esta solución se va a abordar mediante la automatización de una serie de herramientas, que puestas en marcha de manera conjunta pueden proporcionar un alto grado de control sobre los ficheros que entran en los equipos de una red.

Para realizar esta automatización hay que enfrentarse a varios objetivos secundarios, que se explican a continuación:

- Selección del *software* que se va a utilizar, en el cual se profundiza en la sección 3.1.
- Instalación y configuración de los programas con los que se va a trabajar.
- Diseñar y establecer la comunicación que estos programas van a tener y en que equipos van a estar situados.
- Diseñar cómo se va a realizar en una red, de manera automática, la monitorización de los nuevos ficheros en los equipos.
- Implementar y probar la automatización diseñada. Obteniendo los resultados que se amplían en el capítulo 5 con muestras reales.

El *software* seleccionado para la automatización diseñada es el siguiente:

- Herramienta de *malware sandboxing open-source CAPEv2*.

- Conjunto de programas *Elasticsearch, Kibana, Logstash y Filebeat*.
- *API* de *VirusTotal*.
- *Proxy-web NGINX* y *Guacamole*, utilizados para habilitar cierta interactividad durante los análsis.
- Sistema operativo de *Mikrotik*.

Capítulo 2

Detección y análisis de *Malware*

Para la comprensión del proyecto es necesario introducir los principales conceptos relacionados con el análisis, la ejecución y la detección del *malware*. En las siguientes secciones se tratan estos aspectos, sin profundizar en exceso, con el ánimo de proporcionar una visión global.

2.1. Categorías más comunes de *malware*

En primer lugar, es importante recalcar que la entrada en el sistema de los códigos maliciosos se suele realizar a través de puntas de lanza, como pueden ser portales *web*, unidades extraíbles, correo electrónico y otros medios telemáticos que usamos en nuestro día a día. Posicionando, de esta manera, al control sobre los ficheros de entrada en un sistema, como una actividad fundamental, a la hora de garantizar cierta seguridad en una red.

Como se ha mencionado en la introducción, los códigos maliciosos no actúan uniformemente. Esto se debe a varias razones, entre las cuales se encuentran las fases mostradas en la figura 2.1, el objetivo y la intención de cada ciberataque.

Por ejemplo, a la hora de plantear un ataque, no se orienta igual la recopilación de información, que la exfiltración de datos. Puesto que, la recopilación de información suele requerir de técnicas pasivas, como puede ser un *Keylogger* o un *Spyware*, mientras que la exfiltración de datos requiere de acciones activas, como podrían ser la instalación y ejecución de un troyano.

A continuación se describen de manera breve las principales categorías de código malicioso cuyos análisis se muestran en el capítulo 5. Hay que tener en cuenta que, como se ve en los resultados, es común que una muestra de *malware* pertenezca a varias categorías.

Figura 2.1: Etapas de un *ciber-ataque* [1].

2.1.1. Virus

Los virus informáticos están diseñados para replicarse a sí mismos y propagarse a otros equipos. Los virus se propagan al insertar su código en archivos legítimos y engañar al usuario para que los ejecute. Pueden causar daños graves al sistema, como la eliminación de archivos, el robo de información y la corrupción del sistema operativo.

2.1.2. *Ransomware*

El *ransomware* es un tipo de *malware* que restringe el acceso del usuario a su sistema o a sus archivos y exige un rescate a cambio de restaurar el acceso a este. Este tipo de *malware* puede cifrar archivos o bloquear completamente el acceso al equipo, y como veremos en la sección 5.3.1.1 suelen exigir el pago del rescate con una cantidad de dinero en *criptomonedas*.

2.1.3. Gusano

Un gusano, también conocido como *worm*, se propaga a través de las redes y de los sistemas conectados al equipo en el que se ubica. A diferencia de los virus, los gusanos no necesitan infectar archivos legítimos para propagarse, sino que se replican y buscan vulnerabilidades en otros sistemas para infectarlos. Los gusanos pueden ralentizar y corromper los datos de los equipos afectados.

2.1.4. Troyano

Los troyanos suelen tener la apariencia de un programa útil o de un juego y se instalan en el sistema sin el conocimiento del usuario. Una vez instalados, pueden robar información confidencial, controlar el sistema o abrir puertas traseras para permitir un acceso no autorizado al sistema infectado.

2.2. Análisis de *malware*

A la hora de analizar muestras correspondientes a código malicioso existen varias técnicas que aportan diferentes enfoques al análisis. El análisis de *malware* se puede clasificar principalmente en dos categorías: análisis estático y análisis dinámico. Estas categorías se diferencian en que mientras el análisis estático se dedica a examinar el código malicioso sin ejecutarlo, el análisis dinámico ejecuta las muestras en un entorno controlado.

A continuación se resumen las principales características de los métodos más utilizados en este ámbito.

2.2.1. Desensambladores

Los desensambladores son herramientas, que permiten traducir el código binario de un programa en un lenguaje ensamblador, a un código interpretable para su análisis. Esto es útil para entender el comportamiento del código malicioso y analizar su estructura. Los desensambladores tratan de mostrar la lógica subyacente de la muestra que analizan.

2.2.2. Depuradores

Los depuradores o *debuggers* son herramientas que permiten ejecutar el código de un programa paso a paso, lo que habilita la detección de errores y vulnerabilidades. Los depuradores también pueden detener la ejecución del programa en un punto específico y examinar el estado de la memoria y los registros del sistema. Esto es útil para analizar el comportamiento del código malicioso y encontrar posibles puntos de entrada.

2.2.3. *Sandboxing*

La técnica de *sandboxing* consiste en llevar a cabo la ejecución de muestras de código malicioso en un entorno virtual, conocido como *sandbox*.

El término *sandbox* tiene múltiples definiciones según diversos investigadores. Sin embargo, según el artículo científico, *A systematic analysis of the science of sandboxing*,

la definición del concepto *sandbox* en términos de aislamiento se puede unificar como "un mecanismo de encapsulación que se utiliza para imponer una política de seguridad en los componentes de software"[8].

En lo que se refiere a la *sandbox* en términos de análisis de *malware*, se puede definir como "un entorno de redes y computadores aislado construido para analizar el comportamiento de muestras de *software*"[9].

Existen diferentes herramientas de *sandboxing*, tanto de pago como puede ser la aplicación *ANY.RUN* [10], como de código libre, que es el caso de la herramienta que se usa en este proyecto y que se detalla en la sección 3.1.1 [2].

2.3. Técnicas *ANTI-VM*

Ahora que se tiene una visión general, del significado del código dañino, se van a explicar los principales conceptos relacionados con el *ANTI-VM*.

La importancia de este tema reside en que, desde hace años, los autores de *malware* han comenzado a diseñar código malicioso, capaz de detectar si está siendo ejecutado en un entorno virtual. En tal caso, este ajusta su comportamiento, generalmente evitando ejecutar actividades maliciosas o ralentizando su ejecución [11]. En los últimos años, se ha observado que la principal tendencia, en el desarrollo del *malware*, es centrarse únicamente en la detección de un entorno *sandbox*, en lugar de un entorno virtualizado genérico. Esta es una consecuencia de la popularidad de los sistemas de producción, que se ejecutan en entornos virtualizados en la nube [12].

Un aspecto que el *malware* comprueba es la presencia de ciertos artefactos, que deja en el sistema operativo de sus *guest* el *host* que los aloja. Se implantan en los *guest* con el fin de facilitar la gestión y comunicación, con el administrador de la máquina virtual. Estos artefactos incluyen, entre otros, procesos, claves y valores del registro, *DLL* cargadas y exportadas, características relacionadas con la red (por ejemplo, direcciones MAC específicas y nombres de adaptadores), trazas reconocibles en archivos (por ejemplo, *system32\vboxtray.exe*), directorios propios (por ejemplo, *%PROGRAMFILES% VMWare*) y etiquetas de hardware [13].

Las medidas aplicadas en este proyecto para contrarrestar algunas de las técnicas explicadas se detallan en la sección 4.3.

Capítulo 3

Entorno de trabajo

3.1. Herramientas utilizadas

3.1.1. *CAPEv2*

La herramienta de *sandboxing, CAPEv2,* está instalada y configurada, tal y como se explica en el anexo A. Esta herramienta trabaja con la arquitectura de la figura 3.1 y evalúa la muestra seleccionada en varias fases: análisis, procesado y reporte de resultados. Funciona a través del despliegue y comunicación de los 4 servicios que se explican a continuación:

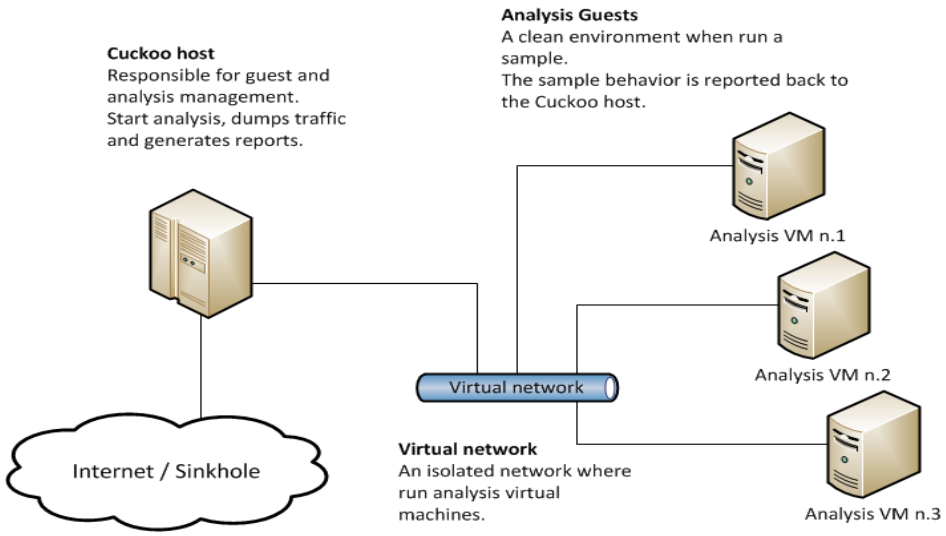

Figura 3.1: Arquitectura de la *sandbox CAPEv2* [2].

- *cape.service*: Es el servicio encargado de enviar la muestra a analizar al *guest*, con las opciones que hayan sido configuradas, de manera adecuada. Cuando se indica la realización de un análisis, este servicio es el encargado de indicar al servicio *cape-rooter* qué opciones han sido seleccionadas para la configuración de red. Tras esto pone el *guest* en marcha, con el *snapshot* que se haya configurado y

le manda al agente que está instalado en el *guest* los *scripts* que se deben ejecutar en el análisis. Mientras se realiza un análisis, este servicio mantiene de manera constante la comunicación con el agente instalado en el *guest*, recopilando de este modo información a cerca de lo que está sucediendo durante el análisis. Tras finalizar el tiempo de análisis, se encarga de apagar el *guest*, indicar al servicio *cape-rooter* que desactive las opciones activadas y de especificarle al servicio *cape-processor* que debe iniciar el procesado de los resultados del análisis.
En resumen, es el servicio organizador, ya que es el que define cuando empiezan y acaban las fases en la *sandbox* para una muestra.

- *cape-processor.service*: Es el servicio encargado de dar un formato interpretable a la información extraída, tras el análisis de una muestra. Este servicio gestiona dos etapas de manera secuencial:

  - Procesado de datos: Procesa los datos con los módulos que tenga configurados, por ejemplo, aplicando scripts de detección de *malware* mediante la aplicación de reglas *YARA*. Otro ejemplo de procesado es el realizado por el módulo *Suricata*, que es un sistema de detección de intrusos en red, que analiza el tráfico en busca de patrones maliciosos y comportamientos sospechosos. Este módulo se ha configurado para que los reportes sean almacenados en un archivo llamado *eve.json*. Además, se pueden añadir módulos personalizados.
  - Reporte de datos: En esta etapa se realiza un reporte de los datos procesados, con los módulos que tenga configurados. En el capítulo 4.5 se explica cómo se ha creado un módulo de reporte propio, cuyo código está disponible en el anexo D.

- *cape-rooter.service*: La herramienta *cape-rooter* es la encargada de enrutar el tráfico saliente del *guest*, tal y como se haya especificado en la configuración del análisis a realizar, mediante la modificación de las *iptables* del *host*. Dando las siguientes posibilidades de enrutamiento:

  - Directamente a internet.
  - A través de una *VPN*.
  - Bloquear la conexión a internet, descartando los paquetes.
  - A través de *TOR*.
  - Usando simuladores de conexión, como *InetSim*.
  - Mediante *sockets* personalizados.

- *cape-web.service*: Es el servicio que permite indicar las opciones de un análisis y la visualización de los datos generados por los módulos de reporte, mediante una interfaz web de manera gráfica. Utiliza la interfaz *localhost* y el puerto 8000. A su vez, es el encargado de comunicarse con el servicio *guac-web*, en los análisis en los que está habilitada la interacción.

En este proyecto se ha configurado *CAPEv2* para habilitar un análisis interactivo, esto quiere decir que mientras se realiza el análisis, desde la interfaz *web*, se puede interactuar con el *guest*, gracias al software detallado en las secciones 3.1.5 y 3.1.4. También existe la posibilidad de realizar un análisis en el que la interacción humana sea simulada, en este caso la aplicación *CAPEv2* lanzaría en el *guest* un *script*, que han creado los desarrolladores de este *software*, llamado *human.py*. En este proyecto se ha adaptado este *script* para que interactúe con software en castellano. Este *script* está disponible en el anexo D. También se han configurado dos *guest*, uno con sistema operativo *Windows 7* y otro con *Windows 10*.

En lo relativo al nivel de red se ha trabajado direccionando el tráfico por una *VPN* en una ubicación externa a la universidad. Esta decisión ha sido tomada para evitar cualquier limitación que pudiera imponer, a la hora de realizar las pruebas con *malware* real, el *firewall* de la universidad. La configuración de esta *VPN* se ha realizado a través de una máquina virtual, con el sistema operativo de *Mikrotik*, ubicada en el domicilio del alumno, cuya configuración está detallada en el anexo B.

Todos los ficheros y procesos de configuración se engloban en el anexo A.

3.1.2. *Elastic*

A continuación se explica de manera breve la función y ubicación de cada componente del *stack* de *Elastic* [14].

- *Elasticsearch*: Está instalado en el equipo *Debian 11* y es el componente encargado de gestionar los índices en los que se almacena la información. Tanto el mapeado, como las políticas, que se han aplicado a los índices utilizados para este proyecto, se detallan en el anexo E.

- *Logstash*: Está instalado en el equipo *Debian 11*, siendo el responsable de procesar y filtrar los *logs* que se envían desde el equipo *Ubuntu Server*. Las opciones y los filtros aplicados para este programa se encuentran en el anexo E.

- *Kibana*: Está instalado en el equipo *Debian 11*, se requiere para crear y mostrar los formatos de visualización que se crean pertinentes, para los datos previamente procesados por *Logstash* y almacenados en un índice configurado de *Elasticsearch*.

Las opciones y los *dashboards* creados están detallados en el anexo E y en el capítulo 5, respectivamente.

- *Filebeat*: Es el agente instalado en el equipo *Ubuntu Server*, que escucha en una carpeta, especificada en su configuración, que apunta al último análisis realizado, donde se generan los *logs* por el módulo de reporte creado, que se detalla en la sección 4.5. Los ficheros de configuración modificados para este programa se especifican en el anexo E.

3.1.3. *VirusTotal*

VirusTotal es un servicio en línea gratuito que permite analizar archivos y *URLs* en busca de malware y otras amenazas de seguridad. Utiliza múltiples motores antivirus y herramientas de análisis para escanear los archivos y proporciona un informe detallado de los resultados.

En la solución detallada en el capítulo 4.4, para realizar un análisis preliminar de las muestras, se hace uso de la versión 3 de la API de *VirusTotal*. Mediante las llamadas a esta API, se recibe un archivo *JSON*, que contiene información relativa al riesgo que tiene el objeto enviado a analizar. [15]

3.1.4. *Guacamole*

Guacamole [16] es la utilidad que permite realizar de manera segura y visual un análisis interactivo sobre las muestras, que se analizan en los *guest*. Su arquitectura se muestra en la figura 3.2, está compuesta por dos servicios:

- *guacd.service*: Es el servicio encargado de establecer con el *guest* la conexión *VNC,* para habilitar la interacción en tiempo real, mientras se realiza el análisis.

- *guac-web.service*: Es el servicio encargado de dar una interfaz gráfica a la interacción con el *guest*, a través del servicio *web*. Utiliza la interfaz *localhost* y el puerto 8008.

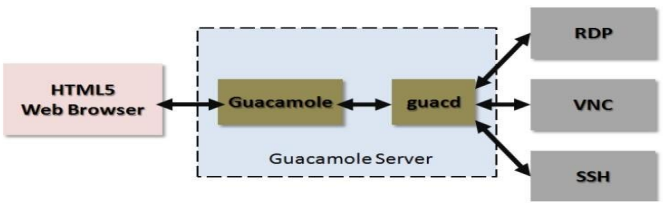

Figura 3.2: Arquitectura de *Guacamole* [3].

3.1.5. *NGINX*

Esta aplicación [17] es el *proxy-web* que se va a utilizar para unificar en un puerto y una dirección *IPv4* los servicios *cape-web* y *guac-web*. Esto se lleva a cabo, para que, según la ruta a la que se acceda en la dirección del *proxy*, se redireccione tal y como se muestra en la figura 3.3.

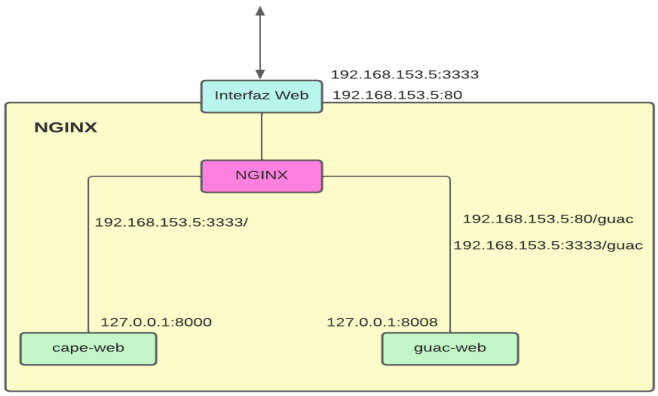

Figura 3.3: Esquema utilizado con el *proxy NGINX*.

3.2. *Software* de virtualización

3.2.1. *KVM*

A la hora de gestionar las máquinas virtuales se emplea el *software* de virtualización *KVM,* que viene integrado en los sistemas *Linux*. Se escoge este hipervisor por varias razones:

- Es el hipervisor más compatible con *CAPEv2*, según su documentación oficial. [2]

- Las técnicas *ANTI-VM* se pueden paliar de manera más efectiva con *KVM* que con otros hipervisores. Esto se debe a que se pueden modificar las trazas típicas, que suelen comprobar las muestras de *malware*, de una forma más creíble que otros hipervisores.

- Es un *software open-source*, esto ahorra la necesidad de destinar recursos económicos para su uso.

3.3. Escenario de trabajo

El entorno en el que se van a desplegar todas las herramientas de *software* utilizadas es el que se muestra en la figura 3.4.

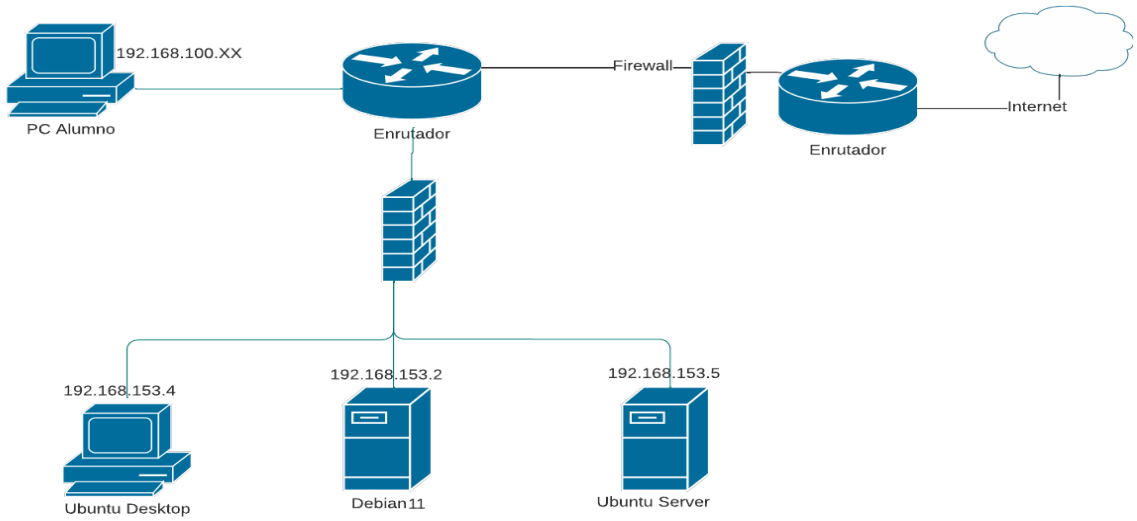

Figura 3.4: Escenario de red principal.

3.3.1. *Ubuntu Server*

La máquina *Ubuntu Server* trabaja con el sistema operativo *Ubuntu 22.04.02 LTS*. Cuenta con una memoria RAM de 32 GB. El procesador utilizado es el *Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz*, con 8 *cores* habilitados. Además, dispone de un espacio de almacenamiento de 125 GB.

En este equipo se instalan y configuran los principales servicios de este proyecto, en lo que se refiere al análisis de muestras. Para comprender la función de cada utilidad usada en este equipo, se ilustra un esquema en la figura 3.5. El diagrama aplica una lógica, en la que las secciones resaltadas en azul son funcionalidades que interactúan con equipos externos y el software remarcado con morado son servicios, que interactúan como intermediarios para realizar funciones, que no implementa la aplicación fundamental, resaltada en verde.

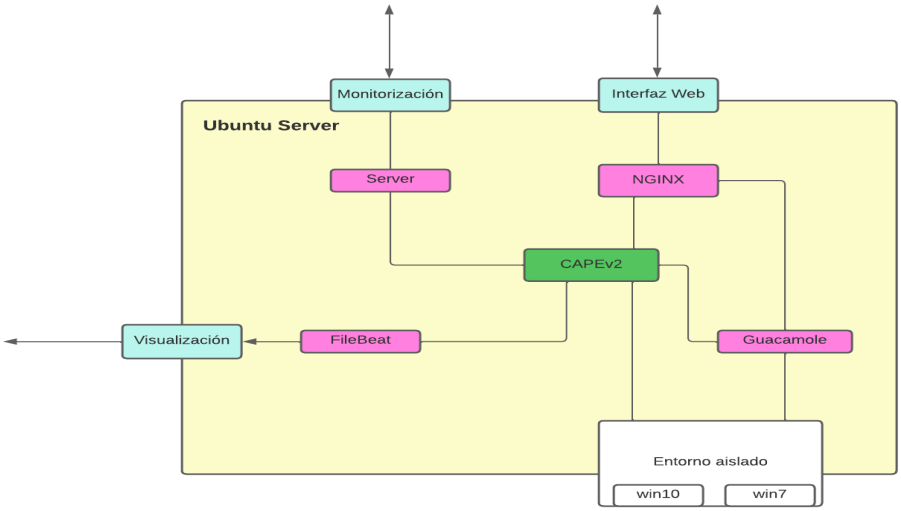

Figura 3.5: Esquema del equipo *Ubuntu Server*.

3.3.2. *Ubuntu Desktop*

La máquina *Ubuntu Desktop* trabaja con el sistema operativo *Ubuntu 22.04.01 LTS*. Opera con una memoria RAM de 16 GB. El procesador utilizado es el *Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz*, con 4 *cores* habilitados. Además, dispone de un espacio de almacenamiento de 125 GB.

Este equipo ha tenido principalmente dos funciones en el desarrollo del proyecto. La primera de estas, ha sido permitir la configuración, de manera gráfica, de las máquinas virtuales alojadas en el *Ubuntu Server*, para aplicar las configuraciones detalladas en el anexo A. Por otro lado, esta máquina también se ha empleado para programar y probar el agente de monitorización que se detalla en la sección 4.4.

3.3.3. *Debian 11*

La máquina *Ubuntu Desktop* utiliza el sistema operativo *Ubuntu 22.04.01 LTS*. Trabaja con una memoria RAM de 8 GB. El procesador utilizado es el *Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz*, con 4 *cores* habilitados. Además, dispone de un espacio de almacenamiento de 100 GB.

La principal función de este equipo es filtrar, procesar y visualizar, de manera compacta y precisa, los datos que son generados por el equipo *Ubuntu Server*. La jerarquía por la que se rigen los programas en este equipo se muestra en la figura 3.6.

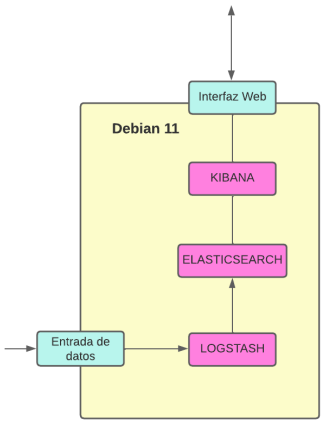

Figura 3.6: Esquema del equipo *Debian 11*.

Capítulo 4

Diseño de la solución

4.1. Enfoque del problema a resolver

El problema global a resolver se enfoca de la siguiente manera. Tenemos una red con varios equipos en los que, para minimizar el riesgo de infección, se cree necesario realizar una monitorización sobre los ficheros que se descargan en estos. Para esta monitorización se plantea la implementación de un seguimiento cíclico, constituido por las siguientes fases:

- Escucha en los equipos.
- Análisis en el equipo *Ubuntu Server*.
- Muestra de resultados de interés, garantizando un control centralizado de todos los análisis.

4.2. Programación en *Python*

La solución planteada en la sección 4.4 se ha programado en *Python*. La principal razón de esta decisión ha sido la gran variedad de bibliotecas que se proporciona en este lenguaje. También ha influido que el 92.1 % del programa *CAPEv2* haya sido desarrollado en este lenguaje, causando así un desarrollo de cierta soltura y comprensión en el estudiante a la hora de manejar este lenguaje.

4.3. Medidas *ANTI-VM* aplicadas en los *guest*

Adicionalmente, a las medidas que aplica *CAPEv2* en sus análisis, se han añadido una serie de precauciones en las máquinas virtuales configuradas para los análisis. Todos los archivos relacionados con estas se pueden ver explícitamente en el anexo A. A continuación se explican estas:

- Se ha modificado la dirección *MAC* de las máquinas virtuales.

- En el archivo *.xml* de los *guest* se han realizado las siguientes modificaciones, para acercar las características cada *guest* a las de una máquina real [18]:

    - Se ha modificado el modelo de *CPU* a uno seleccionado de una lista de *CPU*, compatibles con la que usa el *host*.

    - Se ha modificado la información del fabricante del producto que se muestra en la *BIOS*.

    - Se han añadido en el archivo *.xml* de cada *guest* una serie de comandos que modifican características que *KVM* añade por defecto en algunos registros y que son fácilmente comprobables.

- Se han aceptado *cookies* de diferentes sitios *web* y se han descargado diferentes tipos de ficheros en cada *guest*, para simular el estado de una máquina real.

4.4. Solución planteada

La solución planteada para cada fase ha sido la siguiente:

- Escucha en los equipos: Para garantizar la monitorización, de los directorios deseados, de un equipo específico, se han diseñado y programado una serie de funciones en *Python*. Estas funciones se utilizan en conjunto por un agente, también programado en *Python*. El agente se encarga de acceder y recopilar información sobre la ubicación especificada, del equipo objetivo.

- Análisis en el equipo *Ubuntu Server*: En el equipo *Ubuntu Server*, se ha desarrollado un servidor, programado para recibir de manera segura las solicitudes de análisis, provenientes de los agentes instalados en los equipos de la red. El servidor está configurado para establecer una comunicación cifrada e integra, garantizando así la seguridad de los datos transmitidos. Una vez que se reciben las solicitudes de análisis, el servidor procesa la información enviada por los agentes y realiza las tareas de análisis correspondientes.

- Muestra de resultados de interés: Utilizando la herramienta *Kibana* en el equipo *Debian 11*, se genera un panel de control, dónde se muestran los datos básicos relacionados con las conexiones de red, realizadas durante el análisis de la muestra indicada por el agente, en los equipos monitorizados. *Kibana* permite visualizar

y analizar de manera intuitiva los datos recopilados durante el proceso de monitoreo. Se presentan gráficos, tablas y otros elementos visuales que ayudan a comprender y evaluar las conexiones de red detectadas durante el análisis.

4.4.1. Flujo de trabajo

La comunicación implementada entre agentes y servidor garantiza una comunicación integra y confidencial. La confidencialidad de esta comunicación se garantiza generando un par de claves (pública y privada), tanto en el agente cómo en el servidor. Tras la generación de estas claves, los dos extremos de la comunicación comparten, mutuamente, sus claves públicas para que ambos sean capaces de mantener una comunicación cifrada. Tras este intercambio de claves y de manera cifrada, el agente le envía al servidor el *hash* del archivo que le va a enviar, para que este pueda verificar posteriormente su integridad. A continuación, el servidor envía al agente, de manera cifrada, un usuario y una contraseña con el formato *usuario:contraseña*. Finalmente, con estas credenciales, el agente transfiere al servidor, usando el protocolo *SFPT*, el fichero nuevo que ha detectado. Toda esta comunicación se representa en la figura 4.1. En las siguientes secciones se ve en profundidad la manera de trabajar de cada extremo de esta comunicación.

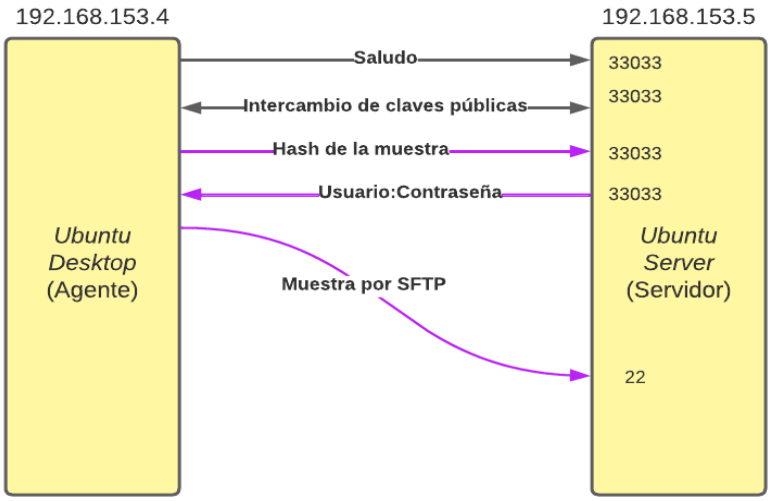

Figura 4.1: Representación de la comunicación realizada entre agente y servidor.

4.4.2. Servidor

En la sección 4.4.1 se ha explicado, parcialmente, el intercambio de mensajes entre servidor y agente. Es importante recalcar que, en este proyecto, el servidor se ha

configurado para que escuche en la dirección *IP* 192.168.153.5 y en el puerto 33033. A continuación se especifica, secuencialmente, como procede el servidor cuando recibe una muestra de un agente, en la figura 4.2 se puede observar esta comunicación de manera gráfica:

- Una vez el servidor recibe una muestra, compara el *hash* de esa muestra con el enviado por el usuario, al principio de la comunicación.

- Si los *hashes* coinciden, el servidor realizará un análisis preliminar, a través de la *API* de *VirusTotal*. Este análisis preliminar puede obtener como resultado 4 calificaciones diferentes (malicioso, sospechoso, no malicioso y no sospechoso). La intención de este análisis preliminar es el ahorro de recursos en el servidor, consultando la base de datos de *VirusTotal*.

- Dicho resultado se cifra y es comunicado al agente, para que decida como quiere proceder.

- Si el usuario de ese equipo decide proceder con un análisis, la muestra se envía a la *sandbox*.

- Los resultados con información de las conexiones de red, generados por el módulo de reporte personalizado *geolookup.py*, son enviados al equipo *Debian 11* a través de *filebeat*.

- Estos resultados y los que genera el propio *CAPEv2*, se pueden consultar de modos:

    - A través de la dirección del *proxy-web* del *host* (192.68.153.5:3333), se pueden consultar los datos detallados sobre el servicio *web*, proporcionado por *CAPEv2*.

    - A través de la dirección 192.68.153.2:5601, se pueden consultar los relativos a las conexiones de red, tal y como se han configurado de manera personalizada en *Kibana*.

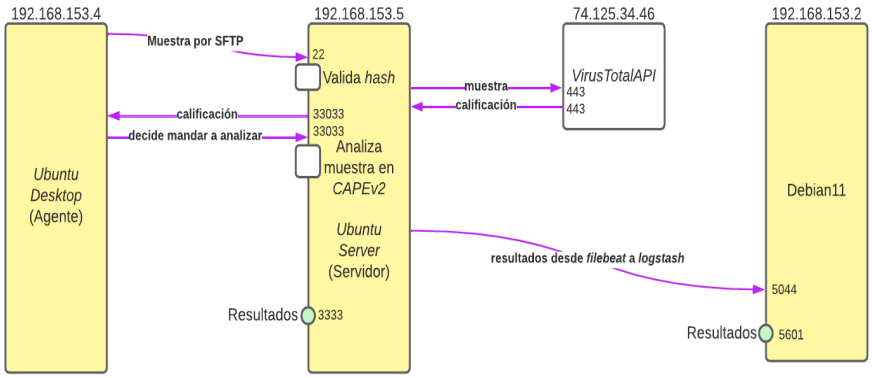

Figura 4.2: Representación de la comunicación realizada por el servidor tras recibir una muestra.

4.4.3. Agente

El agente se debe iniciar desde la terminal en *Linux* o desde la *CMD* en *Windows*, y se ha programado con los siguientes parámetros de entrada:

- *"m"* : Indica el método que se quiere usar para introducir los parámetros relacionados con el servidor. Acepta dos modos:
  - *"auto"*: El modo *auto* requiere que se indiquen los parámetros *"-c"* y *"-d"* mediante la terminal.
  - *"user"*: El modo *user* abre una ventana emergente, por la que se deben introducir los parámetros que se indican. Este modo obvia los parámetros *"-c"* y *"-d"*. La ventana, creada por este modo, se muestra en la figura 4.3.

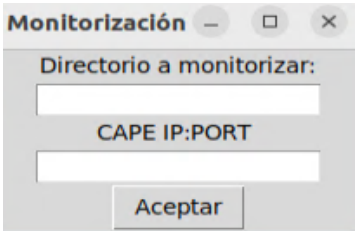

Figura 4.3: Interfaz gráfica del agente programado.

- *"c"*: Indica la ruta del archivo de configuración *.conf*, en el que se incluyen los parámetros de conexión al servidor con el siguiente formato:

```
[analyzer]
ip =
port =
```

- *"d"*: Indica el directorio que se quiere monitorizar.

Este agente es capaz de identificar el último fichero añadido en la carpeta que le sea especificada. Además, es el encargado de establecer con el servidor la comunicación, indicada en las figuras 4.1 y 4.2.

4.5. Reporte de los datos generados

Cómo bien se ha mencionado anteriormente, en este proyecto se ha creado e integrado un módulo de reporte en la aplicación *CAPEv2*. Dicho módulo se ha implementado para ejemplificar las posibilidades que esta aplicación proporciona, gracias a su arquitectura modular.

El módulo que se ha creado obtiene la latitud y la longitud, de manera aproximada, de todas las direcciones involucradas en un análisis. Para ello se ha añadido un *script* en el directorio */opt/CAPEv2/modules/reporting/*, que se puede ver en el anexo D. En este *script* se añaden al reporte generado por *Suricata* (*eve.json*), la longitud y la latitud de cada dirección *IP*. Estas medidas se obtienen a través de consultas a una *API* pública.

Para implementar correctamente un módulo de reporte integrado en la arquitectura de *CAPEv2*, hay que programar el *script* de reporte teniendo en cuenta los siguientes requisitos:[2]

- Crear una clase con el nombre que le quieres dar a tu módulo de reporte.
- Declarar esa clase, de tal modo que herede de la clase *Report*.
- Tener una función llamada *run()*, que realice las operaciones principales.
- Intentar capturar la mayoría de las excepciones y generar un *CuckooReportError*, para notificar los problemas que se pudieran generar durante el funcionamiento de este módulo.

Para la visualización de los datos, proporcionados por este módulo de reporte, se ha generado, en *Kibana*, el panel que se muestra en la figura 4.4. En concreto, el panel presentado, en la figura 4.4, corresponde al análisis de *WannaCry* que se expande en la sección 5.3.1.1.

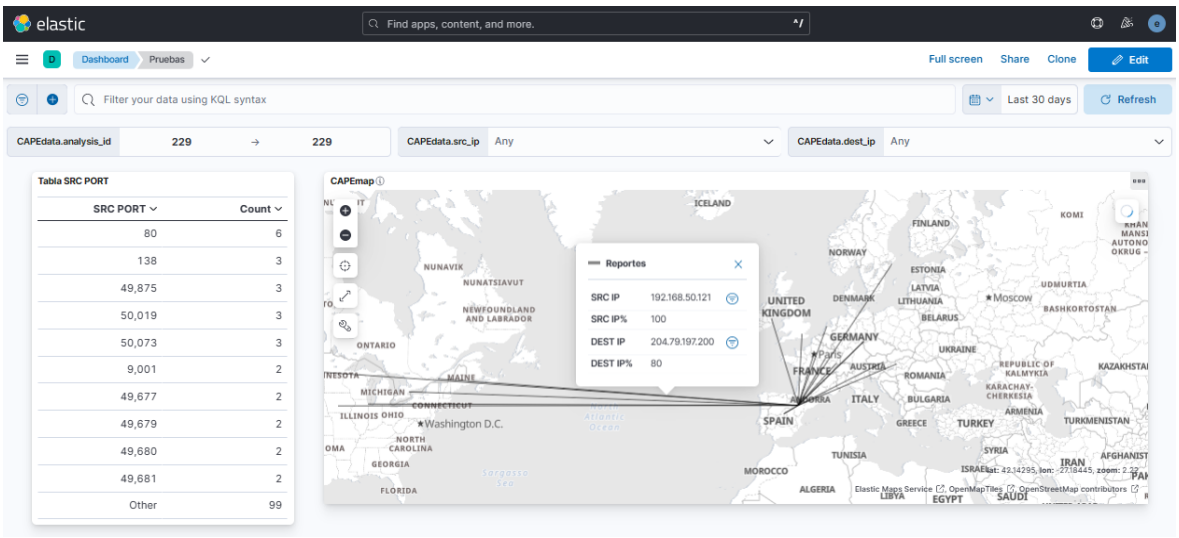

Figura 4.4: Panel de control generado en *Kibana*.

Este panel proporciona una tabla, en la que se indica el número de conexiones asociadas a cada puerto y un mapa en el que se muestran geográficamente, unidas por una línea, las conexiones realizadas desde el *guest* durante el análisis. Si se pulsa en estas líneas se puede observar tanto la dirección de origen, como la de destino. Hay que mencionar, que este panel permite filtrar por el número identificativo de un análisis, por dirección de origen y de destino.

Capítulo 5

Pruebas realizadas y resultados

Las pruebas realizadas se han dividido principalmente en las dos etapas que se describen en las secciones 5.2 y 5.3.

5.1. Formato de los resultados de un análisis

Los resultados de un análisis, se muestran por la interfaz web de *CAPEv2*, tal y como se observa en la figura 5.1. En la navegación sobre los resultados, se encuentran diferentes secciones habilitadas, entre las que se tienen que recalcar las principales:

- *Quick Overview*: En esta sección se proporciona información en términos generales del análisis, como pueden ser capturas de pantalla que se han realizado en su transcurso o posibles indicadores de compromiso extraídos de los análisis. También se muestra, en el caso de que la hubiera, la detección realizada sobre la muestra. Esta información se visualiza en la figura 5.2.

- *Behavioral Analysis*: Corresponde al comportamiento que tiene la muestra. Se expande información a nivel de sistema, como puede ser el árbol de procesos que ha sido generado y todos los elementos con los que interactúa, como son las instrucciones que ejecuta o las direcciones de memoria a las que accede.

- *Network Analysis*: Se muestran las conexiones a nivel de red que provoca la ejecución de la muestra y se identifica cada conexión con su protocolo. También se muestran las alertas, que ha identificado *Suricata*.

- *Dropped Files*: Incluye información de los ficheros que ha lanzado la muestra analizada.

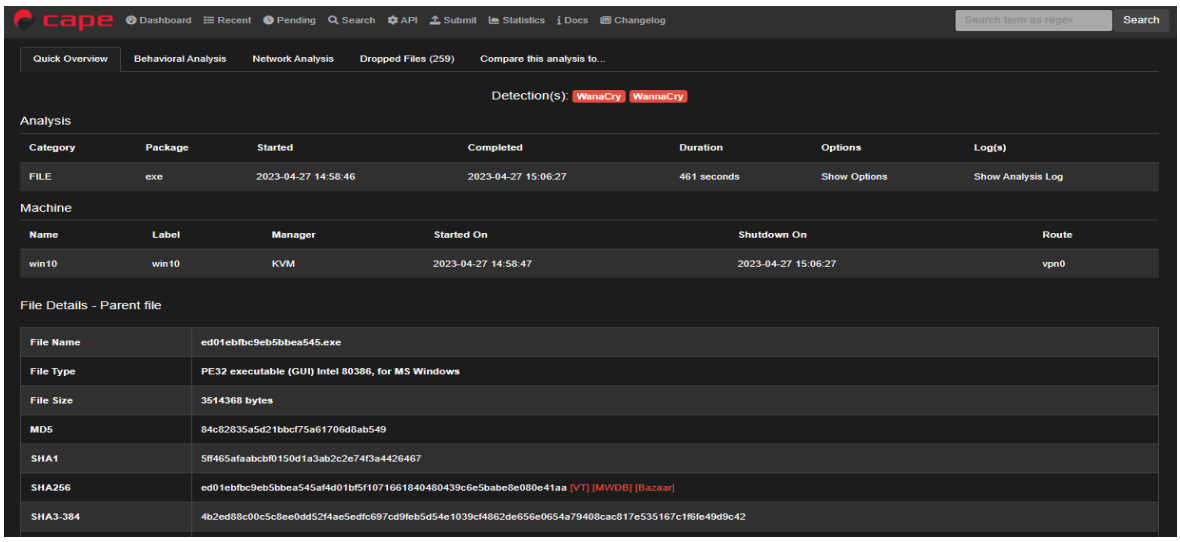

Figura 5.1: Apartados *Overview* en los resultados de un análisis.

Figura 5.2: Apartados interesantes en *Overview*, para los resultados de un análisis.

5.2. Puesta en marcha previa a análisis

Para poner en marcha la *sandbox*, antes de realizar pruebas con *malware* real, se han realizado un total de 216 análisis. Dado que, tras la configuración de todos los servicios necesarios, ha sido necesario ajustar, a medida que se realizaban los análisis, diferentes opciones que causaban errores.

Estos errores han surgido tanto en el análisis, como en el procesado de las muestras. Las causas de estos, han ido desde malas configuraciones, hasta situaciones que no contemplaba el propio programa, como se ha dado en un apartado, durante la configuración de la sesión interactiva.

En el ejemplo de la puesta en marcha de la sesión interactiva, se contactó con los actuales desarrolladores de *CAPEv2*. Después de una larga conversación, que se puede

ver en la sección *Issue #1508* [19] de su *GitHub*, añadieron un pequeño parche que habilitó una funcionalidad no contemplada. En el caso de los errores provocados por malas configuraciones, en el anexo A se especifican las configuraciones realizadas, en cada *guest*, para que no haya ningún problema. Ya que, por ejemplo, a la hora de procesar la información, durante los primeros análisis realizados, se excedía el *timeout* de procesado. Tras inspeccionar la utilidad de procesado de *CAPEv2*, activando la opción del *debugger* de la figura 5.3, se observó que el módulo de procesado se estancaba en la etapa *Network Analysis*. Teniendo en cuenta lo mencionado, se examinaron los archivos generados por el análisis. De este modo, se identificó que el tamaño del archivo *.pcap*, generado por el análisis, era del orden de *MegaBytes*, tamaño mucho mayor del común.

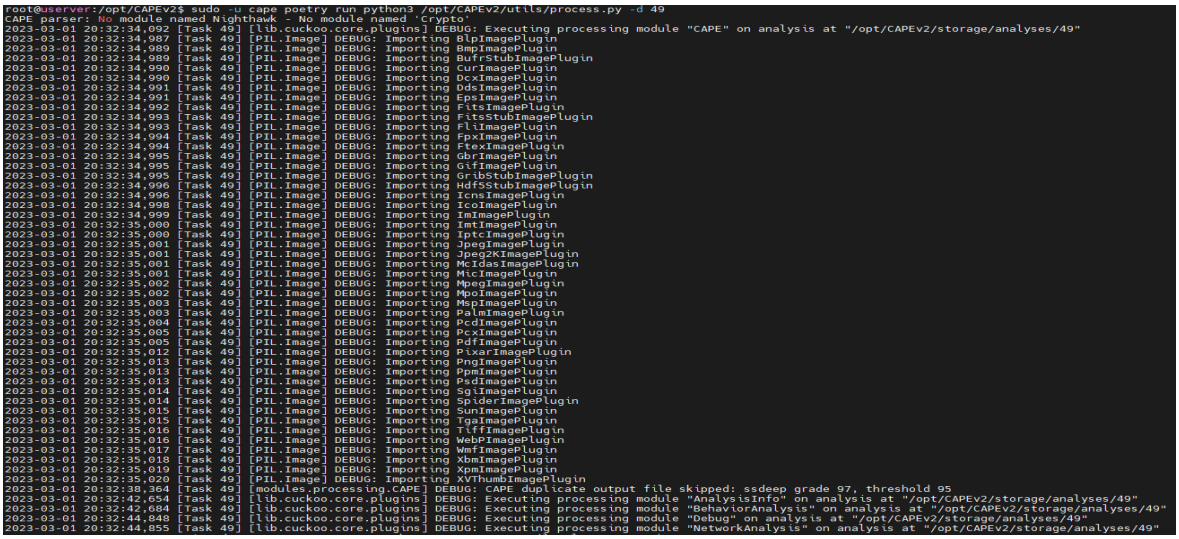

Figura 5.3: Utilidad de procesado con opción *debug* activada en *CAPEv2*.

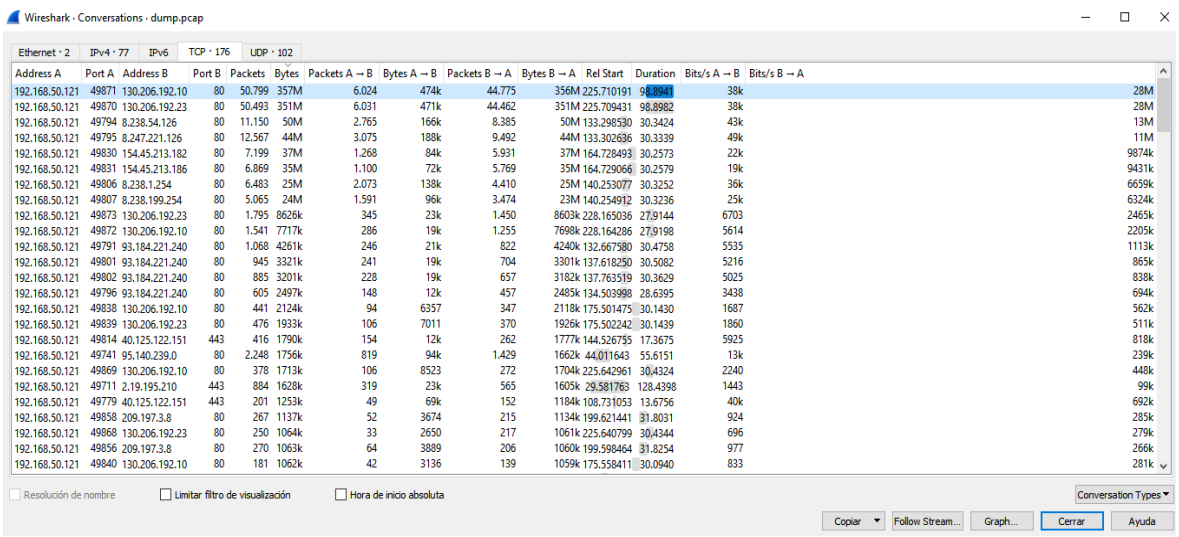

Figura 5.4: Conexiones correspondientes a análisis con *Windows Update* activado.

En vista de la información recopilada, se inspeccionó el *.pcap* generado, en busca de anomalías. Observando así, que el tráfico significativo correspondía al intercambio de mensajes con dos direcciones *IP*, se visualiza en la figura 5.4. A raíz de lo explicado, se inspeccionó estas direcciones a través de medios de recopilación de información, como *TalosIntelligence*, obteniendo una relación de esa dirección con empresas relacionadas con *Microsoft*. Haciendo ver de esta manera, que el problema que se estaba teniendo era una consecuencia directa de haber deshabilitado de manera incorrecta la utilidad *Windows Update*.

5.3. Análisis con *malware* real

En las siguientes secciones se van a mostrar los resultados obtenidos, de los análisis de muestras reales de *malware*. Se realizará una comparativa de estos, con los informes oficiales de esas muestras. Esta se calificará, basándonos en la obtención de *IOC*, respecto a los informes oficiales.

5.3.1. *Malware* detectado

5.3.1.1. *WannaCry*

En el análisis realizado sobre la muestra de *WannaCry*, cuyo *MD5 hash* corresponde con *84c82835a5d21bbcf75a61706d8ab549*, se han extraído ciertos *IOC*, explicados a continuación. Estos corresponden con los que se indican en el informe oficial, elaborado por el *CCN* [20]. En primer lugar, se ha podido observar que la fecha interna de creación del programa corresponde al 20 de noviembre de 2010, siendo la misma que se muestra en el análisis oficial. En la imagen superior de la figura 5.5 se muestra la información oficial, mientras que, en la imagen inferior, aparece la relativa al análisis del proyecto.

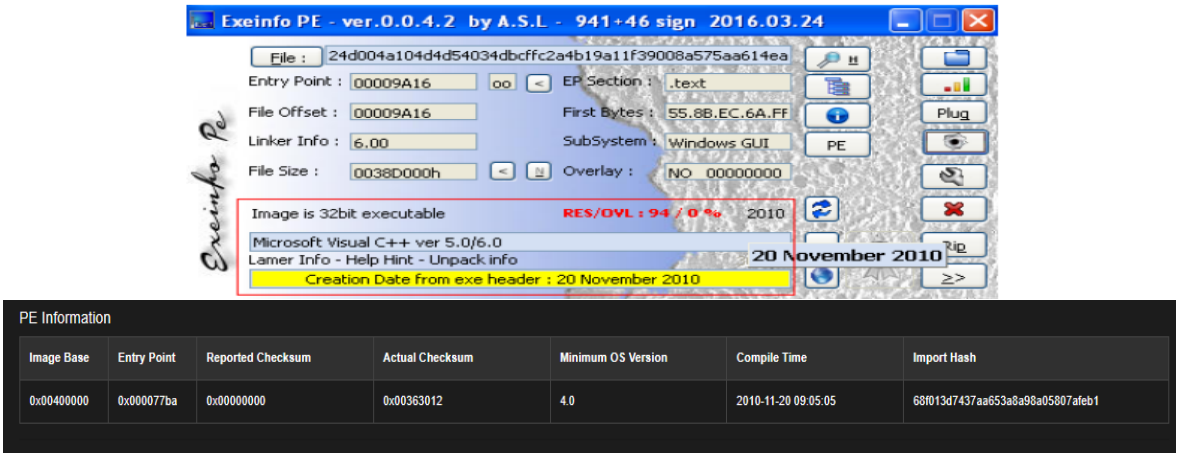

Figura 5.5: Comparativa de fecha de creación de muestra *WannaCry*.

Además, durante el análisis, se ejecuta la orden que se indica en el informe, que garantiza la persistencia en el sistema. Se muestra en la figura 5.6, con el formato anteriormente descrito.

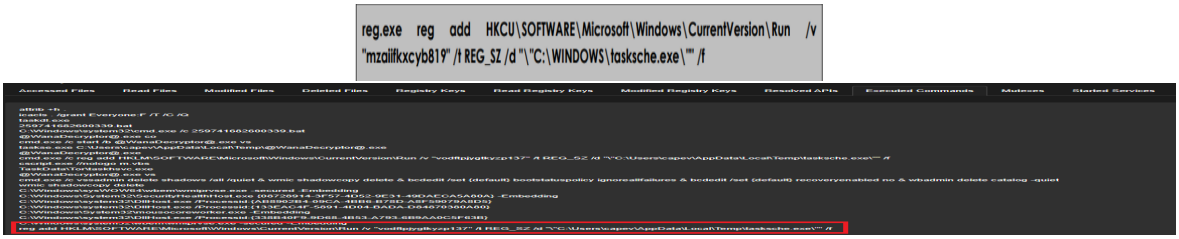

Figura 5.6: Comparativa de la orden ejecutada para garantizar persistencia por la muestra *WannaCry*.

También, en la figura 5.7, se presentan las instrucciones ejecutadas por la muestra, para borrar las *Shadow Copies* presentes en el equipo, evitar que el equipo arranque en modo de recuperación y borrar los catálogos de copias de seguridad.

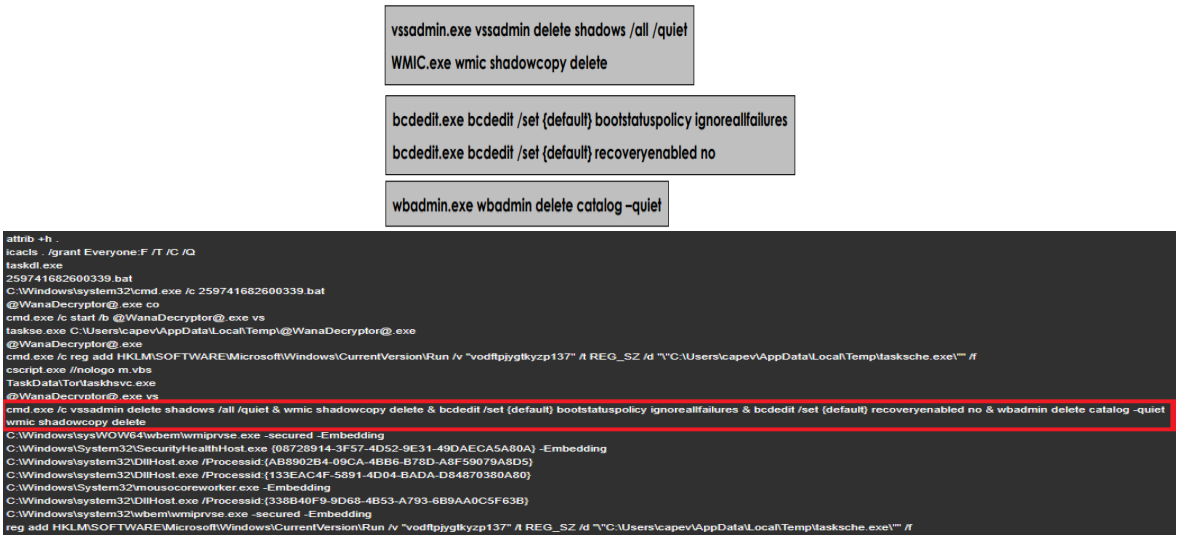

Figura 5.7: Comparativa de las órdenes ejecutadas para evitar la recuperación del sistema por la muestra *WannaCry*.

Por último, en la figura 5.8, se visualiza una captura, en la que se resalta cómo la muestra crea un *mutex* con el mismo nombre que el indicado en [20].

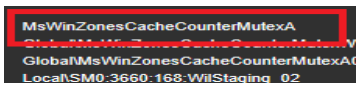

Figura 5.8: Comparativa del *mutex* creado por la muestra *WannaCry*.

5.3.1.2. *Emotet*

En el análisis realizado sobre la muestra de *Emotet*, cuyo *MD5 hash* corresponde con *32289068803d70d40f42172c33760016*, se han extraído ciertos indicadores de compro-

miso del propio análisis. En este caso se van a mostrar los potenciales *IOC* extraídos, sin realizar una comparación con el informe oficial del *CCN*. Debido a la diferente naturaleza de la muestra ejecutable utilizada, con la muestra del informe, correspondiente a un archivo *.doc*.

En la figura 5.9 se muestran las coincidencias que ha tenido la ejecución de la muestra, con las firmas identificadas como comportamientos potencialmente peligrosos. Entre estas firmas se encuentran las coincidencias encontradas con las reglas *YARA*.

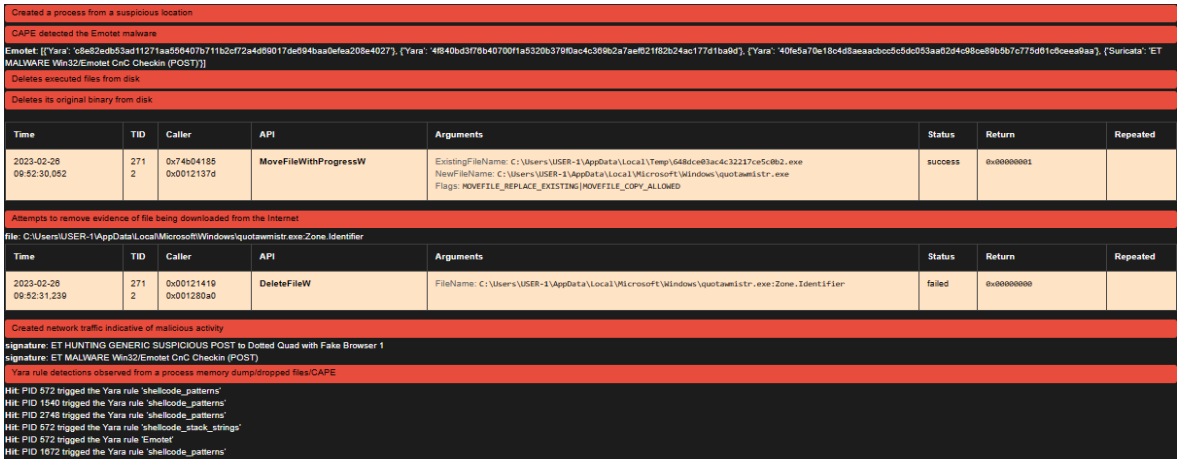

Figura 5.9: Firmas de mayor riesgo, detectadas en la ejecución de la muestra de *Emotet*.

En cuanto a las conexiones de red, se identifica una alerta de *Suricata*, figura 5.10, relacionada con actividad sospechosa propia de *Emotet*. Esta actividad corresponde al intercambio de mensajes, correspondiente a un *POST* y a un *GET* por parte del *guest*, con una dirección ubicada en Corea del Sur. Este intercambio viene indicado en la figura 5.11 y la respuesta al *GET* realizado por el *guest*, corresponde con un archivo *.html*.

Timestamp	Source IP	Source Port	Destination IP	Destination Port	Protocol	GID	SID	REV	Signature	Category	Severity
2023-05-26 09:18:10.549662+0000	192.168.50.91 [VT]	49243	189.199.94.178 [VT]	80	TCP	1	2035053	4	ET MALWARE Win32/Emotet CnC Checkin (POST)	Malware Command and Control Activity Detected	1
2023-05-26 09:18:10.549662+0000	192.168.50.91 [VT]	49243	189.199.94.178 [VT]	80	TCP	1	2018358	10	ET HUNTING GENERIC SUSPICIOUS POST to Dotted Quad with Fake Browser 1	Potentially Bad Traffic	2
2023-05-26 09:18:40.753450+0000	192.168.50.91 [VT]	49251	121.135.19.214 [VT]	80	TCP	1	2035053	4	ET MALWARE Win32/Emotet CnC Checkin (POST)	Malware Command and Control Activity Detected	1
2023-05-26 09:18:40.753450+0000	192.168.50.91 [VT]	49251	121.135.19.214 [VT]	80	TCP	1	2018358	10	ET HUNTING GENERIC SUSPICIOUS POST to Dotted Quad with Fake Browser 1	Potentially Bad Traffic	2

Figura 5.10: Alertas disparadas por *Suricata* durante la ejecución de la muestra de *Emotet*.

5.3.2. *Malware* no detectado

En los análisis correspondientes a las muestras más recientes de *malware*, las detecciones no se han llevado a cabo con éxito. Esto se debe a las instrucciones observadas, correspondientes a técnicas *ANTI-VM*, que se explican a continuación:

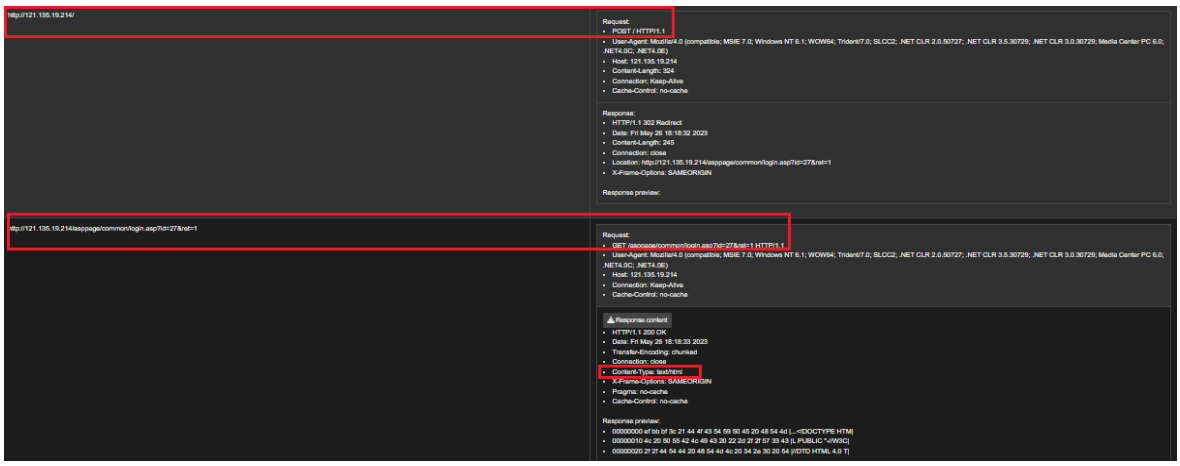

Figura 5.11: Contenido de la conexión potencialmente peligrosa, detectada en las alestas de *Suricata*.

- Las alertas, que se muestran en la figura 5.12, corresponden a detecciones, realizadas por el servicio de *Suricata* en la etapa de procesado. Estas detecciones corresponden con instrucciones marcadas como *ANTI-DEBUG*, que se podrían estar usando para detectar el entorno de análisis, finalizando, tras esta detección, la ejecución de la muestra.

Suricata Alerts

Timestamp	Source IP	Source Port	Destination IP	Destination Port	Protocol	GID	SID	REV	Signature	Category	Severity
2023-05-25 11:31:09.071418+0000	209.197.3.8 [VT]	80	192.168.50.121 [VT]	49763	TCP	1	2018959	4	ET POLICY PE EXE or DLL Windows file download HTTP	Potential Corporate Privacy Violation	1
2023-05-25 11:31:09.197445+0000	209.197.3.8 [VT]	80	192.168.50.121 [VT]	49763	TCP	1	2015744	6	ET INFO EXE IsDebuggerPresent (Used in Malware Anti-Debugging)	Misc activity	3

Suricata Alerts

Timestamp	Source IP	Source Port	Destination IP	Destination Port	Protocol	GID	SID	REV	Signature	Category	Severity
2023-05-24 06:55:22.538377+0000	192.168.50.121 [VT]	61022	8.8.8.8 [VT]	53	UDP	1	2023883	4	ET DNS Query to a *.top domain - Likely Hostile	Potentially Bad Traffic	2
2023-05-24 06:55:38.795649+0000	209.197.3.8 [VT]	80	192.168.50.121 [VT]	49813	TCP	1	2018959	4	ET POLICY PE EXE or DLL Windows file download HTTP	Potential Corporate Privacy Violation	1
2023-05-24 06:55:38.923817+0000	209.197.3.8 [VT]	80	192.168.50.121 [VT]	49813	TCP	1	2015744	6	ET INFO EXE IsDebuggerPresent (Used in Malware Anti-Debugging)	Misc activity	3

Suricata Alerts

Timestamp	Source IP	Source Port	Destination IP	Destination Port	Protocol	GID	SID	REV	Signature	Category	Severity
2023-05-25 11:41:12.349608+0000	212.145.41.170 [VT]	80	192.168.50.121 [VT]	49757	TCP	1	2018959	4	ET POLICY PE EXE or DLL Windows file download HTTP	Potential Corporate Privacy Violation	1
2023-05-25 11:41:12.399528+0000	212.145.41.170 [VT]	80	192.168.50.121 [VT]	49757	TCP	1	2015744	6	ET INFO EXE IsDebuggerPresent (Used in Malware Anti-Debugging)	Misc activity	3

Figura 5.12: Alertas de *Suricata*, correspondientes a muestras de *Adloader, Mint y Msil*.

- Las firmas identificadas en la ejecución de una muestra de *Nymaim*, permiten identificar la aplicación de un posible método *ANTI-VM* en su ejecución. Mediante la llamada a la *API SetUnhandledExceptionFilter*, con un retorno de valor '1'. Lo cual, podría ser interpretado por la muestra como un indicador de análisis. Por otro lado, se observa cómo varios procesos, han disparado distintas reglas

YARA. Por tanto, pese a no tener una detección en esta muestra, hay indicadores que podrían permitir a un analista ajustar la configuración del *guest*, para evitar la eficacia que, en este caso, han tenido las técnicas *ANTI-VM* del *malware* analizado. Los indicadores descritos se observan en la figura 5.13.

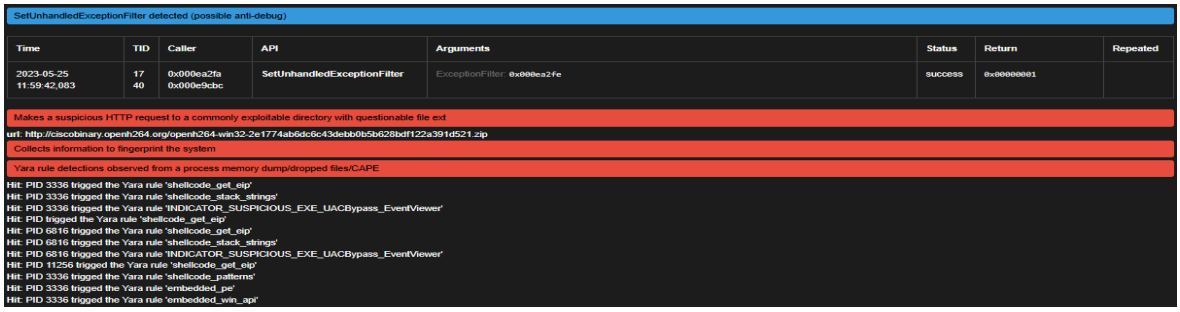

Figura 5.13: Firmas, correspondientes a *Nymaim*.

- Para finalizar la sección de resultados, se ha realizado el análisis de una muestra correspondiente a la familia de *malware Mint*. En las firmas relativas a su ejecución, en la figura 5.14, se observa como esta muestra recopila información relativa a los componentes del sistema. Probablemente, este reconocimiento haya disparado la finalización de la ejecución de la muestra.

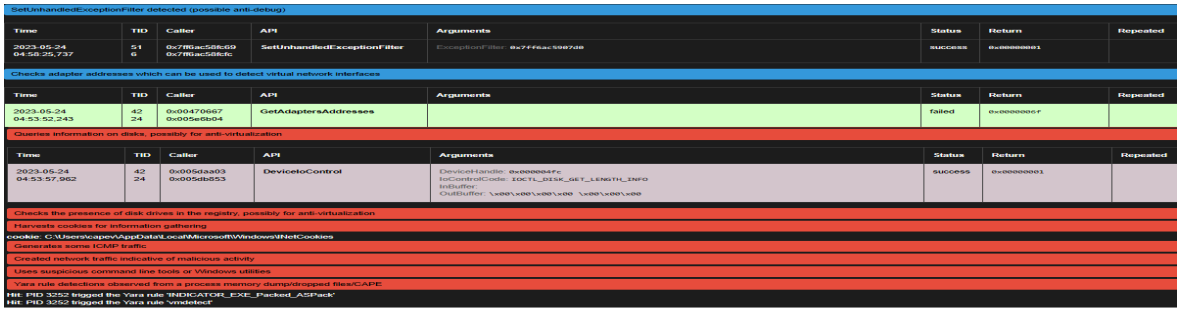

Figura 5.14: Firmas, correspondientes a *Mint*.

Capítulo 6

Conclusiones y líneas futuras

Durante el desarrollo de este trabajo de fin de grado, he experimentado una curva de aprendizaje notablemente acelerada, la cual refleja la naturaleza de toda la carrera, en la que se han planteado una serie de desafíos, que han requerido adquirir rápidamente conocimientos y habilidades para su resolución, en un tiempo limitado.

En gran medida, la capacidad para resolver problemas desarrollada a lo largo de la carrera, combinada con los conocimientos adquiridos en redes y programación, ha sido fundamental para solucionar la mayoría de los obstáculos presentados durante el desarrollo de este trabajo. Ya fuese a través de la captura y evaluación de paquetes, para identificar la causa de los problemas, o al contactar con los desarrolladores de *CAPEv2* en otro idioma, utilizando un lenguaje técnico.

En resumen, pese a tener días de trabajo complicados, en los que planteas y pruebas muchas soluciones y no funciona ninguna o en los que, aparentemente, por ninguna razón, los componentes que se habían montado y puesto en marcha dejan de funcionar. He disfrutado el camino recorrido para la implementación de todo lo que se ha desarrollado.

6.1. Conclusiones

En la realización del proyecto, *CAPEv2* ha sido el núcleo sobre el que se ha trabajado. Se ha observado que esta aplicación, gracias a las personas que participan de manera altruista, proporciona funcionalidades de gran valor a coste cero. Muchas de estas funcionalidades no se han explorado en este proyecto, pero resultan muy interesantes, cómo puede ser el modo de trabajo distribuido que proponen en su documentación los desarrolladores.

Es importante recalcar que al trabajar con una aplicación de *Github*, se ha tenido que trabajar con la herramienta *git*, comprendiendo de este modo, como actualizar y mantener, correctamente, las aplicaciones que se encuentran en esta plataforma.

Además, se ha comprendido el funcionamiento de las aplicaciones actuales, entendiendo como se realiza la comunicación de los diferentes servicios y la necesidad de implementar un gestor de errores de manera correcta, en una aplicación, para proporcionar a los usuarios cierta autonomía a la hora de resolver los errores que puedan surgir, proporcionándo unos *logs* enriquecidos.

Otro punto clave del proyecto ha sido el uso de las herramientas proporcionadas por *Elastic*. Ya que han permitido una gran comprensión de cómo es tratada la información, a la hora de generar un entorno, en el que esta se pueda consultar de manera eficiente. Teniendo que crear, tanto el mapeado del índice en el que se han almacenado los datos, como las políticas de rotación en los índices de almacenaje, para poder aprovechar, de manera eficiente, el espacio disponible en el equipo.

Por otro lado, se ha profundizado en las trazas que genera el *software* de virtualización sobre sus *guest*. Observando y modificando las mismas mediante diferentes técnicas, que hay que desarrollar, puesto que a lo largo del capítulo 5, se han llevado a cabo varios análisis, en los que se ha manifestado la necesidad de mejorar las medidas *ANTI-VM*, aplicadas a los *guest* de la *sandbox*. Ya que, cómo se ha visualizado en los resultados, un número significativo de muestras, son capaces de detectar que se están ejecutando en un entorno de análisis y detienen su ejecución, como consecuencia.

Por último, hay que mencionar que se han observado dos funcionalidades que no han sido implementadas, necesarias en un entorno de producción. A continuación se detallan las principales características:

- Diseñar la interacción con un sistema de cuarentena. Dado que, el grado de control planteado sobre los equipos puede que no sea suficiente, para un entorno de producción. Ya que el encargado, con acceso a las interfaces *web*, donde se describen los detalles de los análisis, debería tomar medidas de manera manual en el caso de encontrarse con algún incidente. Por estos motivos se debería plantear un control centralizado, desde el servidor, para indicar al agente que medidas tomar con la muestra analizada. Entre estas medidas, podría estar la aplicabilidad de un protocolo de cuarentena sobre el fichero analizado.

- Programar el servidor para que acepte solicitudes de análisis de manera concurrente. Al ser una prueba de concepto se ha comprobado el correcto funcionamiento para un solo equipo en la red. Por lo que sería interesante programar el servidor generalizando este comportamiento, para una red de mayor tamaño. De esta manera se podrían plantear diferentes soluciones, en el caso de que la implementación proporcionada fuese poco escalable.

6.2. Líneas futuras

En lo que respecta al diseño de los *guest*, se deberían volver a crear, aplicando técnicas *ANTI-VM* más avanzadas. Estas técnicas se podrían diseñar visualizando, detalladamente, cómo las muestras que han detectado el entorno virtual han llevado a cabo esta detección. En el nuevo *guest*, una vez identificado el método de detección, se podrían aplicar las modificaciones necesarias para neutralizarlo.

Por otro lado, pese a haber desactivado las funciones de *Windows*, que contaminan las capturas de las conexiones en los análisis. Siguen existiendo ciertas conexiones que se repiten en todos los análisis. Se podría crear un módulo de procesado, que eliminase las conexiones que se ejecutan en el *guest* por defecto. Este módulo se plantearía identificando y guardando en un fichero, en un análisis de código no malicioso, las conexiones realizadas. Tras la obtención de este fichero, se continuaría con la creación de un *script* de procesado que filtrase, consultando el fichero creado previamente, todas las conexiones que coincidieran.

Cómo última observación, en los aspectos relacionados con las máquinas *guest*, se tendría que explorar la implementación de un *guest* con el reciente sistema operativo *Windows 11*. Ya que se desconoce cómo se deben realizar, en este sistema operativo, las configuraciones pertinentes, para el correcto desarrollo de los análisis. Sería muy interesante, dado que probablemente este *S.O.* se extienda de manera significativa, a lo largo de los próximos años.

En lo relativo al software usado, para la solución aportada, se debería plantear la adición de plataformas en las que se comparte información sobre amenazas, como puede ser *MISP*. De esta manera, se contribuiría a la mejora en la detección y el análisis de *malware*, compartiendo los indicadores de compromiso identificados en el análisis realizado en la *sandbox*.

Para concluir, hay que destacar que la implementación de todo el *software* relacionado con la solución planteada ha sido un proceso laborioso, que ha requerido una considerable cantidad de horas de trabajo. Sin embargo, se podría estudiar la posibilidad de replicar el proceso realizado a lo largo del proyecto, de manera automática, a través del *software Ansible*. De este modo, se podría gestionar la configuración y el despliegue de la estructura de *sandboxing*, que se ha planteado, de una manera sencilla.

Capítulo 7

Bibliografía

[1] AyudaLey. *Qué es un ciberataque y qué tipos existen.* https://ayudaleyprotecciondatos.es/2018/10/08/ciberataque/, Oct 2018.

[2] Kevin O'Reilly. Capev2. https://github.com/kevoreilly/CAPEv2, 2023.

[3] Csongor Horvath, Trygve Thomessen, and Gabor Sziebig. Overview of modern teaching equipment that supports distant learning. *Recent Innovations in Mechatronics*, 5, 01 2018.

[4] Calvin Nobles, Sharon Burton, and Darrell Burrell. *Cybercrime as a Sustained Business*, pages 98–120. 03 2023.

[5] SonicWall. 2023 sonicwall cyber threat report. "https://www.sonicwall.com/medialibrary/en/white-paper/2023-cyber-threat-report.pdf", 2023.

[6] Liselotte Boven, Renske Kusters, Derrick Tin, Frits Osch, H. Cauwer, Linsay Ketelings, Madhura Rao, Christian Dameff, and Dennis Barten. Hacking acute care: A qualitative study on the healthcare impacts of ransomware attacks against hospitals, 02 2023.

[7] Adrienne de Ruiter. The distinct wrong of deepfakes. *Philosophy & Technology*, 34(4):1311–1332, December 2021.

[8] Michael Maass, Adam Sales, Benjamin Chung, and Joshua Sunshine. A systematic analysis of the science of sandboxing. *PeerJ Computer Science*, 2:5–6, 01 2016.

[9] Cedric Pernet. *ANY.RUN vs. Joe Sandbox: Malware analysis tools comparison.* https://www.techrepublic.com/article/anyrun-vs-joe-sandbox/, March 2023.

[10] Alexey Lapshin. Any.run. https://any.run/, 2023.

[11] Theodoros Apostolopoulos, Vasilios Katos, Kim-Kwang Raymond Choo, and Constantinos Patsakis. Resurrecting anti-virtualization and anti-debugging: Unhooking your hooks. *Future Generation Computer Systems*, 116:393–405, 2021.

[12] A. Yokoyama, K. Ishii, R. Tanabe, Y. Papa, K. Yoshioka, T. Matsumoto, T. Kasama, D. Inoue, M. Brengel, M. Backes, et al. Sandprint: Fingerprinting malware sandboxes to provide intelligence for sandbox evasion. *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 165–187, 2016.

[13] Anoirel Issa. Anti-virtual machines and emulations. *Journal in Computer Virology*, 8(4):141–149, November 2012.

[14] Shay Banon. Elastic stack. `https://www.elastic.co/`, 2023.

[15] Virustotal. Virustotal api. `https://developers.virustotal.com/reference/overview`, 2023. APIv3.

[16] Apache Software Foundation. Apache guacamole. `https://guacamole.apache.org/`, 2023.

[17] Igor Sysoev. Nginx. `https://nginx.org/`, 2023.

[18] Andriy Brukhovetskyy. Kvm settings for malware analysis. `https://www.doomedraven.com/2016/05/kvm.html#modifying-kvm-qemu-kvm-settings-for-malware-analysis`, 2016.

[19] Mario Romeo. Capev2 issue #1508. `https://github.com/kevoreilly/CAPEv2/issues/1508`, 2023.

[20] CCN-CERT. Informe código dañino *CCN-CERT ID-17/17*. `https://www.ccn-cert.cni.es/informes/informes-ccn-cert-publicos/2169-ccn-cert-id-17-17-codigo-danino-wannacry-1/file.html`, May 2017.

[21] Mikrotik. *Mikrotik RouterOS v7*. `https://mikrotik.com/download`, 2023.

Anexos

Anexos A

INSTALACIÓN DE CAPEv2

La instalación en de *CAPEv2* tiene dos fases, que corresponde al proceso correspondiente al *host* y al relativo al *guest*.

A.1. Instalación en *host*

Para instalar todo lo necesario en el *host*, se han ejecutado las siguientes instrucciones:

```
wget https://raw.githubusercontent.com/kevoreilly/CAPEv2/master/
   ↪ installer/cape2.sh
wget https://raw.githubusercontent.com/kevoreilly/CAPEv2/master/
   ↪ installer/kvm-qemu.sh
sudo ./kvm-qemu.sh all xxx | tee kvm-qemu.log
chmod a+x cape2.sh
sudo ./cape2.sh base cape | tee cape.log
poetry install
sudo ./cape2.sh guacamole
sudo apt-get install nginx
sudo apt-get install libwebsockets-dev
```

A.2. Ficheros de configuración del *host*

A continuación se muestran íntegramente los ficheros de configuración que han sido modificados, a lo largo del proyecto.

- */opt/CAPEv2/custom/conf/cuckoo.conf*

```
[cuckoo]

# Which category of tasks do you want to analyze?
categories = static, pcap, url, file
```

```
# If turned on, Cuckoo will delete the original file after its
    ↪ analysis
# has been completed.
delete_original = off

# Archives are not deleted by default, as it extracts and "
    ↪ original file" become extracted file
delete_archive = on

# If turned on, Cuckoo will delete the copy of the original file
    ↪ in the
# local binaries repository after the analysis has finished. (On *
    ↪ nix this
# will also invalidate the file called "binary" in each analysis
    ↪ directory,
# as this is a symlink.)
delete_bin_copy = off

# Specify the name of the machinery module to use, this module
    ↪ will
# define the interaction between Cuckoo and your virtualization
    ↪ software
# of choice.
machinery = kvm

# Enable creation of memory dump of the analysis machine before
    ↪ shutting
# down. Even if turned off, this functionality can also be enabled
    ↪  at
# submission. Currently available for: VirtualBox and libvirt
    ↪ modules (KVM).
memory_dump = on

# When the timeout of an analysis is hit, the VM is just killed by
    ↪  default.
# For some long-running setups it might be interesting to
    ↪ terminate the
# moinitored processes before killing the VM so that connections
    ↪ are closed.
terminate_processes = off

# Enable automatically re-schedule of "broken" tasks each startup.
# Each task found in status "processing" is re-queued for analysis
    ↪ .
reschedule = off

# Fail "unserviceable" tasks as they are queued.
# Any task found that will never be analyzed based on the
```

```
    ↪ available analysis machines
# will have its status set to "failed".
fail_unserviceable = on

# Limit the amount of analysis jobs a Cuckoo process goes through.
# This can be used together with a watchdog to mitigate risk of
    ↪ memory leaks.
max_analysis_count = 0

# Limit the number of concurrently executing analysis machines.
# This may be useful on systems with limited resources.
# Set to 0 to disable any limits.
max_machines_count = 10

# Limit the amount of VMs that are allowed to start in parallel.
    ↪ Generally
# speaking starting the VMs is one of the more CPU intensive parts
    ↪  of the
# actual analysis. This option tries to avoid maxing out the CPU
    ↪ completely.
max_vmstartup_count = 5

# Minimum amount of free space (in MB) available before starting a
    ↪  new task.
# This tries to avoid failing an analysis because the reports can'
    ↪ t be written
# due out-of-diskspace errors. Setting this value to 0 disables
    ↪ the check.
# (Note: this feature is currently not supported under Windows.)
freespace = 0
# Process tasks, but not reach out of memory
freespace_processing = 15000

# Temporary directory containing the files uploaded through Cuckoo
    ↪  interfaces
# (web.py, api.py, Django web interface).
tmppath = /tmp

# Delta in days from current time to set the guest clocks to for
    ↪ file analyses
# A negative value sets the clock back, a positive value sets it
    ↪ forward.
# The default of 0 disables this option
# Note that this can still be overridden by the per-analysis clock
    ↪  setting
# and it is not performed by default for URL analysis as it will
    ↪ generally
# result in SSL errors
```

```
daydelta = 0

# Path to the unix socket for running root commands.
rooter = /tmp/cuckoo-rooter

# Enable if you want to see a DEBUG log periodically containing
    ↪ backlog of pending tasks, locked vs unlocked machines.
# NOTE: Enabling this feature adds 4 database calls every 10
    ↪ seconds.
periodic_log = off

[resultserver]
# The Result Server is used to receive in real time the behavioral
    ↪  logs
# produced by the analyzer.
# Specify the IP address of the host. The analysis machines should
    ↪  be able
# to contact the host through such address, so make sure it's
    ↪ valid.
# NOTE: if you set resultserver IP to 0.0.0.0 you have to set the
    ↪ option
# `resultserver_ip` for all your virtual machines in machinery
    ↪ configuration.
ip = 192.168.50.1

# Specify a port number to bind the result server on.
port = 2042

# Force the port chosen above, don't try another one (we can
    ↪ select another
# port dynamically if we can not bind this one, but that is not an
    ↪  option
# in some setups)
force_port = yes

pool_size = 0

# Should the server write the legacy CSV format?
# (if you have any custom processing on those, switch this on)
store_csvs = off

# Maximum size of uploaded files from VM (screenshots, dropped
    ↪ files, log)
# The value is expressed in bytes, by default 100MB.
upload_max_size = 100000000

# Prevent upload of files that passes upload_max_size?
do_upload_max_size = no
```

```
[processing]
# Set the maximum size of analyses generated files to process.
    ↪ This is used
# to avoid the processing of big files which may take a lot of
    ↪ processing
# time. The value is expressed in bytes, by default 200MB.
analysis_size_limit = 200000000

# The number of calls per process to process. 0 switches the limit
    ↪  off.
# 10000 api calls should be processed in less than 2 minutes
analysis_call_limit = 0

# Enable or disable DNS lookups.
resolve_dns = on

# Enable or disable reverse DNS lookups
# This information currently is not displayed in the web interface
reverse_dns = off

# Use ram to boost processing speed. You will need more than 20GB
    ↪ of RAM for this feature.
# Please read "performance" section in the documentation.
ram_boost = yes

# Enable PCAP sorting, needed for the connection content view in
    ↪ the web interface.
sort_pcap = on

[database]
# Specify the database connection string.
# Examples, see documentation for more:
# sqlite:///foo.db
# postgresql://foo:bar@localhost:5432/mydatabase
# mysql://foo:bar@localhost/mydatabase
# If empty, default is a SQLite in db/cuckoo.db.
# SQLite doens't support database upgrades!
# For production we strongly suggest go with PostgreSQL
connection = postgresql://cape:sj3cAnC1c0v#@localhost:5432/cape

# Database connection timeout in seconds.
# If empty, default is set to 60 seconds.
timeout =

[timeouts]
# Set the default analysis timeout expressed in seconds. This
    ↪ value will be
```

```
# used to define after how many seconds the analysis will
    ↪ terminate unless
# otherwise specified at submission.
default = 300

# Set the critical timeout expressed in (relative!) seconds. It
    ↪ will be added
# to the default timeout above and after this timeout is hit
# Cuckoo will consider the analysis failed and it will shutdown
    ↪ the machine
# no matter what. When this happens the analysis results will most
    ↪  likely
# be lost.
critical = 60

# Maximum time to wait for virtual machine status change. For
    ↪ example when
# shutting down a vm. Default is 300 seconds.
vm_state = 300

[log_rotation]
# Activate log rotation for cuckoo.log and process.log.
enabled = on
# Keep 30 days of log history (default is 7).
backup_count = 30

[tmpfs]
# only if you using volatility to speedup IO
# mkdir -p /mnt/tmpfs
# mount -t tmpfs -o size=50g ramfs /mnt/tmpfs
# chown cape:cape /mnt/tmpfs
#
# vim /etc/fstab
# tmpfs /mnt/tmpfs tmpfs nodev,nosuid,noexec,nodiratime,size=50g 0
    ↪  0
#
# Add crontab with
# @reboot chown cape:cape /mnt/tmpfs -R
enabled = off
path = /mnt/tmpfs/
# in mb
freespace = 2000
```

- */opt/CAPEv2/custom/conf/kvm.conf*

```
[kvm]
# Specify a comma-separated list of available machines to be used.
```

```
    ↪ For each
# specified ID you have to define a dedicated section containing
    ↪ the details
# on the respective machine. (E.g. cuckoo1,cuckoo2,cuckoo3)
machines = win7,win10

interface = virbr1

[win7]
# Specify the label name of the current machine as specified in
    ↪ your
# libvirt configuration.
label = win7

# Specify the operating system platform used by current machine
# [windows/darwin/linux].
platform = windows

# Specify the IP address of the current virtual machine. Make sure
    ↪ that the
# IP address is valid and that the host machine is able to reach
    ↪ it. If not,
# the analysis will fail. You may want to configure your network
    ↪ settings in
# /etc/libvirt/<hypervisor>/networks/
ip = 192.168.50.91

# (Optional) Specify tags to display
# Tags may be used to specify on which guest machines a sample
    ↪ should be run
# tags = windows_xp_sp3,acrobat_reader_6

# (Optional) Specify the snapshot name to use. If you do not
    ↪ specify a snapshot
# name, the KVM MachineManager will use the current snapshot.
# Example (Snapshot1 is the snapshot name):
snapshot = WIN7_con_DNS

# (Optional) Specify the name of the network interface that should
    ↪ be used
# when dumping network traffic from this machine with tcpdump. If
    ↪ specified,
# overrides the default interface specified in auxiliary.conf
# Example (virbr0 is the interface name):
interface = virbr1

# (Optional) Specify the IP of the Result Server, as your virtual
    ↪ machine sees it.
```

```
# The Result Server will always bind to the address and port
    ↪ specified in cuckoo.conf,
# however you could set up your virtual network to use NAT/PAT, so
    ↪  you can specify here
# the IP address for the Result Server as your machine sees it. If
    ↪  you don't specify an
# address here, the machine will use the default value from cuckoo
    ↪ .conf.
# NOTE: if you set this option you have to set result server IP to
    ↪  0.0.0.0 in cuckoo.conf.
# Example:
# resultserver_ip = 192.168.50.1

# (Optional) Specify the port for the Result Server, as your
    ↪ virtual machine sees it.
# The Result Server will always bind to the address and port
    ↪ specified in cuckoo.conf,
# however you could set up your virtual network to use NAT/PAT, so
    ↪  you can specify here
# the port for the Result Server as your machine sees it. If you
    ↪ don't specify a port
# here, the machine will use the default value from cuckoo.conf.
# Example:
# resultserver_port = 2042

# Set the machine architecture
# Required to auto select proper machine architecture for sample
# x64 or x86
arch = x64

[win10]
# Specify the label name of the current machine as specified in
    ↪ your
# libvirt configuration.
label = win10

# Specify the operating system platform used by current machine
# [windows/darwin/linux].
platform = windows

# Specify the IP address of the current virtual machine. Make sure
    ↪  that the
# IP address is valid and that the host machine is able to reach
    ↪ it. If not,
# the analysis will fail. You may want to configure your network
    ↪ settings in
# /etc/libvirt/<hypervisor>/networks/
ip = 192.168.50.121
```

```
# (Optional) Specify tags to display
# Tags may be used to specify on which guest machines a sample
    ↪ should be run
# tags = windows_xp_sp3,acrobat_reader_6

# (Optional) Specify the snapshot name to use. If you do not
    ↪ specify a snapshot
# name, the KVM MachineManager will use the current snapshot.
# Example (Snapshot1 is the snapshot name):
snapshot = VNCCONPASS
#snapshot = SINDEFENDER
# (Optional) Specify the name of the network interface that should
    ↪  be used
# when dumping network traffic from this machine with tcpdump. If
    ↪ specified,
# overrides the default interface specified in auxiliary.conf
# Example (virbr0 is the interface name):
interface = virbr1

# (Optional) Specify the IP of the Result Server, as your virtual
    ↪ machine sees it.
# The Result Server will always bind to the address and port
    ↪ specified in cuckoo.conf,
# however you could set up your virtual network to use NAT/PAT, so
    ↪  you can specify here
# the IP address for the Result Server as your machine sees it. If
    ↪  you don't specify an
# address here, the machine will use the default value from cuckoo
    ↪ .conf.
# NOTE: if you set this option you have to set result server IP to
    ↪  0.0.0.0 in cuckoo.conf.
# Example:
# resultserver_ip = 192.168.50.1

# (Optional) Specify the port for the Result Server, as your
    ↪ virtual machine sees it.
# The Result Server will always bind to the address and port
    ↪ specified in cuckoo.conf,
# however you could set up your virtual network to use NAT/PAT, so
    ↪  you can specify here
# the port for the Result Server as your machine sees it. If you
    ↪ don't specify a port
# here, the machine will use the default value from cuckoo.conf.
# Example:
# resultserver_port = 2042

# Set the machine architecture
```

```
# Required to auto select proper machine architecture for sample
# x64 or x86
arch = x64
```

- */opt/CAPEv2/custom/conf/routing.conf*

```
[routing]
# Default network routing mode; "none", "internet", or "vpn_name".
# In none mode we don't do any special routing - the VM doesn't
    ↪ have any
# network access (this has been the default actually for quite a
    ↪ while).
# In internet mode by default all the VMs will be routed through
    ↪ the network
# interface configured below (the "dirty line").
# And in VPN mode by default the VMs will be routed through the
    ↪ VPN identified
# by the given name of the VPN.
# Note that just like enabling VPN configuration setting this
    ↪ option to
# anything other than "none" requires one to run utils/rooter.py
    ↪ as root next
# to the CAPE instance (as it's required for setting up the
    ↪ routing).
route = internet

# Network interface that allows a VM to connect to the entire
    ↪ internet, the
# "dirty line" so to say. Note that, just like with the VPNs, this
    ↪  will allow
# malicious traffic through your network. So think twice before
    ↪ enabling it.
# (For example, to route all VMs through eth0 by default: "
    ↪ internet = eth0").
internet = ens18

# Routing table name/id for "dirty line" interface. If "dirty line
    ↪ " is
# also default gateway in the system you can leave "main" value.
    ↪ Otherwise add
# new routing table by adding "<id> <name>" line to /etc/iproute2/
    ↪ rt_tables
# (e.g., "200 eth0"). ID and name must be unique across the system
    ↪  (refer to
# /etc/iproute2/rt_tables for existing names and IDs).
rt_table = main
```

```
# When using "dirty line", you can reject forwarding to a certain
    ↪ network segment.
# For example, a request targeting 192.168.12.1/24,172.16.22.1/24
    ↪ will not be
# forwarded, but will be rejected:
# "reject_segments = 192.168.12.1/24,172.16.22.1/24"
reject_segments = none

# When ussing "dirty line", you can reject guest access a certain
    ↪ port.
# For example, a request targeting host's port 8000 and 8080 will
    ↪ be rejected:
# "reject_hostports = 8000,8080"
reject_hostports = none

# To route traffic through multiple network interfaces CAPE uses
# Policy Routing with separate routing table for each output
    ↪ interface
# (VPN or "dirty line"). If this option is enabled CAPE on start
    ↪ will try
# to automatically initialise routing tables by copying routing
    ↪ entries from
# main routing table to the new routing tables. Depending on your
    ↪ network/vpn
# configuration this might not be sufficient. In such case you
    ↪ would need to
# initialise routing tables manually. Note that enabling this
    ↪ option won't
# affect main routing table.
auto_rt = no

# The drop route basically drops any outgoing network (except for
    ↪ CAPE
# traffic) whereas the regular none route still allows a VM to
    ↪ access its own
# subnet (e.g., 192.168.122.1/24). It is disabled by default as it
    ↪  does require
# the optional rooter to run (unlike the none route, where
    ↪ literally nothing
# happens). One can either explicitly enable the drop route or if
    ↪ the rooter
# is enabled anyway, it is automatically enabled.
drop = no

# Should check if the inteface is up
verify_interface = yes

[inetsim]
```

```
# Inetsim quick deploy, chose your vm manager if is not kvm
# wget https://googledrive.com/host/0B6fULLT_NpxMQ1Rrb1drdW42SkE/
    ↪ remnux-6.0-ova-public.ova
# tar xvf remnux-6.0-ova-public.ova
# qemu-img convert -O qcow2 REMnuxV6-disk1.vmdk remnux.qcow2

enabled = no
server = 192.168.1.2
dnsport = 53
interface = virbr1
# Redirect TCP ports (should we also support UDP?). If specified,
    ↪ this should
# represent whitespace-separated src:dst pairs. E.g., "80:8080
    ↪ 443:8080" will
# redirect all 80/443 traffic to 8080 on the specified InetSim
    ↪ host.
# Source port range redirection is also supported. E.g.,
    ↪ "996-2041:80" will
# redirect all traffic directed at ports between 996 and 2041
    ↪ inclusive to port 80
# on the specified InetSim host.
ports =


[tor]
enabled = no
dnsport = 5353
proxyport = 9040
interface = virbr1

[vpn]
# By default we disable VPN support as it requires running utils/
    ↪ rooter.py as
# root next to cuckoo.py (which should run as regular user).
enabled = yes

# select one of the configured vpns randomly
random_vpn = no

# Comma-separated list of the available VPNs.
vpns = vpn0

[vpn0]
# Name of this VPN. The name is represented by the filepath to the
# configuration file, e.g., cuckoo would represent /etc/openvpn/
    ↪ cuckoo.conf
# Note that you can't assign the names "none" and "internet" as
    ↪ those would
```

```
# conflict with the routing section in cuckoo.conf.
name = vpn0

# The description of this VPN which will be displayed in the web
    ↪ interface.
# Can be used to for example describe the country where this VPN
    ↪ ends up.
description = openvpn_tunnel

# The tun device hardcoded for this VPN. Each VPN *must* be
    ↪ configured to use
# a hardcoded/persistent tun device by explicitly adding the line
    ↪ "dev tunX"
# to its configuration (e.g., /etc/openvpn/vpn1.conf) where X in
    ↪ tunX is a
# unique number between 0 and your lucky number of choice.
interface = tun0

# Routing table name/id for this VPN. If table name is used it *
    ↪ must* be
# added to /etc/iproute2/rt_tables as "<id> <name>" line (e.g.,
    ↪ "201 tun0").
# ID and name must be unique across the system (refer /etc/
    ↪ iproute2/rt_tables
# for existing names and IDs).
rt_table = tun0


[socks5]
# By default we disable socks5 support as it requires running
    ↪ utils/rooter.py as
# root next to cuckoo.py (which should run as regular user).
enabled = no

# select one of the configured socks5 proxies randomly
random_socks5 = no

# Comma-separated list of the available proxies.
proxies = socks_ch

[socks_ch]
name = ch_socks
description = ch_socks
proxyport = 5008
dnsport = 10053
```

- */opt/CAPEv2/custom/conf/processing.conf*

```
# Enable or disable the available processing modules [on/off].
# If you add a custom processing module to your Cuckoo setup, you
    ↪ have to add
# a dedicated entry in this file, or it won't be executed.
# You can also add additional options under the section of your
    ↪ module and
# they will be available in your Python class.

# Requires dependencies of software in vm as by:
# https://www.fireeye.com/blog/threat-research/2016/02/
    ↪ greater_visibilityt.html
# Windows 7 SP1, .NET at least 4.5, powershell 5 preferly over v4
# KB3109118 - Script block logging back port update for WMF4
# x64 - https://cuckoo.sh/vmcloak/Windows6.1-KB3109118-v4-x64.msu
# x32 - https://cuckoo.sh/vmcloak/Windows6.1-KB3109118-v4-x86.msu
# KB2819745 - WMF 4 (Windows Management Framework version 4)
    ↪ update for Windows 7
# x64 - https://cuckoo.sh/vmcloak/Windows6.1-KB2819745-x64-
    ↪ MultiPkg.msu
# x32 - https://cuckoo.sh/vmcloak/Windows6.1-KB2819745-x86-
    ↪ MultiPkg.msu
# KB3191566 - https://www.microsoft.com/en-us/download/details.
    ↪ aspx?id=54616
# You should create following registry entries
# reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\
    ↪ PowerShell\ModuleLogging\ModuleNames" /v * /t REG_SZ /d * /f
    ↪  /reg:64
# reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\
    ↪ PowerShell\ScriptBlockLogging" /v EnableScriptBlockLogging /
    ↪ t REG_DWORD /d 00000001 /f /reg:64
# reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\
    ↪ PowerShell\Transcription" /v EnableTranscripting /t
    ↪ REG_DWORD /d 00000001 /f /reg:64
# reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\
    ↪ PowerShell\Transcription" /v OutputDirectory /t REG_SZ /d C
    ↪ :\PSTranscipts /f /reg:64
# reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\
    ↪ PowerShell\Transcription" /v EnableInvocationHeader /t
    ↪ REG_DWORD /d 00000001 /f /reg:64

# exclude files that doesn't match safe extension and ignore their
    ↪  files from processing inside of other modules like CAPE.py
[antiransomware]
enabled = no
# ignore all files with extension found more than X
skip_number = 30
```

```
[curtain]
enabled = no

[sysmon]
enabled = no

[analysisinfo]
enabled = yes

# FLARE capa -> to update rules utils/community.py -cr
# install -> cd /tmp && git clone --recurse-submodules https://
    ↪ github.com/fireeye/capa.git && cd capa && git submodule
    ↪ update --init rules && python -m pip3 install .
[flare_capa]
enabled = no
# Generate it always or generate on demand only(user need to click
    ↪  button to generate it), still should be enabled to use this
    ↪  feature on demand
on_demand = no
# Analyze binary payloads
static = no
# Analyze CAPE payloads
cape = no
# Analyze ProcDump
procdump = no

[decompression]
enabled = no

[dumptls]
enabled = no

[behavior]
enabled = yes
# Toggle specific modules within the BehaviorAnalysis class
anomaly = yes
processtree = yes
summary = yes
enhanced = yes
encryptedbuffers = yes
# Should the server use a compressed version of behavioural logs?
    ↪ This helps
# in saving space in Mongo, accelerates searchs and reduce the
    ↪ size of the
# final JSON report.
loop_detection = no
# The number of calls per process to process. 0 switches the limit
    ↪  off.
```

```
# 10000 api calls should be processed in less than 2 minutes
analysis_call_limit = 0
# Use ram to boost processing speed. You will need more than 20GB
    ↪ of RAM for this feature.
# Please read "performance" section in the documentation.
ram_boost = no
# https://capev2.readthedocs.io/en/latest/usage/
    ↪ patterns_replacement.html
replace_patterns = no

[debug]
enabled = yes

[detections]
enabled = yes
# Signatures
behavior = yes
yara = yes
suricata = yes
virustotal = yes
clamav = no

[dropped]
enabled = yes
# Amount of text to carve from plaintext files (bytes)
buffer = 16384

# It is intended in CAPE for procdump to replace procmemory...
[procdump]
enabled = yes

# ... but this mechanism may still be switched on
[procmemory]
enabled = yes
strings = yes

[procmon]
enabled = no

[memory]
enabled = no

[usage]
enabled = no

[network]
enabled = yes
sort_pcap = yes
```

```
# DNS whitelisting to ignore domains/IPs configured in network.py
# This should be disabled when utilizing InetSim/Remnux as we end
    ↪ up resolving
# the IP from fakedns which would then remove all domains
    ↪ associated with that
# resolved IP
dnswhitelist = yes
# additional entries
dnswhitelist_file = extra/whitelist_domains.txt
ipwhitelist = yes
ipwhitelist_file = extra/whitelist_ips.txt

# Should the server use a compressed version of behavioural logs?
    ↪ This helps
# in saving space in Mongo, accelerates searchs and reduce the
    ↪ size of the
# final JSON report.
[loop_detection]
enabled = no

[static]
enabled = yes
# Scan for UserDB.TXT signature matches
userdb_signature = no
# Enable a WHOIS lookup for the target domain of a URL analyses
whois = yes
# Process files not bigger than value below in Mb. We saw that
    ↪ after 90Mb it has biggest delay
max_file_size = 90

[strings]
enabled = yes
on_demand = no
nullterminated_only = no
minchars = 5

[trid]
# Specify the path to the trid binary to use for static analysis.
enabled = no
identifier = data/trid/trid
definitions = data/trid/triddefs.trd

[die]
# Detect it Easy
enabled = no
binary = /usr/bin/diec

[targetinfo]
```

```
enabled = yes

[virustotal]
enabled = yes
on_demand = no
timeout = 60
# remove empty detections
remove_empty = yes
# Add your VirusTotal API key here. The default API key, kindly
    ↪ provided
# by the VirusTotal team, should enable you with a sufficient
    ↪ throughput
# and while being shared with all our users, it shouldn't affect
    ↪ your use.
key =
    ↪ a0283a2c3d55728300d064874239b5346fb991317e8449fe43c902879d758088
    ↪
do_file_lookup = yes
do_url_lookup = yes
urlscrub = (^http:\/\/serw\.clicksor\.com\/redir\.php\?url=|&
    ↪ InjectedParam=.+$)

[suricata]
# Notes on getting this to work check install_suricata function:
# https://github.com/doomedraven/Tools/blob/master/Sandbox/cape2.
    ↪ sh

enabled = yes
#Runmode "cli" or "socket"
runmode = socket
#Outputfiles
# if evelog is specified, it will be used instead of the per-
    ↪ protocol log files
evelog = eve.json

# per-protocol log files
#
#alertlog = alert.json
#httplog = http.json
#tlslog = tls.json
#sshlog = ssh.json
#dnslog = dns.json

fileslog = files-json.log
filesdir = files
# Amount of text to carve from plaintext files (bytes)
buffer = 8192
#Used for creating an archive of extracted files
```

```
7zbin = /usr/bin/7z
zippass = infected
##Runmode "cli" options
bin = /usr/bin/suricata
conf = /etc/suricata/suricata.yaml
##Runmode "socket" Options
socket_file = /tmp/suricata-command.socket

[cif]
enabled = no
# url of CIF server
url = https://your-cif-server.com/api
# CIF API key
key = your-api-key-here
# time to wait for server to respond, in seconds
timeout = 60
# minimum confidence level of returned results:
# 25=not confident, 50=automated, 75=somewhat confident, 85=very
    ↪ confident, 95=certain
# defaults to 85
confidence = 85
# don't log queries by default, set to 'no' to log queries
nolog = yes
# max number of results per query
per_lookup_limit = 20
# max number of queries per analysis
per_analysis_limit = 200

[CAPE]
enabled = yes
# Ex targetinfo standalone module
targetinfo = yes
# Ex dropped standalone module
dropped = yes
# Ex procdump standalone module
procdump = yes
# Amount of text to carve from plaintext files (bytes)
buffer = 8192
# Process files not bigger than value below in Mb. We saw that
    ↪ after 90Mb it has biggest delay
max_file_size = 90
# Scan for UserDB.TXT signature matches
userdb_signature = no
# https://capev2.readthedocs.io/en/latest/usage/
    ↪ patterns_replacement.html
replace_patterns = no

# Deduplicate screenshots
```

```
# You need to install dependency ImageHash = "4.2.1" or newer
[deduplication]
#
# Available hashs functions:
# ahash: Average hash
# phash: Perceptual hash
# dhash: Difference hash
# whash-haar: Haar wavelet hash
# whash-db4: Daubechies wavelet hash
enabled = no
hashmethod = ahash

[vba2graph]
# Mac - brew install graphviz
# Ubuntu - sudo apt-get install graphviz
# Arch - sudo pacman -S graphviz+
# sudo pip3 install networkx>=2.1 graphviz>=0.8.4 pydot>=1.2.4
enabled = yes
on_demand = yes

# ja3 finger print db with descriptions
# https://github.com/trisulnsm/trisul-scripts/blob/master/lua/
    ↪ frontend_scripts/reassembly/ja3/prints/ja3fingerprint.json
[ja3]
ja3_path = data/ja3/ja3fingerprint.json

[maliciousmacrobot]
# https://maliciousmacrobot.readthedocs.io
# Install mmbot
# sudo pip3 install mmbot
# Create/Set required paths
# Populate benign_path and malicious_path with appropriate macro
    ↪ maldocs (try the tests/samples in the github)
# https://github.com/egaus/MaliciousMacroBot/tree/master/tests/
    ↪ samples
# Create modeldata.pickle with your maldocs (this does not append
    ↪ to the model, it overwrites it)
#
# mmb = MaliciousMacroBot(benign_path, malicious_path, model_path,
    ↪  retain_sample_contents=False)
# result = mmb.mmb_init_model(modelRebuild=True)
#
# Copy your model file and vocab.txt to your model_path
enabled = no
benign_path = /opt/cuckoo/data/mmbot/benign
malicious_path = /opt/cuckoo/data/mmbot/malicious
model_path = /opt/cuckoo/data/mmbot/model
```

```
[xlsdeobf]
# pip3 install git+https://github.com/DissectMalware/
    ↪ XLMMacroDeobfuscator.git
enabled = no
on_demand = no

[boxjs]
enabled = no
timeout = 60
url = http://your_super_box_js:9000

# Extractors
[mwcp]
enabled = yes
modules_path = modules/processing/parsers/mwcp/

[ratdecoders]
enabled = yes
modules_path = modules/processing/parsers/RATDecoders/

[malduck]
enabled = yes
modules_path = modules/processing/parsers/malduck/

[CAPE_extractors]
enabled = yes
# Must ends with /
modules_path = modules/processing/parsers/CAPE/

[reversinglabs]
enabled = no
url =
key =

[script_log_processing]
enabled = yes

# Dump PE's overlay info
[overlay]
enabled = no

[floss]
enabled = no
on_demand = yes
static_strings = no
stack_strings = yes
decoded_strings = yes
tight_strings = yes
```

```
min_length = 5
# Download FLOSS signatures from https://github.com/mandiant/flare
    ↪ -floss/tree/master/sigs
sigs_path = data/flare-signatures
```

- */opt/CAPEv2/custom/conf/reporting.conf*

```
# Enable or disable the available reporting modules [on/off].
# If you add a custom reporting module to your Cuckoo setup, you
    ↪ have to add
# a dedicated entry in this file, or it won't be executed.
# You can also add additional options under the section of your
    ↪ module and
# they will be available in your Python class.

[cents]
enabled = no
on_demand = no
# starting signature id for created Suricata rules
start_sid = 1000000

[mitre]
enabled = no

# https://github.com/geekscrapy/binGraph
# requires -> apt-get install python-tk
[bingraph]
enabled = yes
on_demand = yes
binary = yes
# geenrate bingraphs for cape/procdumps
cape = yes
procdump = yes

[pcap2cert]
enabled = yes

[litereport]
enabled = no
keys_to_copy = CAPE procdump info signatures dropped static target
    ↪  network shot malscore ttps
behavior_keys_to_copy = processtree summary

[jsondump]
enabled = yes
# use the c-optimized JSON encoder, requires fitting entire JSON
    ↪ results in memory
```

```
ram_boost = yes
indent = 4
encoding = latin-1

[reporthtml]
# required for the WSGI interface
enabled = no

[reporthtmlsummary]
# much smaller, faster report generation, omits API logs and is
    ↪ non-interactive
enabled = no

[reportpdf]
# Note that this requires reporthtmlsummary to be enabled above as
    ↪  well
enabled = no

[maec41]
enabled = no
mode = overview
processtree = true
output_handles = false
static = true
strings = true
virustotal = true
deduplicate = true

[maec5]
enabled = no

[mongodb]
enabled = yes
host = 127.0.0.1
port = 27017
db = cuckoo
# Set those values if you are using mongodb authentication
# username =
# password =
# authsource = cuckoo


# Automatically delete large dict values that exceed mongos 16MB
    ↪ limitation
# Note: This only deletes dict keys from data stored in MongoDB.
    ↪ You would
# still get the full dataset if you parsed the results dict in
    ↪ another
```

```
# reporting module or from the jsondump module.
fix_large_docs = yes

# Use ElasticSearch as the "database" which powers Django.
# NOTE: If this is enabled, MongoDB should not be enabled, unless
# search only option is set to yes. Then elastic search is only
    ↪ used for /search web page.
[elasticsearchdb]
enabled = no
searchonly = True
host = 192.168.153.2
port = 9200
# The report data is indexed in the form of {{index-yyyy.mm.dd}}
# so the below index configuration option is actually an index '
    ↪ prefix'.
index = cuckoo
username = elastic
password = tosdkA1AnO_VTWBcWa2A
use_ssl = True
verify_certs = /opt/extra_tfg/certs/http_ca.crt

[retention]
enabled = no
# run at most once every this many hours (unless reporting.conf is
    ↪  modified)
run_every = 6
# The amount of days old a task needs to be before deleting data
# Set a value to no to never delete it
memory = 14
procmemory = 62
pcap = 62
sortedpcap = 14
bsonlogs = 62
dropped = 62
screencaps = 62
reports = 62
mongo = 731
elastic = no

[syslog]
enabled = no
# IP of your syslog server/listener
host = x.x.x.x
# Port of your syslog server/listener
port = 514
# Protocol to send data over
protocol = tcp
# Store a logfile? [in reports directory]
```

```
logfile = yes
# if yes, what logname? [Default: syslog.txt]
logname = syslog.log

[moloch]
enabled = no
base = https://172.18.100.105:8005/
node = cuckoo3
capture = /data/moloch/bin/moloch-capture
captureconf = /data/moloch/etc/config.ini
user = admin
pass = admin
realm = Moloch

[resubmitexe]
enabled = no
resublimit = 5


[compression]
enabled = yes
zipmemdump = yes
zipmemstrings = yes
zipprocdump = yes
zipprocstrings = yes

[misp]
enabled = no
apikey =
url =
#Make event published after creation?
published = no
# minimal malscore, by default all
min_malscore = 0
# by default 5 threads
threads =
# this will retrieve information for iocs
# and activate misp report download from webgui
extend_context = no
# upload iocs from cuckoo to MISP
upload_iocs = no
distribution = 0
threat_level_id = 2
analysis = 2
# Sections to report
# Analysis ID will be appended, change
title = Iocs from cuckoo analysis:
network = no
```

```
ids_files = no
dropped = no
registry = no
mutexes = no

[callback]
enabled = no
# will send as post data {"task_id":X}
# can be coma separated urls
url = http://IP/callback

[distributed]
enabled = no
# save results on master, not analyze binaries
master_storage_only = no
remove_task_on_worker = no
failed_clean = no
# distributed cuckoo database, to store nodes and tasks info
db = sqlite:///dist.db
# tried before declare node as dead and deactivate it
dead_count = 5
# number of threads witch will retrieve files see api.py for dist
dist_threads = 4
# Tags breaks distributed logic,
# don't activate it till you really know what you do
enable_tags = no
# Fetch data over REST API or NFS, see docs how to setup NFS
nfs = no


# CAPE auto-submission of detected samples
# with a selected CAPE package.
[submitCAPE]
enabled = yes
# check root keyword, if found not resubmit, allows custom
    ↪ extractions
keyword = tr_extractor
# distributed CAPE, only enable on clients
distributed = no
# rest api url to master node
url = http://IP:8000/api/tasks/create/file/

# Compress results including CAPE output
# to help avoid reaching the hard 16MB MongoDB limit.
[compressresults]
enabled = yes

[tmpfsclean]
```

```
enabled = no
key = tr_extractor

# This calls the specified command, pointing it at the report.json
    ↪ as
# well as setting $ENV{CAPE_TASK_ID} to the task ID of the run in
    ↪ question.
#
[zexecreport]
enabled=no
command=/foo/bar.pl

# run statistics, this may take more times.
[runstatistics]
enabled = no

[malheur]
enabled = no

[geolookup]
enabled = yes
```

- */opt/CAPEv2/custom/conf/selfextract.conf*

```
# This config is to be able to enable/disable things like MSI/NSIS
    ↪ /UnAutoIt etc

[general]
pefiles = yes
dotnet = yes
office = yes
java = yes
pdf = yes
lnk = yes
windows_script = yes
elf = yes
hwp = yes

# Number of workers for pool to run them in parallel
max_workers = 6

# sudo apt install msitools
[msi_extract]
enabled = yes
binary = /usr/bin/msiextract
timeout = 60
```

```
[kixtart_extract]
enabled = yes
timeout = 60

[vbe_extract]
enabled = yes
timeout = 60

[batch_extract]
enabled = yes
timeout = 60

# cd /opt/CAPEv2/data/
# snap install go --classic
# git clone https://github.com/x0r19x91/UnAutoIt && cd UnAutoIt
# GOOS="linux" GOARCH="amd64" go build -o UnAutoIt
[UnAutoIt_extract]
enabled = yes
binary = data/UnAutoIt/UnAutoIt
timeout = 60

[RarSFX_extract]
enabled = yes
timeout = 60

# apt install upx-ucl
[UPX_unpack]
enabled = yes
timeout = 60

# Nsis, 7Zip SFX, etc
[SevenZip_unpack]
enabled = yes
timeout = 60

# sudo apt install innoextract
[Inno_extract]
enabled = yes
binary = /usr/bin/innoextract
timeout = 60

# https://github.com/mstrobel/procyon/releases
[procyon]
enabled = yes
binary = data/procyon.jar
timeout = 60

# sudo apt install de4dot
```

```
[de4dot_deobfuscate]
enabled = yes
binary = /usr/bin/de4dot
extra_args =
timeout = 60

# https://github.com/SychicBoy/NETReactorSlayer/releases
[eziriz_deobfuscate]
enabled = yes
binary = data/NETReactorSlayer.CLI
extra_args = --no-pause True
timeout = 60

[office_one]
enabled = yes
timeout = 60
```

- */opt/CAPEv2/custom/conf/auxiliary.conf*

```
# Requires dependencies of software in vm as by:
# https://www.fireeye.com/blog/threat-research/2016/02/
    ↪ greater_visibilityt.html
# Windows 7 SP1, .NET at least 4.5, powershell 5 preferly over v4
# KB3109118 - Script block logging back port update for WMF4
# x64 - https://cuckoo.sh/vmcloak/Windows6.1-KB3109118-v4-x64.msu
# x32 - https://cuckoo.sh/vmcloak/Windows6.1-KB3109118-v4-x86.msu
# KB2819745 - WMF 4 (Windows Management Framework version 4)
    ↪ update for Windows 7
# x64 - https://cuckoo.sh/vmcloak/Windows6.1-KB2819745-x64-
    ↪ MultiPkg.msu
# x32 - https://cuckoo.sh/vmcloak/Windows6.1-KB2819745-x86-
    ↪ MultiPkg.msu
# KB3191566 - https://www.microsoft.com/en-us/download/details.
    ↪ aspx?id=54616
# You should create following registry entries
# reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\
    ↪ PowerShell\ModuleLogging\ModuleNames" /v * /t REG_SZ /d * /f
    ↪  /reg:64
# reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\
    ↪ PowerShell\ScriptBlockLogging" /v EnableScriptBlockLogging /
    ↪ t REG_DWORD /d 00000001 /f /reg:64
# reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\
    ↪ PowerShell\Transcription" /v EnableTranscripting /t
    ↪ REG_DWORD /d 00000001 /f /reg:64
# reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\
    ↪ PowerShell\Transcription" /v OutputDirectory /t REG_SZ /d C
    ↪ :\PSTranscipts /f /reg:64
```

```
# reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\
    ↪ PowerShell\Transcription" /v EnableInvocationHeader /t
    ↪ REG_DWORD /d 00000001 /f /reg:64

# Modules to be enabled or not inside of the VM
[auxiliary_modules]
browser = yes
curtain = no
digisig = yes
disguise = yes
evtx = no
human_windows = yes
human_linux = no
procmon = no
screenshots_windows = yes
screenshots_linux = no
sysmon = no
tlsdump = yes
usage = no
file_pickup = no
permissions = no
pre_script = no
during_script = no
stap = no
filecollector = yes
# This is only useful in case you use KVM's dnsmasq. You need to
    ↪ change your range inside of analyzer/windows/modules/
    ↪ auxiliary/disguise.py. Disguise must be enabled
windows_static_route = no

[sniffer]
# Enable or disable the use of an external sniffer (tcpdump) [yes/
    ↪ no].
enabled = yes

# enable remote tcpdump support
remote = no
host = root@192.168.50.1

# Specify the path to your local installation of tcpdump. Make
    ↪ sure this
# path is correct.
tcpdump = /usr/bin/tcpdump

# Specify the network interface name on which tcpdump should
    ↪ monitor the
# traffic. Make sure the interface is active.
interface = virbr1
```

```
# Specify a Berkeley packet filter to pass to tcpdump.
bpf = not arp

[gateways]
#RTR1 = 192.168.153.1
#RTR2 = 192.168.153.5
#INETSIM = 192.168.153.14

[virustotaldl]
# adds an option in the web interface to upload samples via
    ↪ VirusTotal
# downloads for a comma-separated list of MD5/SHA1/SHA256 hashes
enabled = no
# note that unlike the VirusTotal processing module, the key
    ↪ required
# here is a Intelligence API key, not a Public API key
#dlintelkey = SomeKeyWithDLAccess
dlpath = /tmp/
```

- */etc/nginx/sites-available/capesite*

```
map $http_upgrade $connection_upgrade {
    default upgrade;
    '' close;
}
upstream nodeserver1 {
    # CAPE
    #server 192.168.153.5:8000;
    server 127.0.0.1:8000;
}
upstream nodeserver2 {
    # guac-session
    server 127.0.0.1:8008;
}
server {
    listen 192.168.153.5:3333;
    client_max_body_size 1g;
    #proxy_pass_header X-CSRFToken;
    location / {

        proxy_pass http://nodeserver1;
        #proxy_set_header Host $host;
        proxy_set_header Host $http_host;
        proxy_set_header X-Remote-User $remote_user;
        proxy_set_header X-Real-IP $remote_addr;
```

```
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for
            ↪ ;
    }
    location /static/ {
        alias /opt/CAPEv2/web/static/;
    }
    location /guac {
        proxy_pass http://nodeserver2;
        proxy_set_header Host $host;
        #proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_buffering off;
        proxy_http_version 1.1;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for
            ↪ ;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $http_connection;
        client_max_body_size 1g;
        #access_log off;
    }
    location /guac/playback/recfile {
        alias /var/www/guacrecordings/;
        autoindex on;
        autoindex_exact_size off;
        autoindex_localtime on;
    }
}
```

- */etc/nginx/sites-available/default*

```
##
# You should look at the following URL's in order to grasp a solid
    ↪ understanding
# of Nginx configuration files in order to fully unleash the power
    ↪ of Nginx.
# https://www.nginx.com/resources/wiki/start/
# https://www.nginx.com/resources/wiki/start/topics/tutorials/
    ↪ config_pitfalls/
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from sites-
    ↪ enabled/ and
# leave it as reference inside of sites-available where it will
    ↪ continue to be
# updated by the nginx packaging team.
#
# This file will automatically load configuration files provided
```

```
    ↪ by other
# applications, such as Drupal or Wordpress. These applications
    ↪ will be made
# available underneath a path with that package name, such as /
    ↪ drupal8.
#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed
    ↪ examples.
##

# Default server configuration
#
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        # SSL configuration
        #
        # listen 443 ssl default_server;
        # listen [::]:443 ssl default_server;
        #
        # Note: You should disable gzip for SSL traffic.
        # See: https://bugs.debian.org/773332
        #
        # Read up on ssl_ciphers to ensure a secure configuration.
        # See: https://bugs.debian.org/765782
        #
        # Self signed certs generated by the ssl-cert package
        # Don't use them in a production server!
        #
        # include snippets/snakeoil.conf;

        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;

        server_name _;

        #location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
        # try_files $uri $uri/ =404;
        #}

        location /static/ {
                alias /opt/CAPEv2/web/static/;
        }
```

```
        location /guac {
                proxy_pass http://nodeserver2;
                proxy_set_header Host $host;
                #proxy_set_header Host $http_host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_buffering off;
                proxy_http_version 1.1;
                proxy_set_header X-Forwarded-For
                    ↪ $proxy_add_x_forwarded_for;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection $http_connection;
                client_max_body_size 1g;
                #access_log off;
        }

        location /guac/playback/recfile {
                alias /var/www/guacrecordings/;
                autoindex on;
                autoindex_exact_size off;
                autoindex_localtime on;
        }

        # pass PHP scripts to FastCGI server
        #
        #location ~ \.php$ {
        # include snippets/fastcgi-php.conf;
        #
        # # With php-fpm (or other unix sockets):
        # fastcgi_pass unix:/run/php/php7.4-fpm.sock;
        # # With php-cgi (or other tcp sockets):
        # fastcgi_pass 127.0.0.1:9000;
        #}

        # deny access to .htaccess files, if Apache's document root
        # concurs with nginx's one
        #
        #location ~ /\.ht {
        # deny all;
        #}
}


# Virtual Host configuration for example.com
#
# You can move that to a different file under sites-available/ and
    ↪ symlink that
# to sites-enabled/ to enable it.
```

```
#
#server {
# listen 80;
# listen [::]:80;
#
# server_name example.com;
#
# root /var/www/example.com;
# index index.html;
#
# location / {
#  try_files $uri $uri/ =404;
# }
#}
```

- */etc/nginx/sites-available/capesite*

```
map $http_upgrade $connection_upgrade {
    default upgrade;
    '' close;
}
upstream nodeserver1 {
    # CAPE
    #server 192.168.153.5:8000;
    server 127.0.0.1:8000;
}
upstream nodeserver2 {
    # guac-session
    server 127.0.0.1:8008;
}
server {
    listen 192.168.153.5:3333;
    client_max_body_size 1g;
    #proxy_pass_header X-CSRFToken;
    location / {

        proxy_pass http://nodeserver1;
        #proxy_set_header Host $host;
        proxy_set_header Host $http_host;
        proxy_set_header X-Remote-User $remote_user;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for
            ↪ ;
    }
    location /static/ {
        alias /opt/CAPEv2/web/static/;
```

```
    }
    location /guac {
        proxy_pass http://nodeserver2;
        proxy_set_header Host $host;
        #proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_buffering off;
        proxy_http_version 1.1;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for
            ↪ ;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $http_connection;
        client_max_body_size 1g;
        #access_log off;
    }
    location /guac/playback/recfile {
        alias /var/www/guacrecordings/;
        autoindex on;
        autoindex_exact_size off;
        autoindex_localtime on;
    }
}
```

- */etc/libvirt/qemu/win10.xml*

```
<!--
WARNING: THIS IS AN AUTO-GENERATED FILE. CHANGES TO IT ARE LIKELY
    ↪ TO BE
OVERWRITTEN AND LOST. Changes to this xml configuration should be
    ↪ made using:
  virsh edit win10
or other application using the libvirt API.
-->

<domain type='kvm' xmlns:qemu='http://libvirt.org/schemas/domain/
    ↪ qemu/1.0'>
  <name>win10</name>
  <uuid>7617786b-7418-4029-a4ca-1930beeebb5d</uuid>
  <metadata>
    <libosinfo:libosinfo xmlns:libosinfo="http://libosinfo.org/
        ↪ xmlns/libvirt/domain/1.0">
      <libosinfo:os id="http://microsoft.com/win/10"/>
    </libosinfo:libosinfo>
  </metadata>
  <memory unit='KiB'>8388608</memory>
  <currentMemory unit='KiB'>8388608</currentMemory>
  <vcpu placement='static'>2</vcpu>
```

```
<sysinfo type='smbios'>
  <bios>
    <entry name='vendor'>SeaBIOS</entry>
    <entry name='version'>07.08</entry>
    <entry name='date'>12/25/2012</entry>
    <entry name='release'>2.8</entry>
  </bios>
  <system>
    <entry name='manufacturer'>ASUSTeK COMPUTER INC.</entry>
    <entry name='product'>CG8270</entry>
    <entry name='version'>Rev X.0x</entry>
    <entry name='serial'>MT7023012300339</entry>
    <entry name='uuid'>7617786b-7418-4029-a4ca-1930beeebb5d</
       ↪ entry>
    <entry name='sku'>Not Specified</entry>
    <entry name='family'>ASUS</entry>
  </system>
</sysinfo>
<os>
  <type arch='x86_64' machine='pc-q35-7.1'>hvm</type>
  <boot dev='hd'/>
</os>
<features>
  <acpi/>
  <apic/>
  <hyperv mode='custom'>
    <relaxed state='on'/>
    <vapic state='on'/>
    <spinlocks state='on' retries='8191'/>
  </hyperv>
  <vmport state='off'/>
</features>
<cpu mode='custom' match='exact' check='partial'>
  <model fallback='allow'>SandyBridge</model>
  <topology sockets='1' dies='1' cores='2' threads='1'/>
</cpu>
<clock offset='localtime'>
  <timer name='rtc' tickpolicy='catchup'/>
  <timer name='pit' tickpolicy='delay'/>
  <timer name='hpet' present='no'/>
  <timer name='hypervclock' present='yes'/>
</clock>
<on_poweroff>destroy</on_poweroff>
<on_reboot>restart</on_reboot>
<on_crash>destroy</on_crash>
<pm>
  <suspend-to-mem enabled='no'/>
  <suspend-to-disk enabled='no'/>
```

```
</pm>
<devices>
  <emulator>/usr/bin/qemu-system-x86_64</emulator>
  <disk type='file' device='disk'>
    <driver name='qemu' type='qcow2' cache='none' discard='unmap
       ↪ '/>
    <source file='/opt/VMs/win10.qcow2'/>
    <target dev='sda' bus='sata'/>
    <address type='drive' controller='0' bus='0' target='0' unit
       ↪ ='0'/>
  </disk>
  <controller type='usb' index='0' model='qemu-xhci' ports='15'>
    <address type='pci' domain='0x0000' bus='0x02' slot='0x00'
       ↪ function='0x0'/>
  </controller>
  <controller type='pci' index='0' model='pcie-root'/>
  <controller type='pci' index='1' model='pcie-root-port'>
    <model name='pcie-root-port'/>
    <target chassis='1' port='0x10'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x02'
       ↪ function='0x0' multifunction='on'/>
  </controller>
  <controller type='pci' index='2' model='pcie-root-port'>
    <model name='pcie-root-port'/>
    <target chassis='2' port='0x11'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x02'
       ↪ function='0x1'/>
  </controller>
  <controller type='pci' index='3' model='pcie-root-port'>
    <model name='pcie-root-port'/>
    <target chassis='3' port='0x12'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x02'
       ↪ function='0x2'/>
  </controller>
  <controller type='pci' index='4' model='pcie-root-port'>
    <model name='pcie-root-port'/>
    <target chassis='4' port='0x13'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x02'
       ↪ function='0x3'/>
  </controller>
  <controller type='pci' index='5' model='pcie-root-port'>
    <model name='pcie-root-port'/>
    <target chassis='5' port='0x14'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x02'
       ↪ function='0x4'/>
  </controller>
  <controller type='pci' index='6' model='pcie-root-port'>
    <model name='pcie-root-port'/>
```

```
  <target chassis='6' port='0x15'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x02'
     ↪ function='0x5'/>
</controller>
<controller type='pci' index='7' model='pcie-root-port'>
  <model name='pcie-root-port'/>
  <target chassis='7' port='0x16'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x02'
     ↪ function='0x6'/>
</controller>
<controller type='pci' index='8' model='pcie-root-port'>
  <model name='pcie-root-port'/>
  <target chassis='8' port='0x17'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x02'
     ↪ function='0x7'/>
</controller>
<controller type='pci' index='9' model='pcie-root-port'>
  <model name='pcie-root-port'/>
  <target chassis='9' port='0x18'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03'
     ↪ function='0x0' multifunction='on'/>
</controller>
<controller type='pci' index='10' model='pcie-root-port'>
  <model name='pcie-root-port'/>
  <target chassis='10' port='0x19'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03'
     ↪ function='0x1'/>
</controller>
<controller type='pci' index='11' model='pcie-root-port'>
  <model name='pcie-root-port'/>
  <target chassis='11' port='0x1a'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03'
     ↪ function='0x2'/>
</controller>
<controller type='pci' index='12' model='pcie-root-port'>
  <model name='pcie-root-port'/>
  <target chassis='12' port='0x1b'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03'
     ↪ function='0x3'/>
</controller>
<controller type='pci' index='13' model='pcie-root-port'>
  <model name='pcie-root-port'/>
  <target chassis='13' port='0x1c'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03'
     ↪ function='0x4'/>
</controller>
<controller type='pci' index='14' model='pcie-root-port'>
  <model name='pcie-root-port'/>
```

```
  <target chassis='14' port='0x1d'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03'
     ↪ function='0x5'/>
</controller>
<controller type='sata' index='0'>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x1f'
     ↪ function='0x2'/>
</controller>
<controller type='virtio-serial' index='0'>
  <address type='pci' domain='0x0000' bus='0x03' slot='0x00'
     ↪ function='0x0'/>
</controller>
<interface type='network'>
  <mac address='2a:41:86:69:34:da'/>
  <source network='hostonly'/>
  <model type='e1000e'/>
  <address type='pci' domain='0x0000' bus='0x01' slot='0x00'
     ↪ function='0x0'/>
</interface>
<serial type='pty'>
  <target type='isa-serial' port='0'>
    <model name='isa-serial'/>
  </target>
</serial>
<console type='pty'>
  <target type='serial' port='0'/>
</console>
<channel type='spicevmc'>
  <target type='virtio' name='com.redhat.spice.0'/>
  <address type='virtio-serial' controller='0' bus='0' port
     ↪ ='1'/>
</channel>
<input type='tablet' bus='usb'>
  <address type='usb' bus='0' port='1'/>
</input>
<input type='mouse' bus='ps2'/>
<input type='keyboard' bus='ps2'/>
<graphics type='spice' autoport='yes'>
  <listen type='address'/>
</graphics>
<graphics type='vnc' port='-1' autoport='yes' listen='0.0.0.0'
   ↪ passwd='tfgadmin'>
  <listen type='address' address='0.0.0.0'/>
</graphics>
<sound model='ich9'>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x1b'
     ↪ function='0x0'/>
</sound>
```

```
    <audio id='1' type='spice'/>
    <video>
      <model type='qxl' ram='65536' vram='65536' vgamem='16384'
         ↪ heads='1' primary='yes'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x01'
         ↪ function='0x0'/>
    </video>
    <redirdev bus='usb' type='spicevmc'>
      <address type='usb' bus='0' port='2'/>
    </redirdev>
    <redirdev bus='usb' type='spicevmc'>
      <address type='usb' bus='0' port='3'/>
    </redirdev>
    <memballoon model='virtio'>
      <address type='pci' domain='0x0000' bus='0x04' slot='0x00'
         ↪ function='0x0'/>
    </memballoon>
  </devices>
  <qemu:commandline>
    <qemu:arg value='-cpu'/>
    <qemu:arg value='host,-hypervisor,kvm=off'/>
    <qemu:arg value='-L'/>
    <qemu:arg value='/usr/share/qemu/bios.bin'/>
  </qemu:commandline>
</domain>
```

- */etc/libvirt/qemu/win7.xml*

```
<!--
WARNING: THIS IS AN AUTO-GENERATED FILE. CHANGES TO IT ARE LIKELY
   ↪ TO BE
OVERWRITTEN AND LOST. Changes to this xml configuration should be
   ↪ made using:
  virsh edit win7
or other application using the libvirt API.
-->

<domain type='kvm' xmlns:qemu='http://libvirt.org/schemas/domain/
   ↪ qemu/1.0'>
  <name>win7</name>
  <uuid>9a01ddde-0eb1-4833-b76d-b517ae5bb623</uuid>
  <metadata>
    <libosinfo:libosinfo xmlns:libosinfo="http://libosinfo.org/
       ↪ xmlns/libvirt/domain/1.0">
      <libosinfo:os id="http://microsoft.com/win/7"/>
    </libosinfo:libosinfo>
  </metadata>
```

```
<memory unit='KiB'>4194304</memory>
<currentMemory unit='KiB'>4194304</currentMemory>
<vcpu placement='static'>2</vcpu>
<sysinfo type='smbios'>
  <bios>
    <entry name='vendor'>SeaBIOS</entry>
    <entry name='version'>rel-1.14.0-0-g155821a1990b-prebuilt.
       ↪ qemu.org</entry>
    <entry name='date'>04/01/2014</entry>
    <entry name='release'>2.8</entry>
  </bios>
  <system>
    <entry name='manufacturer'>QEMU</entry>
    <entry name='product'>Standard PC (i440FX + PIIX, 1996)</
       ↪ entry>
    <entry name='version'>pc-i440fx-5.2</entry>
    <entry name='serial'>Not Specified</entry>
    <entry name='uuid'>9a01ddde-0eb1-4833-b76d-b517ae5bb623</
       ↪ entry>
    <entry name='sku'>Not Specified</entry>
    <entry name='family'>Not Specified</entry>
  </system>
</sysinfo>
<os>
  <type arch='x86_64' machine='pc-i440fx-7.1'>hvm</type>
  <boot dev='hd'/>
</os>
<features>
  <acpi/>
  <apic/>
  <hyperv mode='custom'>
    <relaxed state='on'/>
    <vapic state='on'/>
    <spinlocks state='on' retries='8191'/>
  </hyperv>
  <vmport state='off'/>
</features>
<cpu mode='custom' match='exact' check='partial'>
  <model fallback='allow'>SandyBridge</model>
  <topology sockets='1' dies='1' cores='2' threads='1'/>
</cpu>
<clock offset='localtime'>
  <timer name='rtc' tickpolicy='catchup'/>
  <timer name='pit' tickpolicy='delay'/>
  <timer name='hpet' present='no'/>
  <timer name='hypervclock' present='yes'/>
</clock>
<on_poweroff>destroy</on_poweroff>
```

```
<on_reboot>restart</on_reboot>
<on_crash>destroy</on_crash>
<pm>
  <suspend-to-mem enabled='no'/>
  <suspend-to-disk enabled='no'/>
</pm>
<devices>
  <emulator>/usr/bin/qemu-system-x86_64</emulator>
  <disk type='file' device='disk'>
    <driver name='qemu' type='qcow2' cache='none' discard='unmap
       ↪ '/>
    <source file='/opt/VMs/win7.qcow2'/>
    <target dev='hda' bus='ide'/>
    <address type='drive' controller='0' bus='0' target='0' unit
       ↪ ='0'/>
  </disk>
  <controller type='usb' index='0' model='ich9-ehci1'>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x05'
       ↪ function='0x7'/>
  </controller>
  <controller type='usb' index='0' model='ich9-uhci1'>
    <master startport='0'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x05'
       ↪ function='0x0' multifunction='on'/>
  </controller>
  <controller type='usb' index='0' model='ich9-uhci2'>
    <master startport='2'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x05'
       ↪ function='0x1'/>
  </controller>
  <controller type='usb' index='0' model='ich9-uhci3'>
    <master startport='4'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x05'
       ↪ function='0x2'/>
  </controller>
  <controller type='pci' index='0' model='pci-root'/>
  <controller type='ide' index='0'>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x01'
       ↪ function='0x1'/>
  </controller>
  <controller type='virtio-serial' index='0'>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x06'
       ↪ function='0x0'/>
  </controller>
  <interface type='network'>
    <mac address='00:0d:60:0d:0d:a1'/>
    <source network='hostonly'/>
    <model type='e1000e'/>
```

```
    <address type='pci' domain='0x0000' bus='0x00' slot='0x03'
       ↪ function='0x0'/>
  </interface>
  <serial type='pty'>
    <target type='isa-serial' port='0'>
      <model name='isa-serial'/>
    </target>
  </serial>
  <console type='pty'>
    <target type='serial' port='0'/>
  </console>
  <channel type='spicevmc'>
    <target type='virtio' name='com.redhat.spice.0'/>
    <address type='virtio-serial' controller='0' bus='0' port
       ↪ ='1'/>
  </channel>
  <input type='tablet' bus='usb'>
    <address type='usb' bus='0' port='1'/>
  </input>
  <input type='mouse' bus='ps2'/>
  <input type='keyboard' bus='ps2'/>
  <graphics type='spice' autoport='yes'>
    <listen type='address'/>
  </graphics>
  <sound model='ich9'>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x04'
       ↪ function='0x0'/>
  </sound>
  <audio id='1' type='spice'/>
  <video>
    <model type='qxl' ram='65536' vram='65536' vgamem='16384'
       ↪ heads='1' primary='yes'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x02'
       ↪ function='0x0'/>
  </video>
  <redirdev bus='usb' type='spicevmc'>
    <address type='usb' bus='0' port='2'/>
  </redirdev>
  <redirdev bus='usb' type='spicevmc'>
    <address type='usb' bus='0' port='3'/>
  </redirdev>
  <memballoon model='virtio'>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x07'
       ↪ function='0x0'/>
  </memballoon>
</devices>
<qemu:commandline>
  <qemu:arg value='-cpu'/>
```

```
    <qemu:arg value='SandyBridge,-hypervisor,kvm=off'/>
    <qemu:arg value='-L'/>
    <qemu:arg value='/usr/share/qemu/bios.bin'/>
  </qemu:commandline>
</domain>
```

A.3. Instalación en *guest*

En ambos *guest*, se instalan programas de uso común en un ordenador personal (*WinRAR, AdobeReader, Mozilla...*). También se descarga una versión de *Python* de *32-bits*, puesto que es la soportada por *CAPEv2*. Adicionalmente se aplican las siguientes políticas tal y como explican en la documentación [2]:

Tanto en el equipo *Windows 7*, como en el equipo *Windows 10* hay que desactivar las utilidades *Windows Defender* y *Windows Update*. Además hay que asignarles una dirección estática, cambiando la configuración por defecto del adaptador de red de cada equipo.

Por último en ambas máquinas virtuales hay que poner en marcha el agente que se va a comunicar con *CAPEv2*, esto se realiza creando en la herramienta de *Windows, Tareas*, una nueva tarea que se lanza con cada inicio de sesión. Esta tarea se configura para que ejecute la utilidad proporcionada por el equipo de *CAPEv2, agent.py*. En este proyecto se ha modificado la extensión de este *script* a *.pyw*, para que se ejecute en un segundo plano.

En el equipo *Windows 10*, el servicio *Update*, se ha deshabilitado configurándolo tal y como se muestra en la figura A.1, ya que el método explicado en la documentación está obsoleto.

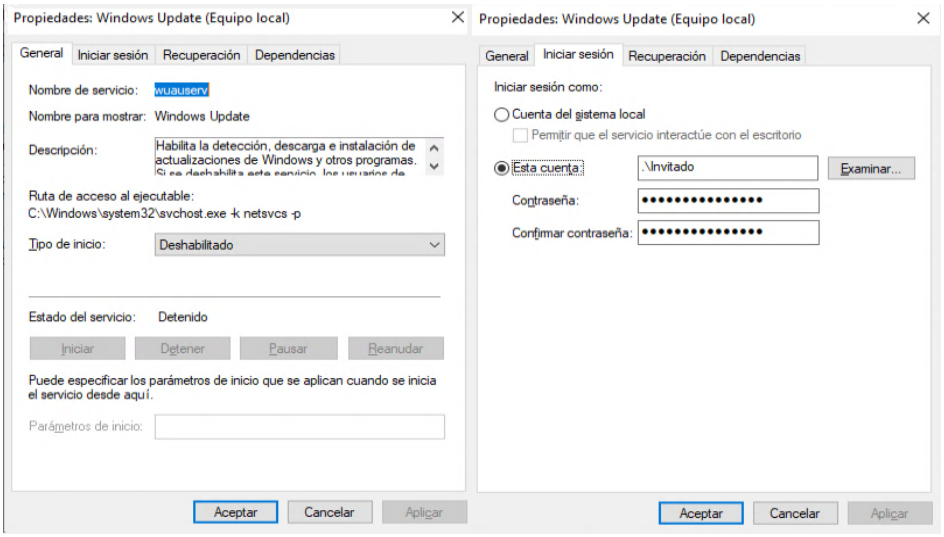

Figura A.1: Configuración del servicio *Windows Update*.

Anexos B

MONTAJE DE *VPN*

B.1. Instalación *router* virtual *Mikrotik*

La *VPN* desplegada sigue el esquema que se muestra en la figura B.1.

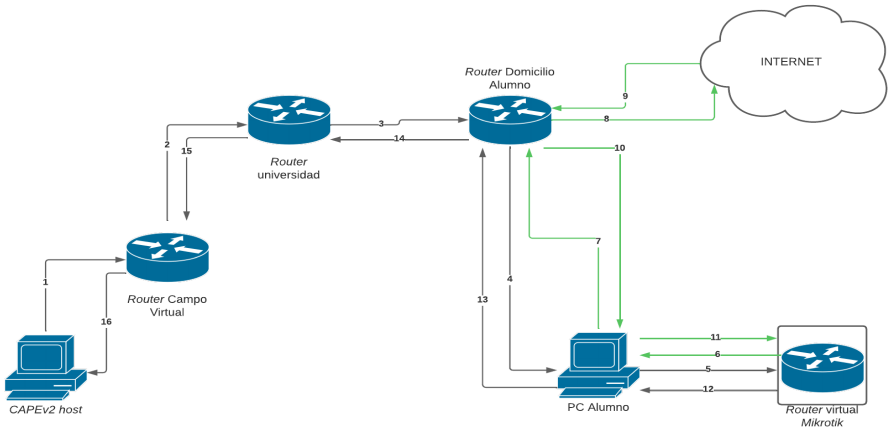

Figura B.1: Esquema que sigue el tráfico de un paquete en la *VPN* que se ha desplegado.

En el ordenador del alumno, haciendo uso de *VirtualBox*, se ha desplegado una máquina virtual con el sistema que proporciona en su página web *Mikrotik*. Tras instalar el *S.O*, se ha aplicado la licencia gratuita que proporciona *Mikrotik*. Esta licencia se consigue a través de una clave, proporcionada tras el registro de una cuenta en la página oficial de *Mikrotik*. [21]

La el *router* virtual se despliega en el ordenador del alumno, puesto que, el *router* del domicilio del alumno no tiene las funcionalidades necesarias para levantar un túnel *VPN*.

B.2. Configuración túnel *OVPN*

B.2.1. Configuración desde el *router Mikrotik*

Tras introducir la clave de la licencia gratuita, en la máquina virtual con el sistema operativo *Mikrotik*, se introducen las siguientes instrucciones en relación al túnel *OpenVPN* que se va a levantar: Instrucciones relacionadas con la generación y firma de certificados:

```
# Se generan certificados
/certificate
add name=CA common-name=myCa key-usage=key-cert-sign,crl-sign
add name=VPNserver common-name=server
add name=VPNclient common-name=client
# Firma de certificados
/certificate
sign CA ca-crl-host=IP_PUBLICA_ALUMNO name=myCa
sign VPNserver ca=myCa name=server
sign VPNclient ca=myCa name=client
# Marcar confianza en los certificados
/certificate
set myCa trusted=yes
set server trusted=yes
# Se exportan los certificados, con clave de seguridad, indicada como
# PASSWORD en este anexo
/certificate export-certificate myCa
/certificate export-certificate client1 export-passphrase=PASSWORD
```

Para levantar el túnel *OVPN* se aplican, los ejecutan las siguientes instrucciones en el *router* virtual:

```
/ip pool add name=ovpn-pool range=192.168.88.2-192.168.88.254
# Se crea en conexiones ppp el perfil VPNtfg indicando que el extremo
# del server tiene IP 192.168.88.1 y que los clientes se encuentran en
    ↪ el
# pool de direcciones 192.168.88.2-192.168.88.254 y se crea clave para
    ↪ client
/ppp profile add name=VPNtfg local-address=192.168.88.1 remote-address=
    ↪ ovpn-pool
/ppp secret
add name=client password=PASSWORD profile=VPNtfg
# Se le asigna el certificado previamente creado
/interface ovpn-server server set enabled=yes certificate=server
```

B.2.2. Configuración desde el equipo *CAPEv2 host*

Desde el *host* de *CAPEv2* se ejecutan las siguientes instrucciones, para iniciar la conexión *VPN* y para redirigir el tráfico correspondiente a este, en la tabla de rutas, con el identificador que ha sido especificado en el fichero de configuración *routing.conf*.

```
ip route add default via 192.168.88.1 proto static table 333
ip route add 192.168.50.0/24 dev virbr1 proto kernel scope link src
   ↪ 192.168.50.1 table 333
openvpn --config config_casa.ovpn
```

El fichero *config_casa.ovpn* es el siguiente:

```
dev tun
proto tcp-client
remote IP_PUBLICA_ALUMNO
port 1194
nobind
persist-key
persist-tun
tls-client
remote-cert-tls server
ca cert_export_CA.crt
cert cert_export_client.crt
key cert_export_client.key
verb 4
mute 10
cipher AES-256-CBC
auth SHA512
pull
auth-user-pass secret.auth
auth-nocache
```

Este fichero se encuentra en un directorio junto a los archivos *cert_export_CA.crt cert cert_export_client.crt key cert_export_client.key y secret.auth*, que han sido generados por el servidor *VPN*.

B.3. Configuración firewall

En el *firewall* del *router* virtual, se aplica un *MASQUERADE* a las conexiones provenientes de la VPN.

B.4. Configuración router de la operadora

Para que el tráfico, correspondiente a la *VPN*, pudiera comunicarse, a través del *router* de la operadora del domicilio del alumno, con el *router Mikrotik*, se ha hecho *DNAT* en el *router* de la operadora. Esto se ha realizado de tal modo que todo el tráfico, entrante por el puerto 1194, se ha redirigido al puerto 1194 de la dirección *IP* asociada al *router Mikrotik* virtual.

Anexos C

REPOSITORIOS DE MALWARE

A continuación se explican las distintas fuentes, utilizadas para la obtención de las distintas muestras de *malware*.

C.1. *TheZoo*

El repositorio *TheZoo* es una colección de *malware*, compartida en un repositorio público de *Github*, para fines didácticos. Algunas de las muestras que se han extraído de este repositorio corresponden a *WannaCry* y *Petya*, que son conocidos por su impacto global y sus técnicas de propagación avanzadas.

El *malware WannaCry* se propagó en mayo de 2017 y afectó a miles de sistemas en todo el mundo mediante la explotación de una vulnerabilidad en el protocolo *SMB* de *Windows*.

No es necesario ningún registro para acceder a estas.

C.2. *Malware Bazar*

Malware Bazar es un repositorio *online* donde se pueden encontrar y adquirir muestras de *malware* para propósitos de investigación y análisis.

Este repositorio contiene muestras recientes de *malware* y no es necesario ningún registro para acceder a estas.

C.3. *VirusShare*

VirusShare es un repositorio que proporciona una amplia colección de muestras de software malicioso. A través de esta plataforma, los investigadores y profesionales de seguridad pueden acceder a una gran cantidad de muestras de *malware* recopiladas de forma colaborativa.

Para acceder a esta plataforma es necesario enviar un correo a la dirección *Melissa97@virusshare.com*, indicando las razones por las que solicitas el acceso a su base de datos.

Anexos D

SOLUCIONES EN *PYTHON*

En el repositorio de *Github*, *https://github.com/MarioRLO/Client-Server-CAPEv2*, se encuentran todos los *scripts* desarrollados a lo largo de este proyecto, con la siguiente organización:

- En el directorio *Agent* se encuentran los *scripts* relativos al desarrollo del agente, preparado para los equipos que se desea monitorizar.

  Este directorio contiene adicionalmente una carpeta con nombre *old*. En este directorio se encuentran los diseños probados de manera previa al diseño final, en los que la transferencia de archivos se trataba de hacer con métodos alternativos al protocolo *SFTP*, sin éxito para archivos con tamaño del orden de *MegaBytes*.

- En el directorio *Report* se encuentra el módulo de reporte *geolookup.py*, cuyo funcionamiento se detalla en la sección 4.5.

- En el directorio *Server* se encuentran los *scripts* relativos al desarrollo del servidor preparado para recibir muestras, cuyo funcionamiento se desarrolla en el capítulo 4.

  Este directorio contiene adicionalmente una carpeta con nombre *old*. En este directorio se encuentran los diseños probados de manera previa al diseño final, en los que la recepción de archivos se trataba de hacer con métodos alternativos al protocolo *SFTP*, sin éxito para archivos con tamaño del orden de *MegaBytes*.

Estos *scripts* se han ejecutado con la versión 3.10.6 de *Python* y se han utilizado las siguientes librerías, en las versiones que se indican a continuación:

```
Package Version
------------ -------
bcrypt 3.2.0
cffi 1.15.1
```

```
cryptography 39.0.0
paramiko 2.9.3
pip 22.2.1
pycparser 2.21
PyNaCl 1.5.0
pysftp 0.2.9
setuptools 63.2.0
six 1.16.0
```

Anexos E

CONFIGURACION ELASTIC

A continuación se incluyen los ficheros de configuración, correspondientes los programas de la sección 3.1.2.

E.1. Configuración de *Elasticsearch*

Por pantalla se muestra el fichero de configuración *elasticsearch.yml*, ubicado en el equipo *Debian 11*:

```
# By default Elasticsearch is only accessible on localhost. Set a
   ↪ different
# address here to expose this node on the network:
#
network.host: 192.168.153.2
#
# By default Elasticsearch listens for HTTP traffic on the first free
   ↪ port it
# finds starting at 9200. Set a specific HTTP port here:
#
http.port: 9200
#
# For more information, consult the network module documentation.
#
# --------------------------------- Discovery
   ↪ ----------------------------------
#
# Pass an initial list of hosts to perform discovery when this node is
   ↪ started:
# The default list of hosts is ["127.0.0.1", "[::1]"]
#
#discovery.seed_hosts: ["host1", "host2"]
#
# Bootstrap the cluster using an initial set of master-eligible nodes:
#
#cluster.initial_master_nodes: ["node-1", "node-2"]
```

```
#
# For more information, consult the discovery and cluster formation
    ↪ module documentation.
#
# ---------------------------------- Readiness
    ↪ -----------------------------------
#
# Enable an unauthenticated TCP readiness endpoint on localhost
#
#readiness.port: 9399
#
# ---------------------------------- Various
    ↪ -----------------------------------
#
# Allow wildcard deletion of indices:
#
#action.destructive_requires_name: false

#----------------------- BEGIN SECURITY AUTO CONFIGURATION
    ↪ -----------------------
#
# The following settings, TLS certificates, and keys have been
    ↪ automatically
# generated to configure Elasticsearch security features on 25-02-2023
    ↪ 11:26:54
#
#
    ↪ --------------------------------------------------------------------------------
    ↪

# Enable security features
xpack.security.enabled: true

xpack.security.enrollment.enabled: true

# Enable encryption for HTTP API client connections, such as Kibana,
    ↪ Logstash, and Agents
xpack.security.http.ssl:
  enabled: true
  keystore.path: certs/http.p12

# Enable encryption and mutual authentication between cluster nodes
xpack.security.transport.ssl:
  enabled: true
  verification_mode: certificate
  keystore.path: certs/transport.p12
  truststore.path: certs/transport.p12
# Create a new cluster with the current node only
```

```
# Additional nodes can still join the cluster later
cluster.initial_master_nodes: ["zeek-s0-IP"]

# Allow HTTP API connections from anywhere
# Connections are encrypted and require user authentication
http.host: 0.0.0.0

# Allow other nodes to join the cluster from anywhere
# Connections are encrypted and mutually authenticated
#transport.host: 0.0.0.0

#----------------------- END SECURITY AUTO CONFIGURATION
   ↪ -------------------------
```

Por pantalla se muestra el *mapping* utilizado por el índice creado y la política de rotación aplicada al mismo:

```
{
  "mappings": {
    "_meta": {
      "version": "8.0.1"
    },
    "dynamic_templates": [],
    "date_detection": false,
    "properties": {
      "@timestamp": {
        "type": "date"
      },
      "CAPEdata": {
        "properties": {
          "alert": {
            "properties": {
              "action": {
                "type": "keyword",
                "ignore_above": 1024
              },
              "category": {
                "type": "keyword",
                "ignore_above": 1024
              },
              "gid": {
                "type": "long"
              },
              "metadata": {
                "properties": {
                  "affected_product": {
                    "type": "text",
                    "fields": {
```

```
        "keyword": {
          "type": "keyword",
          "ignore_above": 256
        }
      }
    },
    "attack_target": {
      "type": "text",
      "fields": {
        "keyword": {
          "type": "keyword",
          "ignore_above": 256
        }
      }
    },
    "created_at": {
      "type": "text",
      "fields": {
        "keyword": {
          "type": "keyword",
          "ignore_above": 256
        }
      }
    },
    "deployment": {
      "type": "text",
      "fields": {
        "keyword": {
          "type": "keyword",
          "ignore_above": 256
        }
      }
    },
    "former_category": {
      "type": "text",
      "fields": {
        "keyword": {
          "type": "keyword",
          "ignore_above": 256
        }
      }
    },
    "performance_impact": {
      "type": "text",
      "fields": {
        "keyword": {
          "type": "keyword",
          "ignore_above": 256
```

```
            }
          }
        },
        "signature_severity": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        },
        "tag": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        },
        "updated_at": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        }
      }
    },
    "rev": {
      "type": "long"
    },
    "severity": {
      "type": "long"
    },
    "signature": {
      "type": "keyword",
      "ignore_above": 1024
    },
    "signature_id": {
      "type": "long"
    }
  }
},
"analysis_id": {
```

```
      "type": "long"
    },
    "app_proto": {
      "type": "keyword",
      "ignore_above": 1024
    },
    "dest_ip": {
      "type": "ip"
    },
    "dest_port": {
      "type": "long"
    },
    "event_type": {
      "type": "keyword",
      "ignore_above": 1024
    },
    "fileinfo": {
      "properties": {
        "end": {
          "type": "long"
        },
        "filename": {
          "type": "keyword",
          "ignore_above": 1024
        },
        "gaps": {
          "type": "boolean"
        },
        "size": {
          "type": "long"
        },
        "start": {
          "type": "long"
        },
        "state": {
          "type": "keyword",
          "ignore_above": 1024
        },
        "stored": {
          "type": "boolean"
        },
        "tx_id": {
          "type": "long"
        }
      }
    },
    "files": {
      "properties": {
```

```
        "end": {
          "type": "long"
        },
        "filename": {
          "type": "keyword",
          "ignore_above": 1024
        },
        "gaps": {
          "type": "boolean"
        },
        "size": {
          "type": "long"
        },
        "start": {
          "type": "long"
        },
        "state": {
          "type": "keyword",
          "ignore_above": 1024
        },
        "stored": {
          "type": "boolean"
        },
        "tx_id": {
          "type": "long"
        }
      }
    },
    "flow": {
      "properties": {
        "age": {
          "type": "long"
        },
        "alerted": {
          "type": "boolean"
        },
        "bytes_toclient": {
          "type": "long"
        },
        "bytes_toserver": {
          "type": "long"
        },
        "end": {
          "type": "keyword",
          "ignore_above": 1024
        },
        "pkts_toclient": {
          "type": "long"
```

```
      },
      "pkts_toserver": {
        "type": "long"
      },
      "reason": {
        "type": "keyword",
        "ignore_above": 1024
      },
      "start": {
        "type": "keyword",
        "ignore_above": 1024
      },
      "state": {
        "type": "keyword",
        "ignore_above": 1024
      }
    }
  },
  "flow_id": {
    "type": "long"
  },
  "geo_dst": {
    "type": "geo_point"
  },
  "geo_src": {
    "type": "geo_point"
  },
  "http": {
    "properties": {
      "content_range": {
        "properties": {
          "end": {
            "type": "long"
          },
          "raw": {
            "type": "keyword",
            "ignore_above": 1024
          },
          "size": {
            "type": "long"
          },
          "start": {
            "type": "long"
          }
        }
      },
      "hostname": {
        "type": "keyword",
```

```
        "ignore_above": 1024
      },
      "http_content_type": {
        "type": "keyword",
        "ignore_above": 1024
      },
      "http_method": {
        "type": "keyword",
        "ignore_above": 1024
      },
      "http_user_agent": {
        "type": "keyword",
        "ignore_above": 1024
      },
      "length": {
        "type": "long"
      },
      "protocol": {
        "type": "keyword",
        "ignore_above": 1024
      },
      "redirect": {
        "type": "text",
        "fields": {
          "keyword": {
            "type": "keyword",
            "ignore_above": 256
          }
        }
      },
      "status": {
        "type": "long"
      },
      "url": {
        "type": "keyword",
        "ignore_above": 1024
      }
    }
  },
  "icmp_code": {
    "type": "long"
  },
  "icmp_type": {
    "type": "long"
  },
  "metadata": {
    "properties": {
      "flowbits": {
```

```
      "type": "keyword",
      "ignore_above": 1024
    },
    "flowints": {
      "properties": {
        "tcp": {
          "properties": {
            "retransmission": {
              "properties": {
                "count": {
                  "type": "long"
                }
              }
            }
          }
        },
        "tls": {
          "properties": {
            "anomaly": {
              "properties": {
                "count": {
                  "type": "long"
                }
              }
            }
          }
        }
      }
    }
  }
},
"pcap_cnt": {
  "type": "long"
},
"proto": {
  "type": "keyword",
  "ignore_above": 1024
},
"response_icmp_code": {
  "type": "long"
},
"response_icmp_type": {
  "type": "long"
},
"src_ip": {
  "type": "ip"
},
"src_port": {
```

```
      "type": "long"
    },
    "tcp": {
      "properties": {
        "ack": {
          "type": "boolean"
        },
        "fin": {
          "type": "boolean"
        },
        "psh": {
          "type": "boolean"
        },
        "rst": {
          "type": "boolean"
        },
        "state": {
          "type": "keyword",
          "ignore_above": 1024
        },
        "syn": {
          "type": "boolean"
        },
        "tcp_flags": {
          "type": "keyword",
          "ignore_above": 1024
        },
        "tcp_flags_tc": {
          "type": "keyword",
          "ignore_above": 1024
        },
        "tcp_flags_ts": {
          "type": "keyword",
          "ignore_above": 1024
        }
      }
    },
    "timestamp": {
      "type": "keyword",
      "ignore_above": 1024
    },
    "tls": {
      "properties": {
        "fingerprint": {
          "type": "keyword",
          "ignore_above": 1024
        },
        "issuerdn": {
```

```
      "type": "keyword",
      "ignore_above": 1024
    },
    "ja3": {
      "properties": {
        "hash": {
          "type": "keyword",
          "ignore_above": 1024
        },
        "string": {
          "type": "keyword",
          "ignore_above": 1024
        }
      }
    },
    "ja3s": {
      "properties": {
        "hash": {
          "type": "keyword",
          "ignore_above": 1024
        },
        "string": {
          "type": "keyword",
          "ignore_above": 1024
        }
      }
    },
    "notafter": {
      "type": "keyword",
      "ignore_above": 1024
    },
    "notbefore": {
      "type": "keyword",
      "ignore_above": 1024
    },
    "serial": {
      "type": "keyword",
      "ignore_above": 1024
    },
    "session_resumed": {
      "type": "boolean"
    },
    "sni": {
      "type": "keyword",
      "ignore_above": 1024
    },
    "subject": {
      "type": "keyword",
```

```
              "ignore_above": 1024
            },
            "version": {
              "type": "keyword",
              "ignore_above": 1024
            }
          }
        },
        "tx_id": {
          "type": "long"
        }
      }
    },
    "tags": {
      "type": "text",
      "fields": {
        "keyword": {
          "type": "keyword",
          "ignore_above": 256
        }
      }
    }
  }
}

{
  "settings": {
    "index": {
      "lifecycle": {
        "name": "politica-prueba",
        "rollover_alias": "reportes.soc"
      },
      "routing": {
        "allocation": {
          "include": {
            "_tier_preference": "data_content"
          }
        }
      },
      "mapping": {
        "total_fields": {
          "limit": "10000"
        }
      },
      "refresh_interval": "5s",
      "number_of_shards": "1",
      "provided_name": "reportes.soc-000006",
```

```
    "creation_date": "1682091745159",
    "priority": "100",
    "number_of_replicas": "1",
    "uuid": "rYGp5USoQ4e_Ej6MwbSSrQ",
    "version": {
      "created": "8060299"
    }
  }
},
"defaults": {
  "index": {
    "flush_after_merge": "512mb",
    "time_series": {
      "end_time": "9999-12-31T23:59:59.999Z",
      "start_time": "-9999-01-01T00:00:00Z"
    },
    "final_pipeline": "_none",
    "max_inner_result_window": "100",
    "unassigned": {
      "node_left": {
        "delayed_timeout": "1m"
      }
    },
    "max_terms_count": "65536",
    "rollup": {
      "source": {
        "name": "",
        "uuid": ""
      }
    },
    "lifecycle": {
      "parse_origination_date": "false",
      "step": {
        "wait_time_threshold": "12h"
      },
      "indexing_complete": "false",
      "origination_date": "-1"
    },
    "mode": "standard",
    "routing_partition_size": "1",
    "force_memory_term_dictionary": "false",
    "max_docvalue_fields_search": "100",
    "merge": {
      "scheduler": {
        "max_thread_count": "2",
        "auto_throttle": "true",
        "max_merge_count": "7"
      },
```

```
      "policy": {
        "floor_segment": "2mb",
        "max_merge_at_once_explicit": "30",
        "max_merge_at_once": "10",
        "max_merged_segment": "5gb",
        "expunge_deletes_allowed": "10.0",
        "segments_per_tier": "10.0",
        "deletes_pct_allowed": "33.0"
      }
    },
    "max_refresh_listeners": "1000",
    "max_regex_length": "1000",
    "load_fixed_bitset_filters_eagerly": "true",
    "number_of_routing_shards": "1",
    "write": {
      "wait_for_active_shards": "1"
    },
    "verified_before_close": "false",
    "mapping": {
      "coerce": "false",
      "nested_fields": {
        "limit": "50"
      },
      "depth": {
        "limit": "20"
      },
      "field_name_length": {
        "limit": "9223372036854775807"
      },
      "nested_objects": {
        "limit": "10000"
      },
      "ignore_malformed": "false",
      "dimension_fields": {
        "limit": "16"
      }
    },
    "source_only": "false",
    "soft_deletes": {
      "enabled": "true",
      "retention": {
        "operations": "0"
      },
      "retention_lease": {
        "period": "12h"
      }
    },
    "max_script_fields": "32",
```

```
"query": {
  "default_field": [
    "*"
  ],
  "parse": {
    "allow_unmapped_fields": "true"
  }
},
"format": "0",
"frozen": "false",
"sort": {
  "missing": [],
  "mode": [],
  "field": [],
  "order": []
},
"routing_path": [],
"version": {
  "compatibility": "8060299"
},
"codec": "default",
"max_rescore_window": "10000",
"bloom_filter_for_id_field": {
  "enabled": "true"
},
"max_adjacency_matrix_filters": "100",
"analyze": {
  "max_token_count": "10000"
},
"gc_deletes": "60s",
"top_metrics_max_size": "10",
"optimize_auto_generated_id": "true",
"max_ngram_diff": "1",
"hidden": "false",
"translog": {
  "generation_threshold_size": "64mb",
  "flush_threshold_size": "512mb",
  "sync_interval": "5s",
  "retention": {
    "size": "-1",
    "age": "-1"
  },
  "durability": "REQUEST"
},
"auto_expand_replicas": "false",
"recovery": {
  "type": ""
},
```

```
"requests": {
  "cache": {
    "enable": "true"
  }
},
"data_path": "",
"highlight": {
  "max_analyzed_offset": "1000000"
},
"routing": {
  "rebalance": {
    "enable": "all"
  },
  "allocation": {
    "disk": {
      "watermark": {
        "ignore": "false"
      }
    },
    "enable": "all",
    "total_shards_per_node": "-1"
  }
},
"search": {
  "slowlog": {
    "level": "TRACE",
    "threshold": {
      "fetch": {
        "warn": "-1",
        "trace": "-1",
        "debug": "-1",
        "info": "-1"
      },
      "query": {
        "warn": "-1",
        "trace": "-1",
        "debug": "-1",
        "info": "-1"
      }
    }
  },
  "idle": {
    "after": "30s"
  },
  "throttled": "false"
},
"fielddata": {
  "cache": "node"
```

```
    },
    "look_ahead_time": "2h",
    "default_pipeline": "_none",
    "max_slices_per_scroll": "1024",
    "shard": {
      "check_on_startup": "false"
    },
    "xpack": {
      "watcher": {
        "template": {
          "version": ""
        }
      },
      "version": "",
      "ccr": {
        "following_index": "false"
      }
    },
    "percolator": {
      "map_unmapped_fields_as_text": "false"
    },
    "allocation": {
      "max_retries": "5",
      "existing_shards_allocator": "gateway_allocator"
    },
    "indexing": {
      "slowlog": {
        "reformat": "true",
        "threshold": {
          "index": {
            "warn": "-1",
            "trace": "-1",
            "debug": "-1",
            "info": "-1"
          }
        },
        "source": "1000",
        "level": "TRACE"
      }
    },
    "compound_format": "0.1",
    "blocks": {
      "metadata": "false",
      "read": "false",
      "read_only_allow_delete": "false",
      "read_only": "false",
      "write": "false"
    },
```

```
"max_result_window": "10000",
"store": {
  "stats_refresh_interval": "10s",
  "type": "",
  "fs": {
    "fs_lock": "native"
  },
  "preload": [],
  "snapshot": {
    "snapshot_name": "",
    "index_uuid": "",
    "cache": {
      "prewarm": {
        "enabled": "true"
      },
      "enabled": "true",
      "excluded_file_types": []
    },
    "repository_uuid": "",
    "uncached_chunk_size": "-1b",
    "delete_searchable_snapshot": "false",
    "index_name": "",
    "partial": "false",
    "blob_cache": {
      "metadata_files": {
        "max_length": "64kb"
      }
    },
    "repository_name": "",
    "snapshot_uuid": ""
  }
},
"queries": {
  "cache": {
    "enabled": "true"
  }
},
"shard_limit": {
  "group": "normal"
},
"warmer": {
  "enabled": "true"
},
"downsample": {
  "source": {
    "name": "",
    "uuid": ""
  },
```

```
        "status": "unknown"
      },
      "override_write_load_forecast": "0.0",
      "max_shingle_diff": "3",
      "query_string": {
        "lenient": "false"
      }
    }
  }
}
```

E.2. Configuración de *Kibana*

Se muestra por pantalla el fichero de configuración de *Kibana*, correspondiente al archivo *kibana.yml*, ubicado en el equipo *Debian 11*:

```
# For more configuration options see the configuration guide for Kibana
   ↪  in
# https://www.elastic.co/guide/index.html

# ================== System: Kibana Server ==================
# Kibana is served by a back end server. This setting specifies the
   ↪ port to use.
server.port: 5601

# Specifies the address to which the Kibana server will bind. IP
   ↪ addresses and host names are both valid values.
# The default is 'localhost', which usually means remote machines will
   ↪ not be able to connect.
# To allow connections from remote users, set this parameter to a non-
   ↪ loopback address.
server.host: "192.168.153.2"


# Enables you to specify a file where Kibana stores log output.
logging:
  appenders:
    file:
      type: file
      fileName: /var/log/kibana/kibana.log
      layout:
        type: json
  root:
    appenders:
      - default
      - file
```

```
# ================== System: Other ==================
# The path where Kibana stores persistent data not saved in
    ↪ Elasticsearch. Defaults to data
#path.data: data

# Specifies the path where Kibana creates the process ID file.
pid.file: /run/kibana/kibana.pid

# This section was automatically generated during setup.
elasticsearch.hosts: ['https://localhost:9200']
elasticsearch.serviceAccountToken: TOKENGENERADOPARALAAUTENTIFICACION
elasticsearch.ssl.certificateAuthorities: [/var/lib/kibana/
    ↪ ca_1677325714158.crt]
xpack.fleet.outputs: [{id: fleet-default-output, name: default,
    ↪ is_default: true, is_default_monitoring: true, type:
    ↪ elasticsearch, hosts: ['https://192.168.153.2:9200'],
    ↪ ca_trusted_fingerprint:
    ↪ b966dc6425a6f1ae5717e1dc7aa274f9d6d36688b06263dbad6485436dd57ebc
    ↪ }]
```

Además, para mantener una comunicación cifrada con *elasticsearch*, se ha copiado al directorio */etc/kibana/certs/* el certificado generado por *elasticsearch*. Con las siguientes instrucciones se ha generado un token, que hace necesario el usuario y contraseña generados por *elasticsearch* en su instalación, para acceder a la interfaz *web* de *Kibana*:

```
/usr/share/elasticsearch/bin/elasticsearch-create-enrollment-token -s
    ↪ node
```

E.3. Configuración y filtros de *Logstash*

El código que se visualiza a continuación corresponde con el archivo con ruta */etc/logstash/conf.d/capereports.conf*. Este archivo permite ingerir y filtrar los datos generados por *Ubuntu Server*. Se ubica en el equipo *Debian 11*.

```
input {
  beats {
    port => "5050"
    type => "filestream"
    enable_metric => true
  }
}


filter {
```

```
    json{
      skip_on_invalid_json => true
      source => "message"
      target => "CAPEdata"
      add_tag => [ "_message_json_parsed" ]
    }

    if [CAPEdata][event_type] == "dns" or [CAPEdata][event_type] == "
       ↪ stats"
    or [CAPEdata][app_proto] == "dns"{
      mutate{
        remove_field => "CAPEdata"
      }
    }

    mutate {
      remove_field => ["message","agent", "event", "original", "ecs", "
         ↪ type",
      "prospector", "input", "beat", "host", "offset", "source", "log",
         ↪ "@version"]
    }

}


output {
  stdout { codec => rubydebug }
  elasticsearch {
    hosts => ["https://localhost:9200"]
    user => "elastic"
    password => "tosdkA1AnO_VTWBcWa2A"
    ssl => true
    cacert => "/etc/logstash/conf.d/certs/http_ca.crt"
    ilm_rollover_alias => "reportes.soc"
    ilm_pattern => "000001"
    ilm_policy => "politica-prueba"
    index => "reportes.soc"
 }

}
```

E.4. Configuración de *Filebeat*

Las siguientes líneas corresponden al fichero de configuración, que corresponde con el fichero */etc/filebeat/filebeat.yml*, ubicado en el equipo *Ubuntu Server*:

```
  # IPv6 addresses should always be defined as: https://[2001:db8
    ↪ ::1]:5601
  #host: "localhost:5601"

  # Kibana Space ID
  # ID of the Kibana Space into which the dashboards should be loaded.
    ↪  By default,
  # the Default Space will be used.
  #space.id:

# ============================== Elastic Cloud
   ↪ ================================

# These settings simplify using Filebeat with the Elastic Cloud (https
   ↪ ://cloud.elastic.co/).

# The cloud.id setting overwrites the `output.elasticsearch.hosts` and
# `setup.kibana.host` options.
# You can find the `cloud.id` in the Elastic Cloud web UI.
#cloud.id:

# The cloud.auth setting overwrites the `output.elasticsearch.username
   ↪ ` and
# `output.elasticsearch.password` settings. The format is `<user>:<
   ↪ pass>`.
#cloud.auth:

# ================================== Outputs
   ↪ ===================================

# Configure what output to use when sending the data collected by the
   ↪ beat.

# ---------------------------- Elasticsearch Output
   ↪ ----------------------------
#output.elasticsearch:
  # Array of hosts to connect to.
# hosts: ["localhost:9200"]

  # Protocol - either `http` (default) or `https`.
  #protocol: "https"

  # Authentication credentials - either API key or username/password.
  #api_key: "id:api_key"
  #username: "elastic"
  #password: "changeme"
```

```
# ------------------------------ Logstash Output
    ↪ --------------------------------
output.logstash:
  # The Logstash hosts
  hosts: ["192.168.153.2:5050"]

  # Optional SSL. By default is off.
  # List of root certificates for HTTPS server verifications
  #ssl.certificate_authorities: ["/etc/pki/root/ca.pem"]

  # Certificate for SSL client authentication
  #ssl.certificate: "/etc/pki/client/cert.pem"

  # Client Certificate Key
  #ssl.key: "/etc/pki/client/cert.key"

# ================================= Processors
    ↪ =================================
processors:
  - add_host_metadata:
      when.not.contains.tags: forwarded
  - add_cloud_metadata: ~
  - add_docker_metadata: ~
  - add_kubernetes_metadata: ~
```

Anexos F

Anexo de términos y siglas

- *Anti-VM*: Técnica o herramienta utilizada para detectar y evitar la ejecución de código malicioso en un entorno virtualizado.
- *Host*: Equipo físico o servidor que aloja una o varias máquinas virtuales.
- *Guest*: Máquina virtual que se ejecuta dentro de un entorno de virtualización.
- *DLL*: Biblioteca de enlace dinámico (*Dynamic Link Library*), que contiene código y datos que múltiples programas pueden usar al mismo tiempo.
- *Snapshot*: Estado o imagen guardada de una máquina virtual en un momento específico, que permite volver a ese estado.
- *Iptables*: Herramienta de filtrado de paquetes y configuración de *firewall* en sistemas basados en Linux.
- Índice: En *Elasticsearch* puede verse como una base de datos que contiene varios tipos, cada uno de los cuales contiene documentos con propiedades, de manera similar a las tablas en una base de datos relacional.
- *Dashboards*: Paneles de control visual que muestran información y métricas.
- *Logs*: Archivos que registran actividades, errores o información importante en un sistema o aplicación.
- *API*: Interfaz de programación de aplicaciones (Application Programming Interface), que define los métodos y protocolos para la comunicación entre diferentes componentes de *software*.
- *VNC*: *Virtual Network Computing*, es un protocolo que permite controlar y acceder a escritorios de forma remota.

- Interfaz *localhost*: Interfaz de red local que se refiere a la propia máquina o equipo en el que se ejecuta el *software*.
- *Proxy-web*: Servidor intermediario que actúa como intermediario entre los clientes y los servidores web.
- Hipervisor: *Software* o firmware que permite la creación y gestión de máquinas virtuales.
- *Open-source*: *Software* cuyo código fuente es público y está disponible para su inspección, modificación y distribución.
- *VPN*: Red privada virtual (*Virtual Private Network*), que permite establecer una conexión segura y cifrada a través de una red pública.
- *Firewall*: Sistema de seguridad que controla y filtra el tráfico de red basado en reglas predefinidas.
- *Script*: Conjunto de instrucciones o comandos escritos en un lenguaje de programación para realizar una tarea específica.
- Reglas *YARA*: Reglas utilizadas en la detección y clasificación de *malware* basadas en patrones y características.
- *InetSim*: Herramienta utilizada para simular servicios de Internet en un entorno controlado para pruebas de seguridad.
- *Sockets*: Punto final de una conexión de red que permite la comunicación entre dos programas en diferentes nodos de una red.
- *Cookies*: Pequeños archivos almacenados en el navegador que contienen información sobre la interacción del usuario con un sitio web.
- *TOR*: *The Onion Router*, una red anónima que permite acceder a Internet de forma privada y segura.
- Sección *Issues*: Sección o área dedicada a informar problemas, errores o sugerencias en un proyecto.
- *GitHub*: Plataforma de desarrollo colaborativo basada en *Git* que permite el almacenamiento y seguimiento de versiones de proyectos.
- *debugger*: Herramienta o programa utilizado para encontrar y solucionar errores y problemas en el código durante el desarrollo de *software*.

- Archivo *.pcap*: Archivo que contiene datos de captura de paquetes de red.
- Archivo *.conf*: Archivo de configuración utilizado por aplicaciones o sistemas para personalizar su funcionamiento.
- Archivo *.xml*: Archivo de lenguaje de marcado extensible (*eXtensible Markup Language*) utilizado para almacenar y transportar datos.
- *TalosIntelligence*: Organización de inteligencia de amenazas que investiga y proporciona información sobre amenazas de seguridad.
- *Windows Update*: Servicio de actualización de *Microsoft* para mantener el sistema operativo *Windows* actualizado.
- *IOCs*: Indicadores de compromiso (*Indicators of Compromise*), que son evidencias que indican una posible violación de seguridad.
- *MD5 hash*: Resumen criptográfico obtenido mediante el algoritmo *MD5*, utilizado para verificar la integridad y autenticidad de datos.
- *Mutex*: Objeto de sincronización, utilizado para asegurar el acceso exclusivo a un recurso compartido en programación concurrente.
- *Wannacry*: Es un tipo de *ransomware* que se propagó a nivel mundial en 2017. Infectaba computadoras aprovechando una vulnerabilidad en los sistemas operativos Microsoft Windows conocida como *EternalBlue*.
- *Petya*: Es un *ransomware* que surgió en 2016.
- *Emotet*: Es un troyano bancario, que se hizo conocido en 2014, propagado a través de correos electrónicos de *phishing*.
- *POST*: Es un método de solicitud utilizado en el protocolo *HTTP* (*Hypertext Transfer Protocol*). Se utiliza para enviar datos al servidor para que sean procesados.
- *GET*: Es un método de solicitud utilizado en el protocolo *HTTP*. Se utiliza para solicitar datos del servidor.
- *.html*: Los archivos con la extensión *.html* contienen el código fuente de una página *web* y se pueden ver en los navegadores *web*.

- *Suricata*: Es un sistema de detección de intrusiones y prevención de amenazas en redes de computadoras. Es capaz de analizar el tráfico de red en busca de patrones maliciosos y comportamientos sospechosos que podrían indicar un ataque o una intrusión.
- *ANTI-DEBUG*: Técnicas y herramientas utilizadas para evitar o dificultar la depuración y el análisis de programas o procesos.
- *Adloader*: *Software* maliciosos que se infiltra en un sistema y cargan anuncios no deseados o potencialmente peligrosos.
- *Nymaim*: Un tipo de *malware* que se clasifica como un troyano de acceso remoto (*RAT*). suele ser utilizado para actividades maliciosas como el robo de información o el despliegue de ataques adicionales.