Trabajo Fin de Grado
Grado de Ingeniería de las Tecnologías de Telecomunicación

Diseño de un Entorno Open Source para Análisis Automatizados de Malware

Autor: Manuel Martín Gutiérrez
Tutor: Francisco Javier Muñoz Calle

Dpto. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Diseño de un Entorno Open Source para Análisis Automatizados de Malware

Autor:
Manuel Martín Gutiérrez

Tutor:
Francisco Javier Muñoz Calle
Profesor Colaborador

Dep. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Trabajo Fin de Grado: Diseño de un Entorno Open Source para Análisis Automatizados de Malware

Autor: Manuel Martín Gutiérrez

Tutor: Francisco Javier Muñoz Calle

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

La memoria de este trabajo se distribuye bajo la licencia GNU GPL v3.

La Licencia Pública General de GNU es una licencia libre, sin derechos para software y otros tipos de trabajos.

Las licencias para la mayoría del software y otros trabajos prácticos están diseñadas para suprimir la libertad de compartir y modificar los trabajos. Por el contrario, la Licencia Pública General de GNU pretende garantizar su libertad para compartir y modificar todas las versiones de un programa -- y garantizar que siga siendo software libre para todos sus usuarios. Nosotros, la Fundación para el Software Libre, utilizamos la Licencia Pública General de GNU para la mayoría de nuestro software; se aplica también a cualquier otro trabajo publicado de esta manera por sus autores. También puede aplicarlo a sus programas.

Cuando hablamos de software libre, nos referimos a la libertad, no al precio. Nuestras Licencias Públicas Generales están diseñadas para asegurar que tiene la libertad de distribuir copias de software libre (y cobrar por ellas si lo desea), de recibir el código fuente o de poder obtenerlo si lo desea, de modificar el software o usar partes del mismo en nuevos programas libres y de saber que puede hacer estas cosas.

Para conseguir más información sobre la licencia y sus términos se pueden seguir los siguientes enlaces:

- Información general en castellano: https://www.gnu.org/licenses/licenses.es.html
- Licencia original en inglés: https://www.gnu.org/licenses/gpl.html

There is nothing so stable as change.

Bob Dylan

Agradecimientos

Numerosas personas han contribuido a la evolución experimentada a lo largo de estos años, desde aquellas que conocí durante mi primer día en la universidad hasta la última que ha aportado su granito de arena para hacer posible este momento. Éstas, en mayor o menor medida, han ido dejando su huella en el camino que día a día voy creando. El proyecto llevado a cabo ha sido consecuencia de ese proceso de desarrollo personal, académico y profesional. A todos ellos va dirigido mi más sincero agradecimiento.

Al profesorado del Departamento de Ingeniería Telemática en la ETSI de la Universidad de Sevilla, sobre todo al tutor de este trabajo, por proporcionarme la base de conocimientos necesaria para profundizar en mis ideas e inquietudes tecnológicas.

A los compañeros de GMV Secure e-Solutions que me han transmitido su saber hacer. Además, a los responsables de la sección de Arquitectura por confiar en mí y darme la oportunidad de crecer profesionalmente.

A aquellos con los que me une una gran amistad tras mi paso por Madrid por hacer que me sintiese como en casa estando tan lejos.

A mis dos compañeros de batallas por compartir todos los momentos importantes de la titulación, por crear ese fuerte vínculo entre los tres y por formar parte de lo mejor que he obtenido de mi experiencia universitaria.

A mis amigos de siempre por seguir teniendo su apoyo después de tantos años y hacer como si no pasase el tiempo con cada reencuentro.

A mis padres, mis hermanos y mis abuelos por ser el pilar fundamental de mi vida; en especial, a Carmen.

Manuel Martín Gutiérrez

Resumen

Hoy en día, la ciberseguridad es un aspecto que preocupa a la totalidad de las organizaciones. Debido a la repercusión de incidentes en algunas de las mayores empresas multinacionales crece cada vez más tanto el interés mostrado en este ámbito como el presupuesto dedicado a él. Ejemplo de ello son los sucesos acontecidos recientemente a nivel global relacionados con malware, como WannaCry y NotPetya. En estos casos se levantó tal revuelo, incluso en medios generalistas, que fueron tema de actualidad en toda la sociedad y no sólo entre los profesionales del sector.

Ante esta tesitura, se decide investigar primero la naturaleza de esa modalidad de software. Es decir, tipos existentes, finalidades con las que se crean, evolución sufrida a lo largo de los años, vectores de ataque por donde suelen introducirse en una organización, etc.

Una vez asentado el conocimiento base sobre malware, se sigue investigando sobre qué tipo de métodos existen para analizarlo y detectarlo. En base a ello, se estudia el mercado para tener constancia de las soluciones comerciales que implementan dichos métodos. Aquí es donde se aprecia que hay disponibles numerosas soluciones propietarias tanto de fabricantes de nicho, ej.: FireEye, como de fabricantes más extendidos en el sector de las tecnologías de la información, ej.: Cisco. En contraposición, se encuentra un buen abanico de herramientas, que no soluciones, Open Source muy potentes, con mucha proyección y de una enorme utilidad.

Es importante aclarar la diferencia entre herramienta y solución para entender por completo el enfoque de este trabajo. Estas herramientas Open Source son muy específicas, para lo que han sido diseñadas, y aportan por separado muchas funcionalidades cruciales a la hora de analizar y detectar malware. En cambio, al ser proyectos independientes, carecen de consistencia entre si y en la mayoría de los casos, incluso de capacidad para intercambiar la información generada. Justo ese es el valor añadido que aportan las soluciones propietarias: un paquete de herramientas, funcionalidades, unificado de forma coherente, con entidad propia y listo para ser comercializado.

Con toda esta información, se procede a diseñar un entorno basado en herramientas Open Source que incluya una solución capaz de integrarlas, de hacer que trabajen conjuntamente de forma automatizada y transparente para el usuario. Se implementa un modelo virtual del entorno junto con la solución para facilitar el desarrollo,

llevar a cabo pruebas de concepto, depurar ideas y sacar conclusiones.

Por último, una vez conseguido los objetivos propuestos con la ayuda de la experimentación en el modelo desarrollado, se detectan los puntos débiles, se halla una forma de pulirlos y se acaba diseñando la hoja de ruta a seguir.

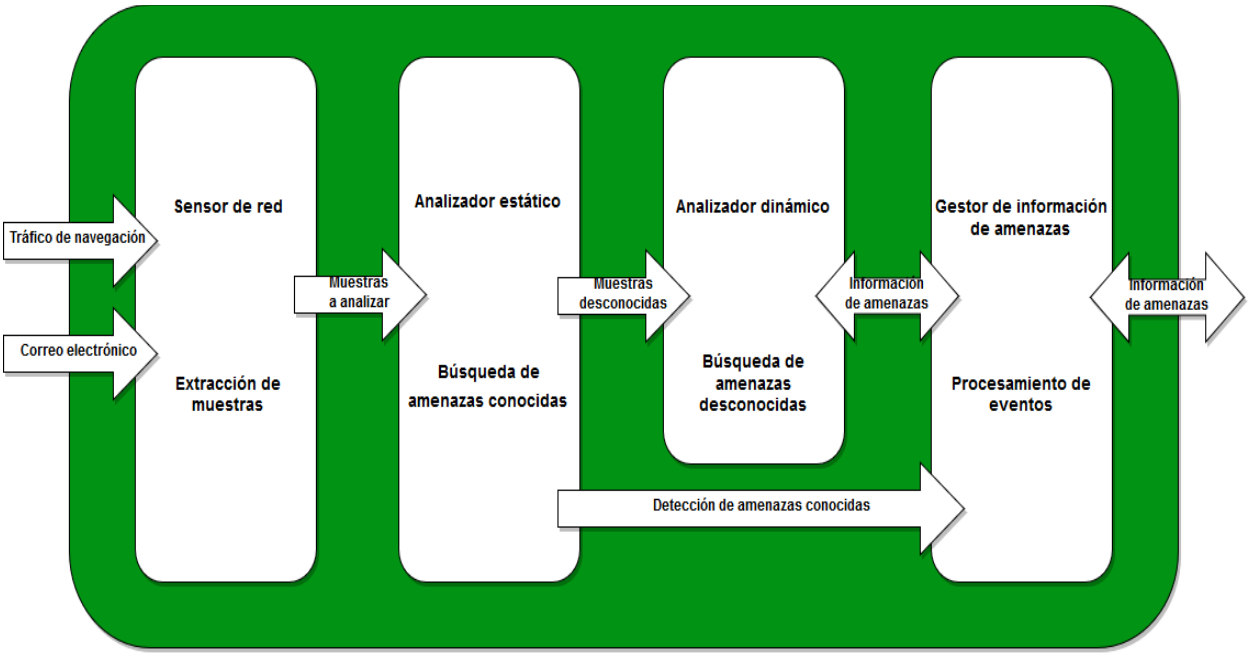


Ilustración I. Diagrama resumen del entorno

Abstract

Today, cybersecurity is an issue of concern to all organizations. Due to the impact of incidents on some of the largest multinational companies, both the interest shown in this area and the budget allocated to it are growing. Examples of this are the recent global events related to malware, e.g. WannaCry and NotPetya. In these cases, such a stir was created, even in the general media, that they were a current issue throughout society and not only among professionals in the sector.

Based on the above, it is decided to investigate the nature of these malicious software. That is to say, existing types, evolution suffered over the years and the future evolution that is expected, attack vectors, etc.

Once the knowledge base on malware has been established, it is carried out further research into what kind of methods exist to analyze and detect it. Based on this, a market study is performed to be aware of the commercial solutions that implement these methods. It is conclude that there are a lot of proprietary solutions available from both niche manufacturers, e.g. FireEye, and widespread vendors in the information technology sector, e.g. Cisco. However, there is a huge amount of powerful and useful open source tools, but not solutions.

It is important to clarify the difference between tool and solution in order to fully understand the approach of this work. These Open Source tools are specific to what they have been designed for and separately provide many critical malware scanning and detection features. On the other hand, as independent projects, they lack consistency among themselves and in most cases, even the capacity to exchange the information generated. This is the added value that proprietary solutions provide: a package of tools, functionalities, unified in a coherent way, with its own entity and ready to be marketed.

With all this information, it is proceed to design an environment based on Open Source tools and then design a solution capable of integrating them, making them work together in an automated and transparent way for the user. A virtual model of the environment is implemented along with the solution to facilitate development, carry out proof of concept, refine ideas and draw conclusions.

Finally, once the proposed objectives have been achieved with the help of experimentation in the developed model, weak points are detected, a way of strengthen them is found and the roadmap to follow is designed.

Índice

Índice de Tablas

Índice de Figuras

Notación

$ comando	Ejecución de “comando” en la terminal del sistema operativo
[…]	Una o más líneas omitidas de la salida de un comando o del contenido de un fichero

1 INTRODUCCIÓN

Hey you! Don't tell me there's no hope at all.

Together we stand, divided we fall.

Roger Waters, Pink Floyd

Antes de comenzar, es importante matizar que el trabajo académico realizado siguiendo esta idea es realidad gracias a dos personas, ambas caracterizadas por su dedicación y entrega a la hora de transmitir sus conocimientos. Por una parte, Javier Muñoz, tutor de este proyecto y profesor de una de las asignaturas más prácticas e interesantes del último curso. Por otra, Godofredo Fernández, también profesor y el "culpable" de que tuviese claro que me quería dedicar a la seguridad de la información a lo largo de mi carrera profesional.

1.1 Motivación

Como se verá a lo largo del trabajo, los métodos basados en firmas que usan principalmente productos como los Antivirus y los IDS/IPS sólo son útiles con amenazas conocidas. Si la defensa de una organización se basa de forma exclusiva en este tipo de tecnologías, la seguridad se puede ver comprometida en cuanto reciba una muestra de malware y se dé una de las siguientes casuísticas:

- La muestra es una amenaza desconocida porque se ha liberado recientemente y el analizador aún no tiene la firma en su base de datos.

- La muestra pertenece a una amenaza conocida, pero ha modificado su composición lo suficiente como para tener una firma diferente y el analizador aún no la tiene en su base de datos.
- La muestra es una amenaza desconocida porque ha sido diseñada específicamente para atacar a un objetivo determinado después de haber recolectado suficiente información del entorno de la víctima y el analizador aún no la tiene en su base de datos.

Este último caso es el que más preocupa a organizaciones que gestionen sistemas críticos y de alta probabilidad a estar en el punto de mira de los atacantes, como las pertenecientes al sector bancario y al sector energético, por ejemplo. Los fabricantes de productos de ciberseguridad son conscientes de ello. Ya no se limitan a comercializar únicamente este tipo de analizadores, sino que los complementan con productos capaces de analizar el comportamiento y automatizar el ciclo de tratamiento de las muestras: Extracción → Análisis Estático (no se ejecuta la muestra previamente extraída) → Análisis Dinámico (sí se ejecuta de forma controlada). En la sección 2.6 se aborda con más detalle ambos tipos de análisis.

También existen diferentes proyectos Open Source con mucho potencial tanto estáticos en sus diferentes enfoques (ClamAV, Yara, etc.) como capaces de inspeccionar con detalle el comportamiento (Cuckoo Sandbox, F-Secure SEE, etc.). Aun así, se echa en falta ese añadido que aportan muchas de las soluciones propietarias de automatización de tareas y de facilidades de uso para los equipos de operaciones de seguridad.

Dicha carencia es la que genera la idea que posteriormente se ve materializada en este trabajo. La clave para combatir el malware y, a su vez, competir comercialmente con soluciones propietarias es trabajar juntos integrando los puntos fuertes de cada proyecto.

Este proyecto podría usarse en auditorías de seguridad; como de servicios o de red. Serviría para determinar de forma puntual si existe algún tipo de malware circulando por el entorno auditado. No obstante, está pensado para ser desplegado en producción y operado en el día a día de una organización. Ya que los objetivos son mantener un registro del estado del entorno a lo largo del tiempo y detectar amenazas lo antes posible.

1.2 Objetivos

Lo que se pretende conseguir es un entorno estable, que una vez desplegado y bien configurado sea capaz de automatizar el ciclo de tratamiento de las muestras. Esto incluye desde su extracción del tráfico de red, hasta su paso por los diferentes tipos de analizadores.

En cuanto a las herramientas usadas, las únicas limitaciones son que deben ser Open Source y que la muestra no salga de la infraestructura propia para ser analizada por motivos de privacidad y protección de los datos. Servicios online como VirusTotal y Malwr son realmente útiles y ofrecen un resultado rápido, pero se estarían exponiendo a terceros posibles archivos con información privada de la organización.

Por otra parte, se tiene en cuenta desde el comienzo de la investigación que la futura solución desarrollada que integre las diferentes herramientas debe ofrecer:

- Estabilidad y robustez durante su funcionamiento
- Automatización del intercambio de información entre herramientas
- Claridad en los resultados
- Flexibilidad a la hora de decidir qué tipos de análisis se quieren llevar a cabo:
    - Sólo estático
    - Sólo dinámico
    - Combinación de ambos para aprovechar las ventajas de cada uno
- Independencia de los métodos y productos usados para extraer las muestras. De esta forma, se aporta libertad durante la implementación por si la organización ya cuenta con IDS/IPS disponibles o productos capaces de extraer los archivos del tráfico de red.

1.3 Fases del trabajo

El trabajo se compone de cinco fases bien diferenciadas entre sí. A continuación, se enumeran las actividades llevadas a cabo en cada una de las mismas:

- Investigación:
  - Se estudia la naturaleza de los softwares maliciosos, su tipología, sus motivaciones, su evolución y vectores usados en los ataques para tener un sólido conocimiento sobre este tipo de amenazas.
  - Se pone el foco en los métodos que hay para detectarlos, es decir, en los tipos de análisis existentes para conocer con qué opciones se puede contar a la hora de defenderse de ellos.
  - Se profundiza en la información relativa a la detección de amenazas que se obtiene de los análisis. Es decir, qué datos se pueden extraer de ella, cómo tratarla, cómo almacenarla y cómo compartirla entre todas las herramientas que sean capaces de beneficiarse de dicha información.
- Diseño:
  - Se definen los puntos a cubrir en el futuro entorno y después se buscan los proyectos Open Source más interesantes relacionados con esas tecnologías. Se investigan con detenimiento para decidir cuáles de ellos formarán parte de la solución.
  - Se tienen en cuenta los requisitos de cada proyecto elegido, sus posibilidades de despliegue, sus opciones de intercambio de información y se dibuja la arquitectura del entorno indicando cómo interaccionan entre sí.
  - Se idea una forma de automatizar el envío de cada nuevo fichero extraído de la red hacia los diferentes analizadores y, en función de los resultados, generar eventos en la plataforma centralizada de información.
- Implementación:
  - Se despliega un modelo virtual plenamente funcional basado en la arquitectura diseñada con anterioridad tratando de minimizar los requisitos hardware. La idea es poder usarlo como centro de experimentación tanto para las herramientas elegidas como para la posterior solución que se desarrolle.
  - Se programa la solución encargada de hacer que todas las herramientas trabajen en conjunto. Se usan de guía los requisitos expuestos en el apartado anterior y la idea desarrollada durante la fase de diseño.
- Experimentación:
  - Se comprueba que todas las herramientas han sido desplegadas correctamente y después se verifica si la solución desarrollada cumple con su cometido. Se observan resultados satisfactorios, ya que están dentro de lo esperado.
  - Se usa el modelo virtual como laboratorio para buscar formas de mejorar la flexibilidad de despliegue, la escalabilidad ante entornos exigentes, la experiencia de los usuarios y la interfaz gráfica de cara a ofrecer también una imagen unificada, una entidad propia.
- Documentación:
  - Se usan las anotaciones propias, las referencias, el código desarrollado y las ideas utilizadas para darle forma a este documento.
  - Se añaden como anexos diferentes guías propias de instalación, configuración y puesta a punto derivadas de la fase de implementación.

En total, ha tenido una duración aproximada de 15 semanas con una dedicación media de 20 horas semanales. En la siguiente tabla se muestra la distribución temporal por fases:

Tabla 1-1. Fases del trabajo

Fases	Semanas														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Investigación	20 h	20 h	20 h	20 h	10 h	10 h									
Diseño					10 h	10 h	20 h	10 h							
Implementación								10 h	20 h	20 h	20 h	20 h	10 h		
Experimentación													10 h	10 h	
Documentación														10 h	20 h

2 FUNDAMENTOS DE MALWARE

So you run and you run to catch up with the sun,
but it's sinking racing around to come up behind you again.

Roger Waters, Pink Floyd

La palabra “malware” proviene del concepto anglosajón “malicious software”. Se usa como término genérico para denominar a cualquier software que cumpla de forma deliberada la intención dañina de su creador. Por lo tanto, no se considera malware todo aquel software defectuoso que, aun teniendo comportamientos peligrosos, se deban a errores durante su programación inintencionados.

Durante este capítulo se sentará la base de conocimiento sobre la que se desarrolla el trabajo realizado en el resto del proyecto. En determinados casos, se utilizarán términos en inglés debido al escaso uso de sus traducciones en español para hacer referencia a ellos. De esta manera se facilita encontrar más información acudiendo a diferentes fuentes si se desea.

Antes de seguir, es necesario definir qué es un “Command & Control”. Abreviaturas habituales del término son C&C y C2. Se trata servidores que gestionan las comunicaciones con las máquinas comprometidas a través del malware instalado. Por un lado, el envío de órdenes y por otro, la recepción de datos de la víctima.

2.1 Tipología

En base al comportamiento del malware y al objetivo con el que han sido desarrollados, es decir, cómo y a qué afectan, se detallan a continuación los principales tipos. Debido a que una misma muestra puede realizar diversas actividades en función de la fase en la que se encuentre, los grupos que se describen en este apartado no son excluyentes entre sí.

2.1.1 Dropper – Downloader

Suelen ser sencillos programas cuyo único objetivo es instalar malware adicional. Normalmente pertenecen a la primera fase de un ataque y se quedan permanentemente en el sistema para insertar más malware en un futuro.

La principal diferencia entre los Droppers y los Downloaders es que los primeros traen consigo el malware a instalar y los otros se conectan a un servidor remoto para descargarlo. La URL puede venir predefinida en el código o generarse automáticamente mediante el uso de algoritmos.

2.1.2 Rogue software

Este tipo de herramientas se publicita como software pare prevenir y/o limpiar infecciones por malware y su principal objetivo es conseguir que el usuario realice un pago. Un detalle importante para distinguir a la mayoría de ellas es que no tienen un nombre de producto, una marca, para identificarse. En cambio, utilizan nombres genéricos como “Anti-Malware Tool”, “Malware Cleaner”, etc.

No todas las muestras de este tipo instalan malware de forma paralela al programa principal, pero sí que contactan con sus respectivos C&C con el fin de poder vender posteriormente la infección mediante el uso actualizaciones automáticas.

Son altamente peligrosas porque pueden afectar a la continuidad del negocio. Una de sus variantes es el ransomware, que hace imposible el uso de la máquina bloqueándola y genera un aviso pidiendo un pago por parte del usuario para liberarla. Hoy en día, en la mayoría de los casos cifran los datos del disco duro en segundo plano mientras muestran el aviso para pagar. A cambio, ofrecen la clave que devuelve el acceso a la información, aunque normalmente no se facilita ni cediendo a la extorsión y realizando el pago.

2.1.3 Troyano y Remote Administration Toolkit (RAT)

Están diseñados para tomar el control sobre la máquina comprometida. El objetivo es conseguir toda la información posible mediante la exfiltración de datos. Suelen requerir la interacción del usuario para iniciarse. Existen RATs prefabricados para comercializar con ellos y posteriormente poder ser personalizados. Ésa es una de las causas de que haya bastante disparidad en la sofisticación de las muestras encontradas pertenecientes a este tipo.

2.1.4 Rootkit

Son muy sigilosos y sofisticados. Suelen trabajar a nivel de los controladores, lo que los hace bastante complicados de detectar y eliminar. Esta misma razón es la que provoca que sean difíciles de desarrollar y requieran una buena inversión de tiempo, ya que un error en los controladores podría tumbar el sistema por completo.

Uno de los usos más frecuentes es esconder la actividad de otros tipos de malware. En el caso de los denominados RATs, podría encubrir de cara al sistema conexiones creadas desde un determinado puerto.

El único método capaz asegurar por completo una desinfección de esta categoría es formatear el disco y reinstalar el sistema operativo. En muchas de las muestras no es suficiente con sólo reinstalar el sistema operativo porque afectan a la partición de arranque y serían capaces de reinstalar el malware al volver a iniciar

la máquina.

2.1.5 Gusano

Su principal característica es que se difunden de forma rápida y automática. Hacen uso de diferentes métodos de propagación como discos extraíbles, correos electrónicos, la red, etc. Por lo tanto, rara vez se ven sin contener algunas de las funcionalidades pertenecientes al resto de categorías. Su capacidad para extenderse a lo largo de una organización los hace realmente peligrosos y, sobre todo, difíciles de erradicar por completo.

Hace tiempo eran típicas las muestras de este tipo que se realizaban sólo con el fin de molestar al usuario mostrando algún mensaje o reproduciendo música. En la actualidad, no suele ocurrir esto y normalmente están asociadas a motivos relacionados con los del resto de malware.

2.1.6 Virus

Se trata de ejecutables que infectan a otros ejecutables del sistema o a los sectores de arranque del disco. A día de hoy, es menos común que afecten a los sectores de arranque a excepción de si tienen una componente de tipo rootkit. De hecho, es probable que cuenten con funcionalidades de las otras categorías como droppers y downloaders, troyanos y RATs, etc.

Suelen ser bastante dañinos y como método de desinfección se recomienda un formateo del disco y una reinstalación completa del sistema operativo. Esto se debe a que buscan ejecutables y DLLs para infectarlos e incluso aunque se eliminen todas las copias en memoria, podrían volver a cargar su contenido en la máquina al ejecutar algún programa que no hubiese sido limpiado.

2.2 Propósitos

En los siguientes subapartados se detallan los principales propósitos que puede tener un determinado malware. En el primero de ellos se especifica que los beneficios económicos son directos porque a través de los otros dos también se podrían acabar consiguiendo. Dañar a alguna víctima concreta o robar información confidencial podría generar ganancias derivadas de estas acciones.

2.2.1 Obtener beneficios económicos directos

Para conseguir ganancias se usan dos métodos:

- Robar información bancaria: tarjetas, pins, credenciales de banca online, etc.
    - Según la variante de malware, es posible que ni siquiera importe qué tipo de cifrado use la página web del banco. Esto se debe a que consigue la información directamente de la memoria inyectando código en el proceso del navegador.
- Engañar o forzar al usuario para que realice pagos.
    - Se basan en rogue software, normalmente en ransomware, utilizando el procedimiento detallado en el apartado anterior.

2.2.2 Dañar los activos comprometidos

Consiste en causar la indisponibilidad de un servicio, máquina o incluso una organización. Esta era la motivación original cuando se comenzó a diseñar malware en la década de los 70. Podía venir derivada de la búsqueda de notoriedad y de la demostración de habilidades por parte del desarrollador. Ejemplos típicos de ello eran los programas encargados de modificar el arranque para mostrar algún mensaje indicando que había

sido comprometido.

El ejemplo más famoso hasta la fecha en este ámbito se dio en 2010 y se corresponde con el malware llamado Stuxnet. Tenía como objetivo la infraestructura nuclear iraní y consiguió causar daños importantes en los sistemas industriales de la central nuclear de Natanz.

2.2.3 Robar información

Suele ser malware usado para atacar de forma dirigida a una organización concreta y estar específicamente diseñado para tener éxito en ella. Esto conlleva que, en la mayoría de los casos, sean muestras nunca vistas anteriormente. El objetivo principal es hallar y exfiltrar datos relativos a la propiedad intelectual de la víctima.

En general, conllevan ataques complejos que dependen de diferentes fases. Normalmente se les relaciona con las amenazas persistentes avanzadas. En los próximos apartados se entrará en detalle a estudiar la naturaleza de este tipo de amenazas.

2.3 Vectores de ataque

Los vectores de ataque son las localizaciones, respecto al sistema amenazado, en las que se explotan las vulnerabilidades, ej.: red, local, etc. Éstas son utilizadas por los atacantes para acceder a sus víctimas. Varias de ellas se sirven de la ingeniería social. Dicho término hace referencia al conjunto de técnicas y herramientas usados para conseguir engañar al usuario y que facilite la infección de la organización a la que pertenece. Las interacciones del usuario pueden desencadenar la fuga de información o que el atacante tenga éxito con alguno de los otros objetivos descritos anteriormente.

Los vectores más comunes a día de hoy se encuentran descritos a continuación:

- Red: Se trata de evadir la seguridad perimetral y comprometer a la organización a través de vulnerabilidades explotables de forma remota. Tecnologías normalmente afectadas son:
    - Servicios disponibles: Ejemplos de este tipo son el DNS, los volúmenes compartidos, etc.
    - Navegación web: Consiste en hacer uso de vulnerabilidades en las aplicaciones web, de descargas de malware directamente al equipo o de infectar al propio navegador.
- Local: Se trata de evadir la seguridad en los equipos finales y comprometer a la organización a través de vulnerabilidades explotables de forma local. Tecnologías normalmente afectadas son:
    - Correo Electrónico: Se suele enviar malware adjunto o a través de enlaces en el cuerpo del correo. Mediante ingeniería social, se busca que el destinatario caiga en la trampa y libere la muestra en su entorno.
    - Dispositivos extraíbles: Un ejemplo de esta vía suele ser la infección mediante un dispositivo de almacenamiento extraíble. Una vez conectado al puerto USB del equipo, libera el malware. Al igual que mediante el correo electrónico, en la mayoría de los casos conlleva una importante componente de ingeniería social para conseguir que el usuario realice la acción necesaria, como insertar el dispositivo USB en este caso.

2.4 Activadores

En busca de hacer el malware lo más difícil posible de detectar, existe la posibilidad de diseñarlo para permanecer inactivo hasta que se produzcan ciertas condiciones. Muchas de ellas están destinadas a evitar que el malware se active en una máquina virtual.

Esta lista puede ser muy extensa y pormenorizada si se llega a bajo nivel, tanto que se saldría del alcance de estas primeras nociones sobre malware. Es interesante conocer las más frecuentes de cara a saber las

limitaciones e incluso a implementar posibles contramedidas en el entorno virtual. Por lo tanto, se describen a continuación los grupos más comunes de medidas que utilizan para intentar pasar desapercibidos en los análisis.

2.4.1 Básicos

En este grupo encontramos, en mayor medida, dos tipos de condiciones para que se inicie el malware:

- Parámetros de ejecución: Sólo se iniciará después de que se haya ejecutado con unos determinados parámetros de la línea de comandos.
- Reinicio: Sólo se iniciará después de un reinicio completo de la máquina en la que se encuentra.

2.4.2 Indicadores de entorno virtual

- Detección de hardware: Comprobación de diversos aspectos indicadores de que la máquina es virtual, por ejemplo, la dirección MAC.
- Detección de claves del registro: Relacionado con el anterior, pero enfocado en el registro de Windows, ya que por defecto hay algunas claves indicadoras de que se encuentra en una máquina virtual.

2.4.3 Temporizadores

A veces, la muestra iniciará su actividad después de que se cumpla alguna condición temporal. Éstas suelen ser de los siguientes tipos:

- Tiempo inicial de inactividad: Sólo se iniciará después de que pase un determinado tiempo (horas, días, semanas, meses) desde que se produzca su instalación.
- Ejecución en un determinado momento: Soló iniciará su actividad a una hora y/o en una fecha específica.

2.4.4 Señales provenientes del C&C

En algunas ocasiones, el malware no comenzará a desarrollar su comportamiento hasta que no reciba instrucciones específicas de su correspondiente servidor de Command and Control. En función de lo que el servidor le mande, podría comenzar a exfiltrar información, a tratar de hacer uso de una vulnerabilidad, etc. Por eso, en la mayoría de los casos, no sería suficiente con enviarle una señal cualquiera simulando ser su Command & Control.

2.4.5 Actividad del usuario

Diversos desarrollos de malware pueden ser diseñados para permanecer inactivos hasta que detecten eventos que denoten la interacción de un usuario real con el sistema. Algunos ejemplos que confirman la existencia de un usuario y, por lo tanto, de un entorno físico son:

- Actividad del ratón
- Pulsaciones de teclas
- Manipulación de ventanas

2.5 Amenaza persistente avanzada – APT

Las amenazas persistentes avanzadas, normalmente conocidas como APT (Advanced Persistent Threat), son ataques muy complejos. Estos cuentan con un estudio previo de la organización en el punto de mira y un elaborado diseño por fases para conseguir no ser detectadas. Otro de los aspectos que las caracterizan es que están totalmente personalizadas para desenvolverse en el entorno de la víctima y, en gran parte de las ocasiones, hacen uso de 0-days. Los ataques de tipo 0-day se basan en explotar vulnerabilidades no conocidas hasta entonces ni por el fabricante ni por el público en general.

Si se tienen en cuenta todas las características aportadas, es fácil llegar a la conclusión de que prácticamente todas las amenazas persistentes avanzadas son de naturaleza desconocida, nunca vistas anteriormente. Esto supone todo un reto no sólo para detectarlas, sino también para categorizarlas y analizarlas.

Se podrían ver motivadas por cualquiera de los principales propósitos descritos previamente, pero lo habitual es que su finalidad sea hallar y exfiltrar datos relativos a la propiedad intelectual de la víctima.

2.5.1 Fases

Antes de entrar en detalle con cada una de las fases, es necesario indicar que no siempre los ataques basados en amenazas persistentes avanzadas las ejecutan todas. De hecho, en ocasiones un ataque se sirve de comprar la información sobre una determinada organización obtenida mediante otros ataques previos realizados con el objetivo de comercializar los accesos conseguidos.

2.5.1.1 Reconocimiento

Se descubren las vulnerabilidades de la futura organización víctima y se obtiene toda la información posible que sea de dominio público. Se llega incluso a conseguir información personal de empleados a través de las redes sociales. Con esto se acaba elaborando una base sobre la que empezar a diseñar el malware.

2.5.1.2 Infección

Se comienza la intrusión haciendo uso de uno o varios de los vectores de ataque descritos en este capítulo. Una vez dentro, suelen establecer una puerta trasera por la que descargar malware adicional, ya que en esta fase lo habitual es que sean sencillas muestras de tipo dropper – downloader.

2.5.1.3 Operación

Se explotan las vulnerabilidades del entorno, es decir, del sistema, de la red interna, etcétera. Además, se intenta expandir la infección a la mayor parte posible de la organización buscando activos valiosos y vulnerabilidades en diferentes sistemas. Con ello se pretende hallar credenciales de usuarios con suficientes privilegios, establecer las conexiones necesarias con el exterior y desplegar todas las herramientas necesarias para llevar a cabo su cometido.

2.5.1.4 Exfiltración

Una vez que ya se tiene desplegado el malware, se ha encontrado la localización de la información sensible y se ha conseguido la forma de acceder a ella, comienza el verdadero propósito del ataque. En primer lugar, se prepara la información cifrándola y comprimiéndola. Una vez que esté lista, se envía al servidor de Command and Control.

2.5.1.5 Persistencia

Con la información sensible ya en manos de los responsables del malware y el objetivo conseguido, se eliminan las evidencias del ataque, pero no se desarticula el complejo diseño desplegado a lo largo de la organización. Al contrario, se continúa creando puertas traseras adicionales, analizando los datos recopilados, infectando nuevos sistemas, etc.

2.6 Análisis

Con los análisis de malware se pretende sacar la suficiente información de una muestra como para averiguar el riesgo que conlleva. Hay diferentes métodos para conseguir información sobre de qué tipo es, el propósito que tiene, a qué va dirigido…

Los desarrolladores de malware son conscientes de ello, por lo que se esmeran en crear mecanismos cada vez más elaborados y sofisticados para evitar ser detectados durante la realización de los análisis. No obstante, fabricantes y profesionales del sector intentan implementar todas las medidas posibles para contrarrestar a estos mecanismos y seguir detectado el mayor número de muestras posibles. Es una lucha constante, aunque habitualmente el malware es el que lleva la delantera.

La gran diversidad de opciones existentes para analizar una muestra se puede dividir en dos grandes grupos: análisis estático y análisis dinámico.

2.6.1 Estático

Este grupo de análisis se centra en sacar toda la información posible de una muestra sin ejecutarla. Los tipos de informaciones que se suelen obtener con este método son:

- Tipo de archivo
- Dependencias de librerías u otras herramientas
- Metadatos
- Cadenas que lo identifiquen
- Firmas

Precisamente las firmas son el primer valor que se trata de conseguir, ya que es la “huella digital” de un determinado archivo. Existen diferentes funciones de un solo sentido para generar la firma en base a su complejidad, algunos ejemplos son md5, sha-1 y sha-256. La mayor diferencia entre ellos es el número de bits usados para generar la firma.

Éste es el método usado por productos de seguridad como los antivirus. Si en su base de datos se encuentra la firma de la amenaza, detectan rápidamente a la muestra. En cambio, si la amenaza no es conocida por el fabricante del motor de antivirus y, por tanto, no tienen su firma, serán incapaz de detectarla.

2.6.1.1 Aspectos relevantes

Ventajas:

- Rapidez de detección si la amenaza es conocida.
- Bajos requisitos computacionales.

Inconvenientes:

- Inutilidad a la hora de detectar amenazas vistas por primera vez.
- Necesita de un tiempo de reacción por parte de los fabricantes para que generen la firma de una nueva

amenaza.

2.6.2 Dinámico

Al contrario que el análisis estático, éste grupo centra su método de obtener información en ejecutar la muestra y observar su comportamiento. Los aspectos más determinantes son los relativos a la interacción del ejecutable tanto con el sistema en el que se encuentra como con sistemas remotos a través de la red. Estas interacciones se pueden dividir en diferentes ámbitos en función de contra qué se realicen:

- Otros procesos
- Sistema de ficheros
- Registro de Windows
- Llamadas a la API del sistema
- Actividad a través de las interfaces de red

A la hora de crear un entorno que tenga como uso exclusivo analizar el comportamiento del malware, se pueden valorar diferentes planteamientos. Todos ellos se basan en el concepto de sandbox, entorno completamente aislado del resto de la infraestructura en el que poder liberar las muestras para que desarrollen toda su actividad programada.

Siendo los requisitos tener un entorno aislado siempre en el mismo punto de partida para cada nueva muestra de malware, queda claro que utilizar máquinas virtuales y sus snapshots los cubre a la perfección. No sólo eso, sino que además automatiza todo el tedioso proceso de deshacer los cambios introducidos por la última muestra analizada. Como contrapartida, hay que tener en cuenta los activadores descritos anteriormente en este capítulo. Son utilizados por el malware para saber si se encuentran en un entorno virtual y permanecer inactivos.

2.6.2.1 Aspectos relevantes

Ventajas:

- Capaces de detectar amenazas desconocidas dejando que se ejecuten libremente dentro del entorno de análisis y estudiando su comportamiento.
- Su capacidad de detección no depende directamente de actualizaciones de la base de datos de información por parte de los fabricantes.

Inconvenientes:

- El tiempo que se necesita por muestra para terminar el análisis no es despreciable. Incluso en entornos completamente automatizados puede tardar del orden de 3-5 minutos de media.
- Se necesita un hardware completo dedicado o, en el caso de ser un entorno virtual, un anfitrión con altos requisitos de procesamiento y memoria para dar cabida a las diferentes máquinas virtuales analizadoras.

2.7 Información de amenazas

Tras realizar diversos análisis, se empieza a acumular información sobre las muestras de malware conocidas. Ésta va desde firmas hasta características muy concretas del comportamiento como ficheros que escribe en un determinado directorio del sistema, entradas del registro que modifica, URLs con las que conecta, etc. Es en este momento cuando surgen cuestiones como:

- “¿En qué formato se almacena?”

- "¿Es sólo para consumo propio o puede ser útil compartirla?"
- "¿Se puede recibir información extra de amenazas de fuentes externas?"
- "¿Existe algún estándar para poder intercambiar información usando la misma notación?"

En primer lugar, sí, es muy útil compartir la información obtenida de amenazas analizadas. Lo es tanto como recibir actualizaciones de los resultados generados por fuentes de confianza que completen la nuestra. Para ello, se introduce el concepto de IOC (Indicator Of Compromise), indicador de compromiso. Hace referencia a toda aquella información relevante para describir atributos estáticos o dinámicos de una determinada muestra. Proviene de la idea de que, si se analiza un sistema y se encuentran los detalles recogidos en un indicador de compromiso concreto, se está ante una infección provocada por el malware al que hace referencia dicho indicador de compromiso.

En segundo lugar, sí, existen sintaxis ya diseñadas para describir la información recopilada sobre las amenazas. El problema es que a día de hoy el sector está en un punto demasiado prematuro en este ámbito como para tener una única sintaxis universal. Se encuentran varias opciones, todas ellas abiertas y de uso público, intentos de conseguir una estandarización. En los siguientes subapartados se dan unas breves pinceladas sobre las alternativas más extendidas.

2.7.1 OpenIOC

OpenIOC (Open Indicator Of Compromise) es un formato creado por Mandiant, una compañía perteneciente al grupo empresarial FireEye. Se puede decir que son los precursores del concepto de indicador de compromiso y, sobre todo, de la implantación de su uso a gran escala en la gestión de información sobre amenazas y la respuesta a incidentes. Está definido por un esquema XML y aunque en un primer momento era una herramienta interna de sus creadores, a día de hoy está abierto su uso.

Ventajas:

- Está más orientado a los detalles técnicos, lo que lo hace muy útil para describir los resultados de un análisis de malware.
- Hay disponibles herramientas libres para facilitar su uso desarrolladas por los propios creadores:
    - IOC Editor: Facilita la edición de los XML para generar y modificar indicadores de compromiso.
    - IOC Finder: Busca en el sistema cualquier rastro de los indicadores de compromiso que se le faciliten en una lista y reporta su presencia si los encuentra.

Inconvenientes:

- No sirve para detallar aspectos a más alto nivel como vectores de ataque y tácticas seguidos por el malware.
- Menos flexibilidad de integración. Queda patente que en sus inicios estaba enfocado en detallar la información de los productos de un solo fabricante.

2.7.2 CybOX – MAEC – STIX

Son formatos que tienen su origen en MITRE, una organización sin ánimo de lucro centrada en desarrollos tecnológicos colaborativos. Se ha apoyado en organizaciones gubernamentales de Norte América, como el CERT de los Estados Unidos, para ir planteando diversos intentos de conseguir un estándar a lo largo de los últimos años.

Comenzaron con CybOX (Cyber Observable eXpression), definido en JSON, y posteriormente desarrollaron STIX (Structured Threat Information Expression). Éste estaba basado principalmente en el primero, pero mejoraba la representación de los objetos y de las relaciones existentes entre ellos. En la actualidad, CybOX ha sido integrado por completo en STIX y ya no se recomienda su uso.

En paralelo a STIX, también se basaron en CybOX para desarrollar MAEC (Malware Attribute Enumeration

and Characterization). Está diseñado para convivir con STIX e incluso para ser combinado con él, aunque exista cierto solapamiento entre las capacidades de ambos.

Por un lado, sus creadores recomiendan usar MAEC para detallar aspectos más técnicos, a bajo nivel, como qué hace el malware y cómo lo hace. Por otro lado, recomiendan usar STIX para los aspectos de alto nivel como quién está detrás del malware y dónde ha sido utilizado. La combinación de los dos formatos es posible integrando MAEC en STIX y aplicaría cuando se desea relacionar conceptos de ambos tipos.

Una de las mayores virtudes de STIX es su flexibilidad para aprovechar la información descrita en otros formatos. De hecho, es posible integrarle varios formatos bastante dispares entre sí como OpenIOC, reglas de Snort y reglas de YARA entre otros.

2.7.3 TAXII

TAXII (Trusted Automated Exchange of Intelligence Information) Está diseñado también por MITRE y su desarrollo se relaciona directamente con el de STIX. En este caso no se trata de otro formato más para describir los atributos de las amenazas. Es un protocolo de la capa de aplicación para transferir dicha información de forma simple, segura y escalable. Funciona sobre HTTPS y la información intercambiada va escrita usando STIX.

En cuanto a la infraestructura necesaria, dispone de diferentes formas de despliegue para adaptarse lo mejor posible a las necesidades de las organizaciones. Ofrece desde un enfoque más clásico de tipo Cliente – Servidor basado en peticiones hasta una arquitectura distribuida en clientes productores, un servidor broker central y clientes consumidores suscritos para recibir todos los nuevos eventos.

3 DISEÑO DE LA ARQUITECTURA

Did you exchange a walk on part in a war
for a lead role in a cage?

Roger Waters, Pink Floyd

Una vez afianzados los conocimientos básicos sobre malware expuestos a lo largo del capítulo anterior, se empieza a trabajar en el entorno que dará lugar al sistema de análisis automatizados. Los primeros esfuerzos se centran en dilucidar qué necesidades se tienen para posteriormente llevar a cabo el diseño que satisfaga cada una de ellas.

Hay tres necesidades que se desprenden directamente del estudio de las características del malware:

- Realizar análisis estático
- Realizar análisis dinámico
- Gestionar la información sobre las amenazas.
    - Además, es importante buscar la forma de poder compartir y recibir información para complementar la generada a través de los análisis.

Existe una necesidad extra derivada del deseo de automatizar todo el proceso completo. Se trata más concretamente de identificar y extraer posibles muestras de malware a analizar de forma directa desde el tráfico de la red.

3.1 Tecnologías

En base a las necesidades descritas, se buscan tipos de tecnologías capaces de solventarlas. Durante esta búsqueda se tiene también en cuenta qué alternativas Open Source existen dentro de cada tecnología para incluir las más interesantes en el diseño del entorno.

3.1.1 Monitorización de red y extracción de ficheros

La solución basada en el diseño de este entorno, que se expondrá en el siguiente capítulo, es independiente de la forma en la que se extraigan los archivos del tráfico de red. Se incluye esta tecnología por complementar el entorno y, a su vez, aportar una alternativa viable y plenamente funcional en el caso de que no se cuente ya con un método para conseguir las muestras directamente del tráfico.

Existen diferentes maneras de obtener los archivos que recorren la red. Se opta por utilizar el método que basa la extracción de estos elementos en los tipos de contenido MIME (Multipurpose Internet Mail Extensions). Dicho concepto se define en varias “Request For Comments” del IETF (Internet Engineering Task Force), pero las más relevantes son la RFC 2045 y la RFC 2046. Lejos de lo que pueda parecer por su nombre, los tipos de contenido que define el estándar son usados extensamente en otros protocolos que poco o nada tienen que ver con el correo electrónico y SMTP, por ejemplo, HTTP y FTP.

Para llevar a cabo esta tarea, se decide utilizar Bro Network Security Monitor. Aunque cumple con muchas de las características propias de los IDS/IPS, va mucho más allá en sus capacidades y se le puede considerar como un framework de monitorización y análisis de la red. Tiene su propio lenguaje de scripting con el que ampliar las funcionalidades y de él se hace uso para configurar la extracción de archivos.

Sólo se desean extraer los tipos de archivos potencialmente peligrosos, los que suelen formar parte de los ataques relacionados con malware. Para ello, se especifica en el script una lista de valores de la cabecera “content-type” de MIME. Si Bro Network Security Monitor detecta dentro del tráfico monitorizado alguno de estos valores, extraerá el archivo en la carpeta configurada. Los protocolos soportados son HTTP, FTP, SMTP e IRC. En otra parte del script se le indica a Bro con qué tipo de extensión debe guardar el archivo en función del “content-type” en cuestión. A modo de referencia, se proporciona una lista de tipos MIME considerados interesantes de ser extraídos junto con sus extensiones asociadas:

- ["application/x-dosexec"] = "exe"
- ["application/x-msi"] = "msi"
- ["application/x-msdownload"] = "dll"
- ["application/pdf"] = "pdf"
- ["application/msword"] = "doc"
- ["application/vnd.ms-excel"] = "xls"
- ["application/vnd.ms-powerpoint"] = "ppt"
- ["application/vnd.openxmlformats-officedocument.wordprocessingml.document"] = "docx"
- ["application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"] = "xlsx"
- ["application/vnd.openxmlformats-officedocument.presentationml.presentation"] = "pptx"
- ["application/java-archive"] = "jar"
- ["application/x-java-applet"] = "jar"
- ["application/x-java-jnlp-file"] = "jnlp"
- ["application/x-shockwave-flash"] = "swf"
- ["application/javascript"] = "js"
- ["application/zip"] = "zip"

- ["application/x-rar-compressed"] = "rar"
- ["application/x-7z-compressed"] = "7z"
- ["application/x-tar"] = "tar"
- ["application/x-gzip"] = "gz"
- ["application/x-bzip"] = "bz"
- ["application/x-bzip"] = "bz2"

Para modificar esta lista con otro tipo de valores se puede consultar el enlace de IANA (Internet Assigned Numbers Authority) donde se recopila de forma actualizada los tipos registrados junto a sus referencias:

https://www.iana.org/assignments/media-types/media-types.xhtml

La instalación de Bro Network Security Monitor está soportada en las distribuciones Linux basadas en Debian y en Red Hat entre otros sistemas operativos. Incluso si se quiere tener un sensor con un gran número de herramientas para monitorizar la red, se puede utilizar la distribución Security Onion basada en Ubuntu. Ésta trae entre sus múltiples aplicaciones instaladas por defecto Bro Network Security Monitor.

El código fuente y demás información relacionada con el proyecto se encuentran en los siguientes enlaces:

https://github.com/bro/bro

https://www.bro.org

3.1.2 Motores de antivirus

La filosofía seguida en la elección de qué tipos de análisis utilizar se basa en aprovechar las ventajas que aporta cada uno de ellos, descritas en el capítulo anterior. Por lo tanto, se decide incluir un componente en el entorno capaz de realizar análisis estáticos de las muestras. Con esto se persigue poder detectar lo más rápido posible todas las amenazas conocidas, todo el malware ya publicado y analizado con anterioridad.

El concepto popularizado por VirusTotal de analizar en paralelo una muestra con múltiples motores de antivirus y devolver cuáles de ellos han detectado una infección es realmente útil en este escenario. De una sola pasada, se podría descartar las amenazas conocidas no sólo por un fabricante de antivirus, sino por una multitud de ellos.

El problema que supone usar VirusTotal está relacionado con que es un servicio “en la nube” y no existe posibilidad de tener una instancia propia. Se reservan el derecho de utilizar y compartir toda la información recopilada sobre las muestras analizadas. Eso conlleva incompatibilidades de uso con la actividad de ciertas organizaciones para velar por la privacidad y evitar la fuga de información sensible.

Se inicia la búsqueda de una herramienta que facilite la tarea de analizar archivos bajo demanda con múltiples motores de antivirus y se pueda alojar dentro del entorno diseñado. Durante este proceso se encuentran diferentes enfoques a la hora de abordar el problema, pero se decide optar por Malice. Esto se debe a que se adapta a la perfección tanto a despliegues pequeños, donde prima la optimización de recursos, como a despliegues en grandes infraestructuras, donde prima la escalabilidad y la rapidez a la hora de obtener resultados.

Malice basa su funcionamiento en orquestar la ejecución de contenedores de Docker. Cuando se le solicita el análisis de una muestra, levanta tantos contenedores como motores de antivirus haya configurados y espera a que terminen para devolver un informe con la información de todos.

Cuenta con un buen número de plugins desarrollados de forma oficial, entre los que se incluyen otros servicios y herramientas además de motores de antivirus. A la hora de configurarlo, hay que desactivar los relativos a servicios donde se deba sacar la muestra de la propia infraestructura. También hay que tener en cuenta que sólo algunos de los antivirus son Open Source y que en otros puede ser necesario el uso de una clave. De todas formas, sólo para que sirva como referencia de las opciones disponibles, se refleja a continuación la lista de los plugins oficiales relacionados con antivirus:

- Avast

- AVG
- Avira
- Bitdefender
- ClamAV
- Comodo
- DrWeb
- eScan
- F-Prot
- F-Secure
- Kaspersky
- Sophos
- Windows-Defender
- Zoner

Aunque funciona de forma estable, el proyecto está en su fase inicial. Prueba de ello es la falta de documentación más allá de los aspectos básicos, la falta de funcionalidades como una API que facilite su automatización e integración, etc. No obstante, tiene un desarrollo muy activo introduciendo numerosos cambios y junto con la buena base que ya posee lo convierten en una herramienta con mucha proyección.

Dispone de varias opciones para su instalación; desde el código fuente y binarios precompilados, hasta una versión en la que el propio programa de Malice se encuentra en un contenedor encargado de gestionar al resto.

A través del siguiente enlace se puede acceder al código fuente, a la documentación y seguir la evolución de este proyecto:

https://github.com/maliceio/malice

3.1.3 Sandboxing

Una vez que ya se han descartado todas las amenazas conocidas con el análisis estático, es el turno de contar con un análisis dinámico para detectar las nuevas amenazas, no vistas anteriormente, en base al comportamiento que desarrollen. Esta tarea supone más tiempo invertido y una mayor carga del sistema por cada muestra tratada, por lo que es importante pasar primero el filtro del análisis estático.

El hecho de ejecutar la muestra para analizar su comportamiento crea la necesidad de contar con un entorno controlado donde hacerlo no suponga un riesgo para la organización. Aquí es donde toma protagonismo el término sandboxing. En seguridad de la información, es una técnica usada para aislar a un programa proporcionándole un entorno de ejecución aislado sin importar los cambios que pudiesen ocurrir en el sistema durante el proceso.

Todos los fabricantes del sector están empezando a comercializar a gran escala productos basados en sandboxing. Sin embargo, no se encuentran muchas alternativas dentro del ámbito Open Source. Eso no significa que no exista alguna herramienta interesante basada en esta tecnología. Prueba de ello es el proyecto Cuckoo Sandbox. Nacido en 2010 y madurado a lo largo de estos años. A día de hoy, cuenta con un desarrollo muy estable y una amplia comunidad detrás, así como una fundación creada en su nombre para velar por el crecimiento tanto de la herramienta como de otros proyectos e iniciativas relacionados con ella.

La arquitectura interna de Cuckoo Sandbox consiste en un software capaz de encargarse de la realización de los análisis, recopilación de resultados, encolado de muestras pendientes, etc. Lo consigue haciendo las veces de anfitrión mediante el uso de un grupo de máquinas virtuales en las que ejecuta las muestras. Dichas máquinas virtuales las gestiona a través de una red virtual. En la siguiente figura se ilustra un ejemplo de la arquitectura de esta herramienta. La imagen original pertenece a la documentación oficial del proyecto.

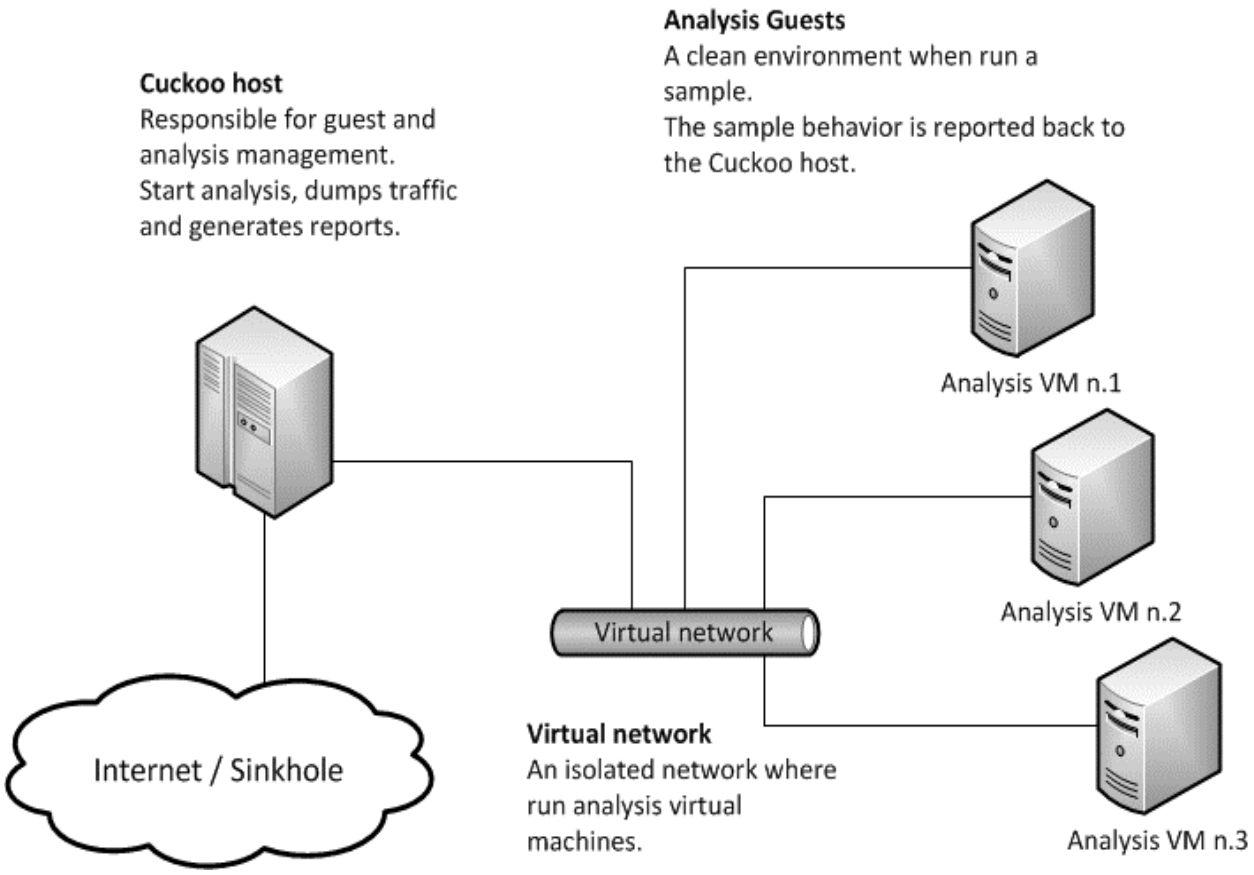


Figura 3-1. Ejemplo de la arquitectura básica de Cuckoo Sandbox

Desde sus inicios se ha diseñado de manera muy modular, por lo que es altamente personalizable a la hora de integrarse con otras herramientas para extender sus funcionalidades. Dos de los módulos más útiles son los encargados de aprovechar las ventajas de Volatility y YARA, que llevan a cabo intensivos y avanzados análisis sobre la memoria de la máquina virtual. La siguiente lista enumera los aspectos del comportamiento de la muestra de malware que Cuckoo Sandbox extrae y analiza:

- Llamadas realizadas a la API del sistema por todos los procesos generados por el malware.
- Volcados de memoria de los procesos generados por el malware.
- Volcados de la memoria completa de las máquinas virtuales.
- Archivos PCAP con el tráfico de red.
- Archivos descargados, creados y/o eliminados por los procesos generados por el malware.
- Capturas de pantalla realizadas durante el análisis.

Además de instalar y configurar el programa principal, Cuckoo Host, es necesario aplicar medidas en las máquinas virtuales, Cuckoo Guests, para evitar que las muestras de malware detecten que se encuentran en un entorno de pruebas. En función del software de virtualización que se utilice (VirtualBox, KVM, VMware, etc.) se implementan de una forma u otra. Los métodos usados para detectar si el sistema es virtualizado son principalmente los expuestos en el capítulo anterior.

Para poder probar Cuckoo Sandbox sin necesidad de desplegarla existe un servicio online creado y administrado por los propios fundadores de la herramienta. Puede servir de utilidad para familiarizarse con los resultados arrojados y para comparar muestras con su histórico de análisis llevados a cabo. Se facilita el enlace a través del que se puede acceder a él:

https://malwr.com

El código fuente, la documentación oficial y más información general sobre Cuckoo Sandbox están

disponibles en los siguientes enlaces:

https://github.com/cuckoosandbox/cuckoo

https://cuckoosandbox.org

3.1.4 Gestión e intercambio de información de amenazas

Con la información obtenida de los distintos tipos de análisis surgen dos grandes necesidades en cuanto a la automatización de la gestión que se hace de ella. Por un lado, la recolección de la información. Por otro, su adaptación y normalización en los formatos adecuados para compartirla tanto con el resto de herramientas de la infraestructura como con otras organizaciones. Ante esta tesitura, también se empieza a pensar en enriquecer la información disponible a la hora de hacer análisis con la que puedan aportar organizaciones y comunidades de confianza.

Para hacerse una idea de la relevancia que está empezando a obtener en el sector esta idea, incluso multinacionales, como IBM con su producto QRadar, están adoptando algunos de los formatos estándares descritos en el capítulo anterior (STIX / TAXII, etc.). Por ello, es de vital importancia que en el entorno que se está diseñando haya un elemento capaz de gestionar todas las tareas relativas al tratamiento de la información de amenazas.

Tras estudiar las posibles alternativas dentro del ámbito Open Source, se encuentran algunos frameworks y herramientas enfocados en este rol. Se decide usar MISP (Malware Information Sharing Platform) por las funcionalidades que ofrece y por las organizaciones que ya hacen uso intensivo de este proyecto. Lo relevante no es ya sólo que un gran número de organizaciones lo utilicen, más de 2500 según su página web oficial, sino el principal tipo de éstas que lo promocionan y lo han incluido dentro de su flujo de trabajo. Se trata de los CSIRTs más importantes a nivel global como, por ejemplo, CERT-EU, CIRCL, etc. Un CSIRT (Computer Security Incident Response Team) es un equipo encargado de monitorizar los activos y de dar respuesta a incidentes relacionados con la seguridad de la información.

MISP tiene un desarrollo modular y una documentación oficial detallada, por lo que aporta facilidad a la hora implementar modificaciones para que se integre con cualquier tipo de herramienta y formato que se necesite. No obstante, ya cuenta por defecto con un amplio número de módulos y extensiones oficiales. A parte de sus funcionalidades principales enfocadas en el intercambio de información entre organizaciones, se han encontrado realmente útil dos de las capacidades con las que cuenta:

- Exportar reglas consumibles por IDS/IPS basadas en los indicadores de compromiso disponibles en la plataforma.
- Consumir feeds de listas negras, de indicadores de compromiso o de cualquier tipo de información sobre malware y parsearlos para distribuir las actualizaciones entre las herramientas de seguridad en los formatos adecuados.

Su funcionamiento está orientado a eventos. Cada muestra subida a la plataforma, cada resultado de análisis compartido, en definitiva, cada aporte de información aparece como un evento en MISP para ser tratado por los operadores. De forma automatizada, realiza una correlación entre ellos y es capaz de relacionarlos en función de los indicadores de compromiso que posean. Cuenta con un sistema de notificaciones personalizable para estar constantemente al tanto de las actualizaciones que suceden en la plataforma. En la figura facilitada a continuación se muestran posibles usos de MISP. La imagen original pertenece a la documentación sobre MISP en la página web de CIRCL.

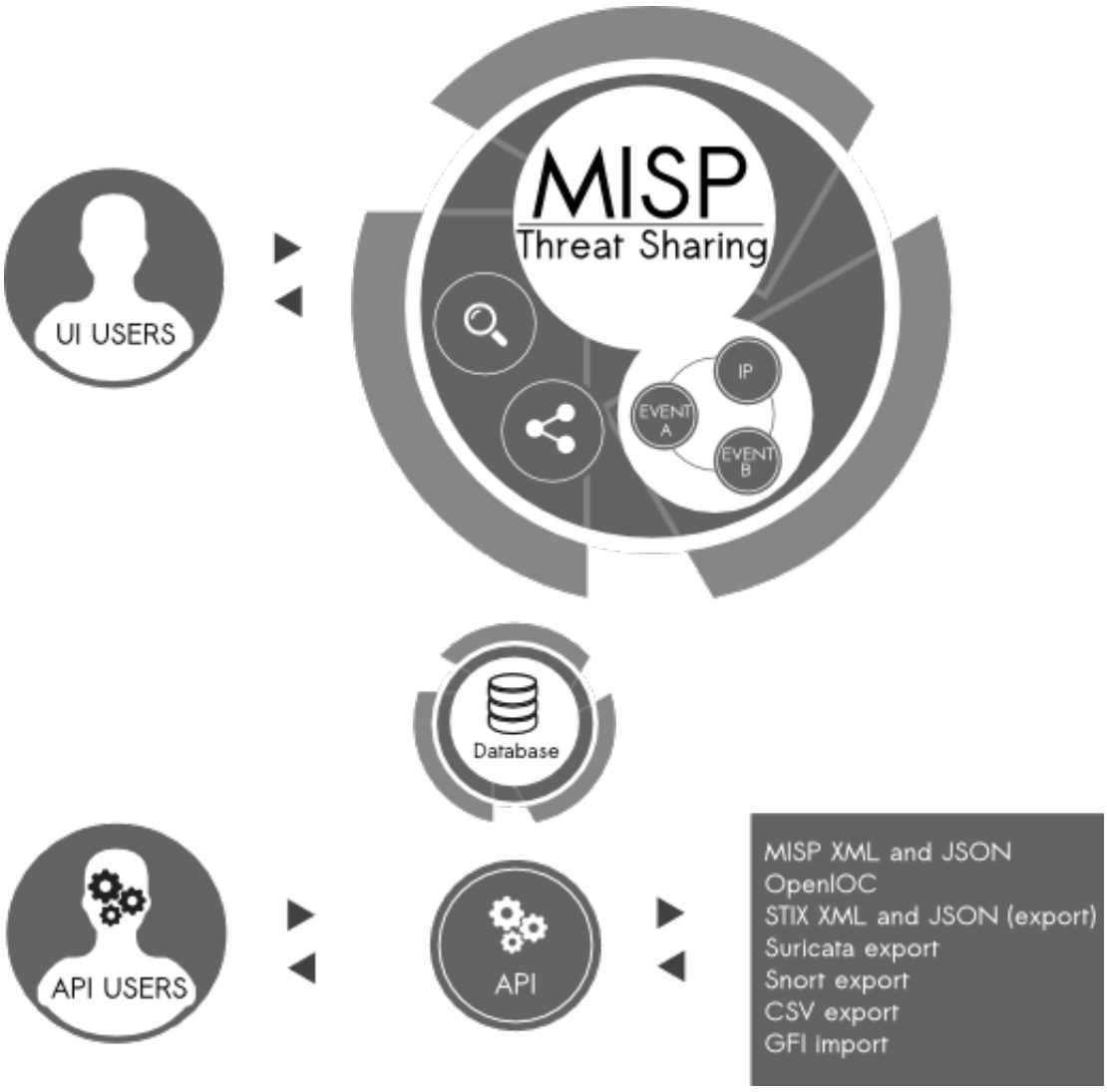


Figura 3-2. Ejemplos de uso de MISP

Por último, indicar que organizaciones como CIRCL disponen de formación para sacarle el máximo partido a su uso diario, a sus capacidades de integración, etc. El código fuente, la documentación oficial, formación y demás información relevante relacionada con el proyecto se pueden encontrar en los siguientes enlaces:

https://github.com/MISP/MISP

http://www.misp-project.org

https://www.circl.lu/doc/misp

https://www.circl.lu/services/misp-training-materials

3.2 Disposición de componentes

Se le asigna un rol dentro del entorno a las herramientas seleccionadas en base a las funciones para las que han sido desarrolladas y se obtienen cuatro componentes:

- Sensor de red → Bro Network Security Monitor
- Analizador estático → Múltiples motores de antivirus orquestados por Malice
- Analizador dinámico → Cuckoo Sandbox
- Gestor de información de amenazas → MISP

A continuación, se representan los componentes junto con las interacciones entre ellos. Esta automatización del ciclo de tratamiento de cada muestra corre a cargo de la solución detallada en el próximo capítulo.

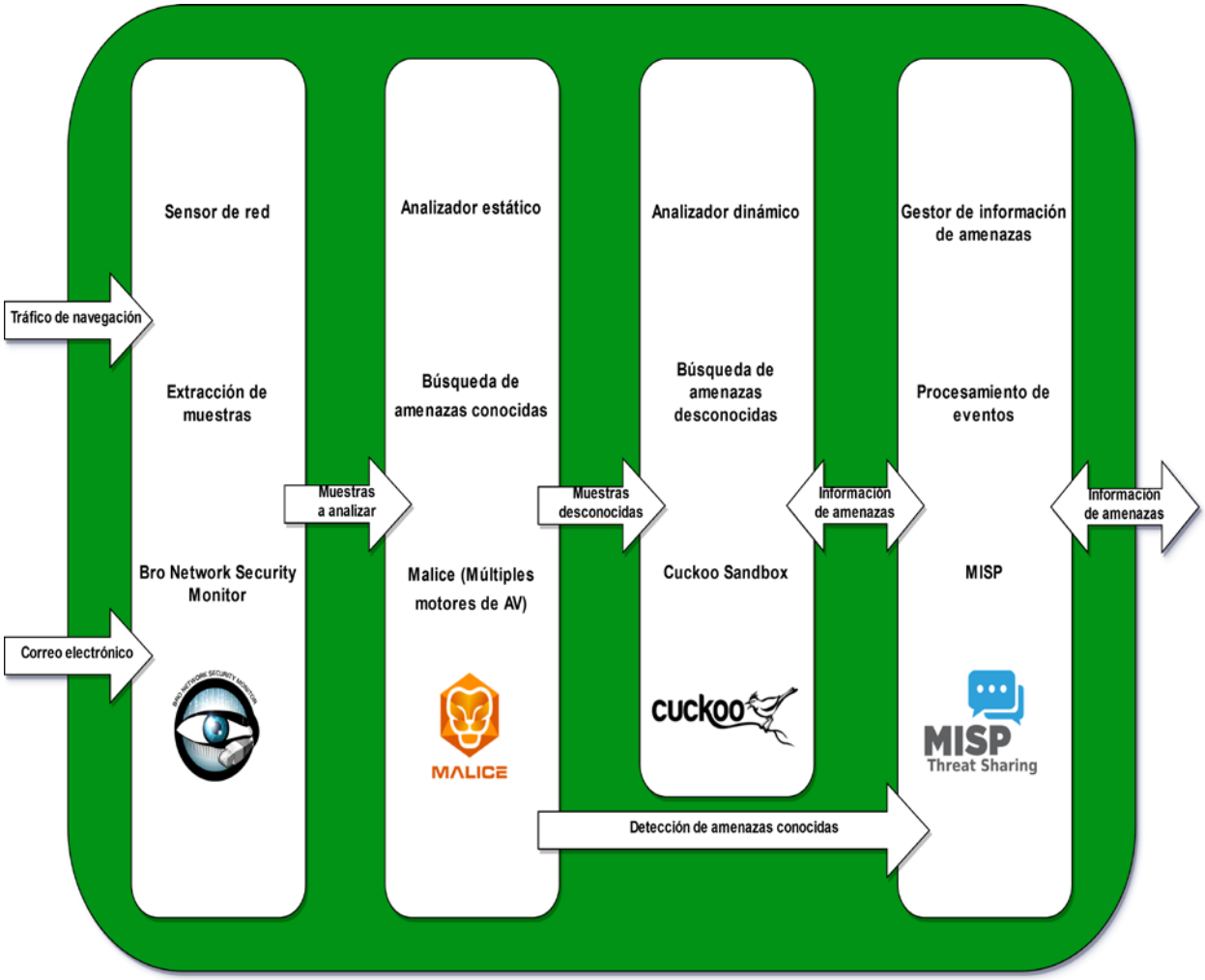


Figura 3-3. Componentes del entorno y sus interacciones

3.3 Modelo para pruebas

Una vez que las piezas claves se definen por completo, se procede a la implementación de una maqueta usando Oracle VirtualBox para llevar a la práctica todos los conceptos adquiridos. Posee una doble finalidad. Por un lado, experimentar y obtener conclusiones sobre el trabajo realizado. Por otro, servir de apoyo en pruebas de concepto y demostraciones.

Mientras se lleva cabo, se van realizando guías de los aspectos más específicos y claves. En general, siguen buenas prácticas y recomendaciones de los proyectos; así como los procedimientos más optimizados para evitar la aparición de errores comunes a la hora de instalar, configurar o poner a punto una determinada herramienta. Dichas guías se encuentran repartidas entre los anexos incluidos en la última parte de este documento.

3.3.1 Máquinas Virtuales

El modelo consta de tres máquinas virtuales entre las que se agrupan los cuatro componentes descritos en el apartado anterior. Se opta por implementar en el mismo sistema el sensor de red y el analizador estático. Esta decisión se toma para optimizar los recursos del equipo anfitrión.

Para poner en contexto la toma de dicha decisión, hay que indicar que esta implementación es un entorno de desarrollo y que la red monitorizada solo llevará tráfico cuando se vaya a realizar alguna prueba. Se ve viable que los recursos que quedan disponibles en el sensor de red asuman el consumo de Malice cuando ejecuta escáneres de antivirus sobre las muestras facilitadas.

Para las elecciones de los sistemas operativos se tienen en cuenta las recomendaciones de las diferentes herramientas. No obstante, dentro de esas recomendaciones y siempre que sea posible, se han seguido preferencias personales. En este caso, la distribución de Linux Debian y, en su defecto, derivadas de ella como Ubuntu.

3.3.1.1 Sensor de red + Analizador estático

Sistema Operativo: Ubuntu 14.04.5 LTS - Trusty

CPU Cores: 1

Memoria: 4 GB

Disco: 60 GB

Incluye:

- Bro Network Security Monitor
- Docker Community Edition
- Malice – Malware Analysis Framework
- Contenedores de Docker con motores de antivirus

3.3.1.2 Analizador dinámico

Sistema Operativo: Debian GNU/Linux 8.9 - Jessie

CPU Cores: 2

Memoria: 4 GB

Disco: 80 GB

Incluye:

- Cuckoo Sandbox Host: Analizador, aplicación web y REST API
- Cuckoo Sandbox Guest: Máquina virtual para ejecutar muestras de malware
    - Sistema Operativo: Windows 7 SP1
    - CPU Cores: 1
    - Memoria: 1 GB
    - Disco: 20 GB

3.3.1.3 Gestor de información de amenazas

Sistema Operativo: Ubuntu 16.04.2 LTS - Xenial

CPU Cores: 1

Memoria: 2 GB

Disco: 40 GB

Incluye:

- Malware Information Sharing Platform

3.3.2 Redes virtuales

Los requisitos en cuanto a conexiones son:

- Conexión a Internet de cada una de las máquinas.
    - Actualizaciones de información sobre las amenazas.
- Conexión entre las herramientas alojadas en diferentes máquinas y el anfitrión.
    - Trabajar de manera conjunta con la ayuda de la solución a desarrollar más adelante.
- Conexión interna en exclusiva entre las máquinas virtuales.
    - Monitorizar la actividad en esta conexión aislada de cara a la realización de pruebas y presentaciones

Teniendo en cuenta los dos primeros puntos a cumplir, se pensó en aprovechar una de las últimas incorporaciones de Oracle VirtualBox en cuanto a funcionalidades de red se refiere, “NAT Network”. Consiste en crear una red que toma al anfitrión como router haciendo NAT para salir hacia el exterior e incluye a todas las máquinas virtuales permitiendo visibilidad entre ellas.

Finalmente, tras implementar dicha opción, no se consigue la estabilidad deseada debido a esporádicos errores relacionados con la memoria del anfitrión y el servicio “NAT Network”. Por lo tanto, se opta por una alternativa más clásica que aporta las mismas funcionalidades, pero que añade complejidad a la configuración de las interfaces de red en las máquinas virtuales. Consiste en lo siguiente:

- Crear tantas redes de tipo “NAT” como máquinas haya que les permita salir a internet a través del anfitrión sin tener visibilidad entre ellas.
- Crear una red de tipo “Host-Only” que permita la visibilidad entre todas las máquinas virtuales y el anfitrión sin salida al exterior.

Por otra parte, para cumplir el tercer requisito en cuanto a redes virtuales, se crea una de tipo “Internal” que permite la visibilidad entre las máquinas virtuales, pero no con el anfitrión ni con el exterior.

3.3.3 Esquema del modelo

A continuación, se muestra la primera implementación mediante el uso de las funcionalidades “NAT Network” e “Internal Network” en Oracle VirtualBox:

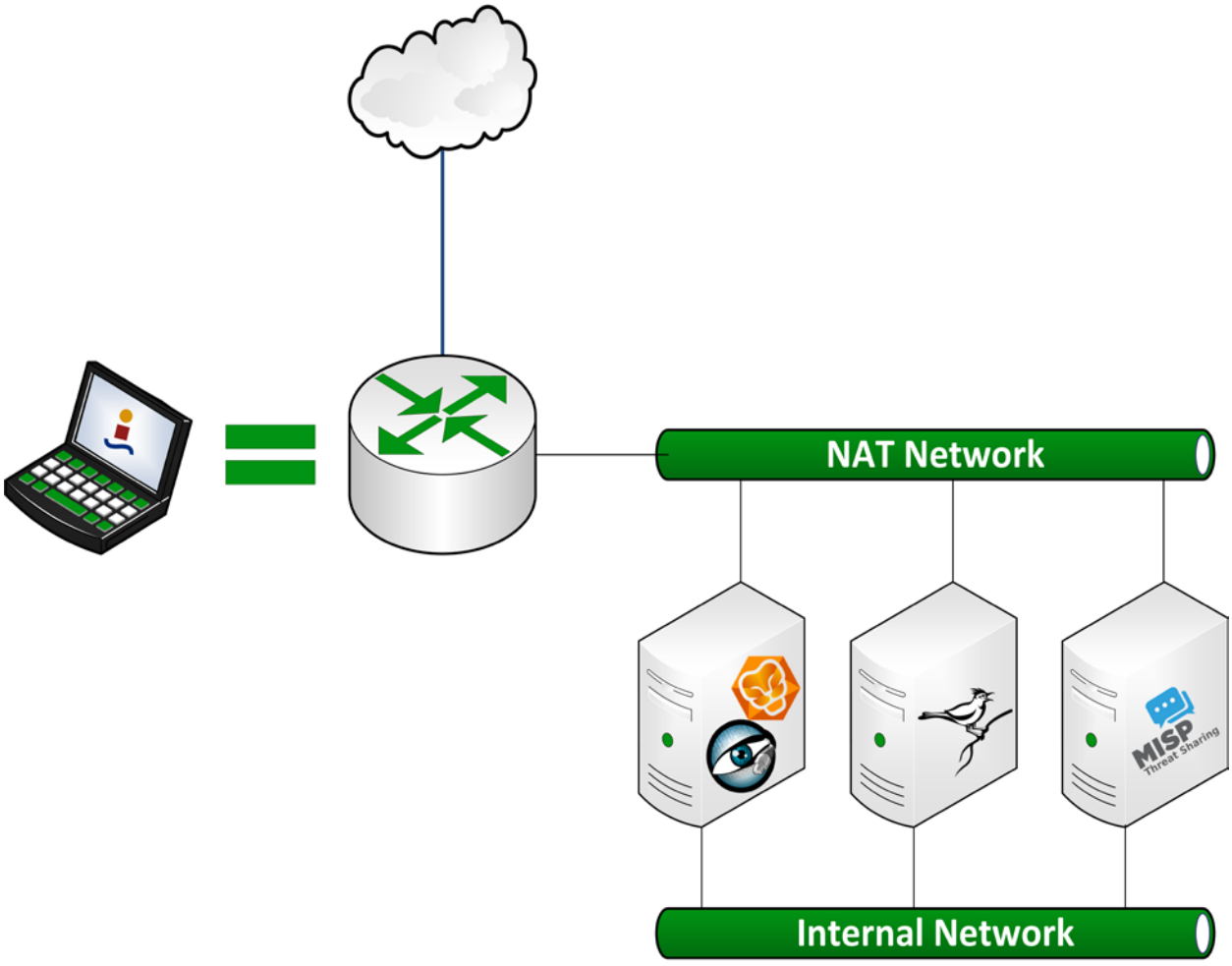


Figura 3-4. Esquema de red del modelo virtual

4 DISEÑO DE LA SOLUCIÓN

Remember when you were young, you shone like the sun.
Shine on you crazy diamond!

Roger Waters, Pink Floyd

Este primer diseño para la solución que integra las herramientas del entorno creado se ve muy condicionado por tratar de sacar el máximo partido de las actuales limitaciones de Malice. El problema reside en el envío de los archivos desde el sensor de red hacia la máquina donde se encuentre Malice. Al carecer todavía de API en el momento en el que se lleva a cabo este proyecto, la mejor opción sería enviárselos por SCP a un directorio que estuviese monitorizado ejecutando un escáner por cada nuevo archivo. No obstante, por los motivos ya detallados en el capítulo anterior se decidió poner la instancia de Malice en la misma máquina que el sensor de red y en ese escenario se basa la versión de la solución implementada.

Se trata de que la interacción humana necesaria para que el programa realice todas sus funciones sea nula desde el momento en el que se arranca con una determinada configuración inicial. Por lo tanto, se quedará a la espera de tratar nuevas muestras indefinidamente hasta que se le mande alguna de las señales del sistema operativo, por ejemplo, SIGINT usando CTRL+C.

Para que el programa sea lo más flexible posible en cuanto a las herramientas con las que ha sido diseñado para trabajar, no es obligatorio el uso de ninguna de ellas para que pueda ejecutarse correctamente. Una vez elegidos los tipos de análisis a realizar (grupo de diferentes motores de antivirus y/o una sandbox virtual), se puede incluir opcionalmente en el flujo de trabajo la creación de eventos en MISP.

En la sección de anexos al final de este documento se puede encontrar una copia íntegra del código del programa, “ninjaMind”.

4.1 Características

De los objetivos marcados y del diseño de la arquitectura se desprenden las características a implementar:

- Posibilidad de elegir entre realizar análisis estáticos, dinámicos o ambos.
- Monitorizar el directorio especificado donde se extraen los archivos de la red.
- Automatización completa del intercambio de información entre herramientas.

4.2 Atributos de calidad

Durante el proceso de diseño del programa se decide añadir una serie de atributos no recogidos en las características anteriores. En definitiva, detalles que sirven para obtener un mejor y más completo resultado final:

- Monitorización del directorio mediante las llamadas al sistema y no en base a una comprobación periódica del mismo.
- Procesamiento en paralelo de las diferentes tareas.
- Gestión de la salida estándar y de la salida de error.
- Gestión de errores.
- Gestión de subprocesos.
- Gestión de contenedores de Docker.
- Gestión del cierre del programa – Señales del sistema.

4.3 Diagrama de flujo

A continuación, se proporciona el diagrama de flujo del programa a alto nivel. Es decir, centrado en el tratamiento que se le realiza a cada nueva muestra. No se tienen en cuenta aspectos adyacentes recogidos en el código como el parseo y la validación de la configuración al iniciar el programa, la gestión de subprocesos ante una señal de terminación, la gestión errores producidos en cualquiera de los procesos, etc.:

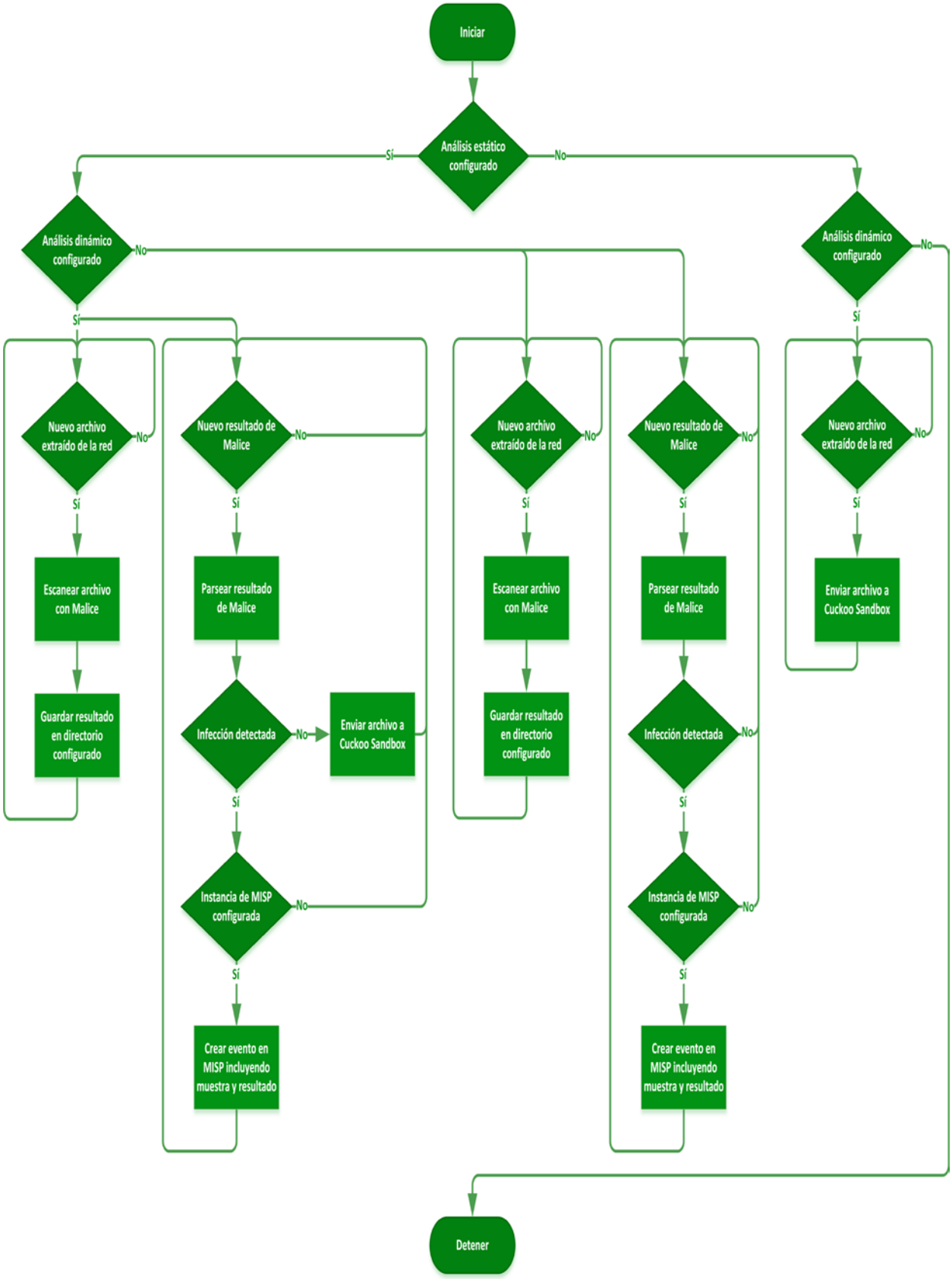


Figura 4-1. Diagrama de flujo de la solución

4.4 Programación en Bash

Se ha hecho uso de Bash, Bourne-Again Shell, para desarrollar este programa. Es una Shell de Unix, intérprete de comandos, desarrollada como parte del proyecto GNU en 1989 para sustituir a Bourne Shell. A día de hoy, es ampliamente utilizada, incluso se distribuye como el intérprete de comandos por defecto en la mayoría de las distribuciones de Linux.

Para tomar la decisión de en qué lenguaje programar, se tienen en cuenta tanto el tipo de tareas que debe realizar la solución como los potenciales sistemas donde podría acabar siendo utilizada. En este caso, la elección de Bash es consecuencia de los siguientes motivos:

- Lenguaje interpretado, es decir, el código no necesita de una compilación previa para ser ejecutado. Eso aporta una enorme agilidad en la fase de implementación a la hora ir comprobando los cambios realizados en el código.
- Integración con acciones del sistema operativo de forma transparente y eficiente. Aspecto muy importante para poder monitorizar las llamadas al sistema relacionadas con un directorio determinado en vez de ejecutar una comprobación periódica del estado del mismo.
- No tiene la necesidad de incluir numerosas librerías y dependencias para realizar las acciones descritas. Con el uso de otras alternativas, cabía la posibilidad de anclar el programa a la existencia de determinadas versiones de algunas dependencias pudiendo crear conflictos con las usadas por las herramientas.

Como se acaba de exponer, todos los comandos usados suelen estar ya presentes en el sistema, si bien es posible que haya que instalar dos paquetes en función del sistema operativo utilizado:

- curl → Dedicado a la transferencia de archivos mediante protocolos como HTTP y FTP entre otros.
- inotify-tools → Dedicado a la monitorización de los eventos del sistema de ficheros a través de las llamadas al sistema.

4.4.1 Programación defensiva

Cuando se habla de programación defensiva no se hace referencia a generar un código “limpio”, se trata de algo más, de un código seguro. Si se tuviese que resumir el concepto en una única frase, sería la siguiente:

“Prepara el código para lo improbable porque lo improbable, por definición, sucederá.”

Hay una larga lista de máximas aplicables cuando se trata de programar de forma defensiva, así que se deja a continuación la que quizás haya sido más relevante durante el desarrollo de esta solución junto con algunos ejemplos pertenecientes al código:

- No se confía nunca en la introducción de datos del usuario.
    - Si se espera un directorio como argumento, comprobar si realmente es un directorio y si se tienen los permisos necesarios para trabajar con el mismo:

```
  elif [[ ! -d "$extFilDir" ]]; then
    err "$extFilDir is not a directory"
    exit 1
  elif [[ ! (-r "$extFilDir" && -w "$extFilDir" && -x "$extFilDir") ]];
then
    err "Unable to use $extFilDir - User $USER must have rwx
permissions"
    exit 1
```

    - Si se espera una URL como argumento para usarla de raíz de diferentes llamadas a una REST API, comprobar si se le ha añadido la barra final para evitar dobles barras o la falta de barra según la implementación de dicha llamada:

```
  while getopts ":e:a:m:b:qvh" opt; do
    case "$opt" in
[…]
      m)
        [[ "$OPTARG" != */ ]] && OPTARG="$OPTARG/"
        mispURL="$OPTARG"
        ;;
      b)
        [[ "$OPTARG" != */ ]] && OPTARG="$OPTARG/"
        cuckooURL="$OPTARG"
        ;;
[…]
    esac
  done
```

Otro punto relevante es el estilo de código seguido. Conseguir que sea no sólo legible, sino también comprensible mediante el uso de las siguientes buenas prácticas, entre otras:

- Indentar adecuadamente.
- Comentar las acciones menos intuitivas.
- Crear variables y funciones con nombres que se describan por sí mismos.

Se han aplicado los conceptos vistos de forma específica a la programación en Bash y éstas son las recomendaciones que se desprenden de ello:

- Hacer todas las variables locales siempre que sea posible, por defecto son globales:

```
local extFilDir=""
local avResDir=""
local mispURL=""
local cuckooURL=""
```

- Pasar las variables a “solo lectura” una vez que se les haya asignado un valor definitivo:

```
readonly extFilDir
readonly avResDir
readonly mispURL
readonly cuckooURL
```

- Todos los comandos están agrupados dentro de funciones, incluyendo la función “main”. La única sentencia fuera de una función es la llamada a “main” con todos los argumentos de la línea de comandos:

```
main "$@"
```

- Generar un mensaje de ayuda que incluya descripción, sintaxis de la línea de comandos, índice de argumentos y ejemplos haciendo uso de las opciones más relevantes:

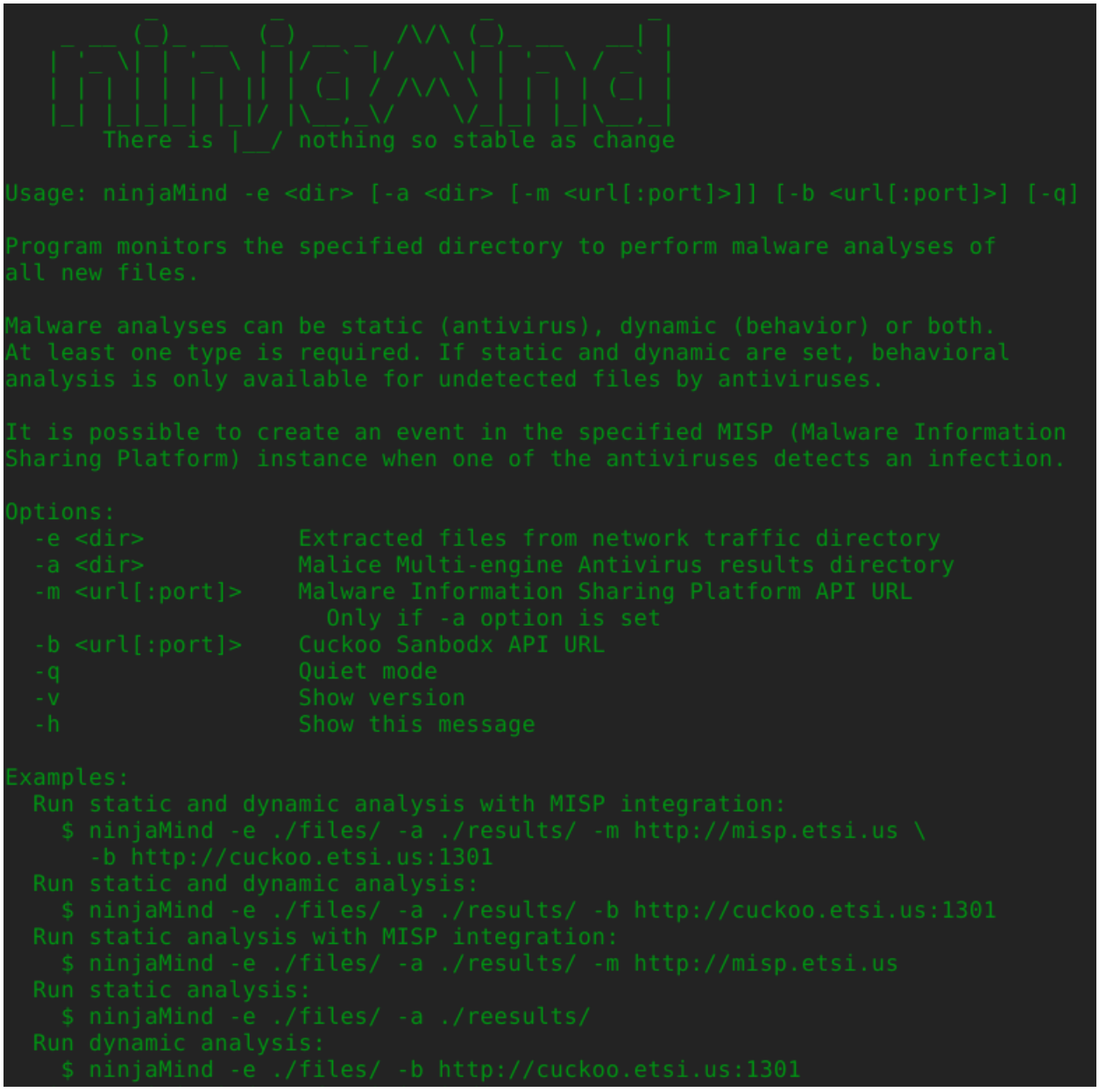


Figura 4-2. Mensaje de ayuda de ninjaMind

- Evitar el uso de la función “eval” siempre que sea posible. Sobre todo, con datos introducidos por el usuario, ya que podría ejecutarse código inyectado en forma de argumentos.

En general, a lo largo del código fuente se han hecho comprobaciones para reducir todo lo posible el número de errores y, en el caso de que se produzcan, tratarlos correctamente. De esta forma, se consigue que el programa se comporte de una manera predecible incluso ante acciones o introducciones de datos inesperadas.

4.5 Aspectos a destacar

En este apartado se verá en detalle la implementación de los atributos de calidad más específicos del programa. Varios de ellos se derivan directamente de la filosofía expuesta a lo largo del capítulo.

4.5.1 Monitorización de las llamadas al sistema dentro un directorio

En el siguiente segmento de código se monitoriza el directorio pasado como parámetro, "$1", y se genera un evento cada vez que se cierre el descriptor de un archivo contenido en él después de haber sido abierto en modo escritura. Dentro del bucle "while" se trata el archivo que ha generado dicho evento:

```
    # Monitor specified directory
    inotifywait -m -r -q -e close_write --format "%w%f" "$1" | while
read malResult; do
  […]
    Done
```

4.5.2 Gestión de la salida estándar y de la salida de error

Lo primero es diferenciar ambas salidas por las que generar los mensajes pertinentes. Después, se crean funciones específicas para imprimir el texto por cada una de ellas añadiendo si es un mensaje de información o de error junto con la fecha y la hora de su creación:

```
 ######################################
 # Print error messages along with date and time
 # Globals:
 #   quietMode
 # Arguments:
 #   1 Error message to print
 # Returns:
 #   None
 #--------------------------------------
 err() {
   if [[ -z "$quietMode" ]]; then
     echo "[ERR][$(date +'%Y/%m/%d-%H:%M:%S%z')]: $1" >&2
   fi
 }

 ######################################
 # Print information messages along with date and time
 # Globals:
 #   quietMode
 # Arguments:
 #   1 Information message to print
 # Returns:
 #   None
 #--------------------------------------
 inf() {
   if [[ -z "$quietMode" ]]; then
    echo "[INF][$(date +'%Y/%m/%d-%H:%M:%S%z')]: $1"
   fi
 }
```

4.5.3 Gestión de errores

Cuando se producen errores en las funciones, se genera un mensaje detallando el motivo. Si se encuentran dentro de un bucle precedido por "inotifywait" y se quiere escalar el error para terminar con la ejecución, es necesario romper la tubería y capturar la señal.

Si se usa “exit” únicamente, se queda en ejecución hasta que ocurre el siguiente evento del sistema de ficheros y entonces sale. En el código se explican los pasos necesarios para tratar bien los errores en esta situación mediante los comentarios que comienzan con ###:

```
  ####################################
  # Monitor spcified directory and send each new file to Cuckoo Sandbox
  # Globals:
  #   None
  # Arguments:
  #   1 Extracted files directory to monitor
  #   2 Cuckoo Sandbox API URL
  # Returns:
  #   0 --> Success
  #   1 --> Error
  #----------------------------------- #
  extrFilesMonCuckoo() {
    local extrFile=""

    ### NOTE: Follow "###" comments to manage exit properly inside
intofywait loops
    ### Use "trap" to capture "PIPE" BEFORE inotifywait loops
    trap ':' PIPE

    # Monitor specified directory
    inotifywait -m -r -q -e close_write -e moved_to --format "%w%f"
"$1" | while read extrFile; do
      ### Check errors: kill PIPE and then exit INSIDE inotifywait
loops
      # Send the file to Cuckoo Sandbox
      if ! sendFileCuckoo "$2" "$extrFile"; then
        kill -PIPE 0
        exit 1
      fi
    done

    ### Use "trap" to capture "PIPE" AFTER inotifywait loops
    trap PIPE
  }
```

4.5.4 Gestión de subprocesos

Por una parte, se guarda el PID del subproceso cuando se crea y se almacena en una lista de subprocesos:

```
    extrFilesMonMalice "$extFilDir" "$avResDir" &
    nmPIDs="$nmPIDs $!"
    maliceResultsMon "$avResDir" "$extFilDir" "$cuckooURL" "$mispURL" &
    nmPIDs="$nmPIDs $!"
```

Por otra parte, se usa la lista como argumento para la función encargada de gestionar el fin de la ejecución del programa. Termina todos los subprocesos antes de salir:

```
  ####################################
  #  Manage exit cleaning up Docker containers and killing all
subprocesses
  # Globals:
```

```
  #   None
  # Arguments:
  #   1 Malice results directory
  #   2 PIDs of the subprocesses
  # Returns:
  #   0 --> Success
  #   1 --> Error
  #-------------------------------------
  manageExit() {
    if [[ -n "$1" ]]; then
      checkAVContainers &>/dev/null
    fi

    kill "$2" &>/dev/null
    echo
    exit
  }
```

Por último, se mantiene al proceso principal en ejecución mientras los subprocesos están llevando a cabo las diversas tareas. En cuanto uno de ellos termine con errores, se acaba la ejecución del programa.

```
  # Wait for all currently active child processes
    wait -n
```

4.5.5 Gestión de contenedores de Docker

Se comprueba que los contenedores que incluyen a ElasticSearch y a Kibana están ambos corriendo antes de lanzar los escáneres de las muestras. Si están en un estado erróneo, se eliminan y se vuelven a arrancar:

```
  #####################################
  # Check and start Malice ELK Docker containers
  # Globals:
  #   None
  # Arguments:
  #   None
  # Returns:
  #   0 --> Success
  #   1 --> Error
  #-------------------------------------
  checkMaliceELKContainers() {
    local elaStatus
    local kibStatus

    # Get containers information
    elaStatus="$(docker  inspect  -f  {{.State.Status}}  malice-elastic
2>/dev/null)"
    kibStatus="$(docker  inspect  -f  {{.State.Status}}  malice-kibana
2>/dev/null)"

    #Check containers information
    # If both containers are running, do not start Malice ELK
    if [[ "$elaStatus" != "running" || "$kibStatus" != "running" ]];
then
      if [[ "$elaStatus" != "running" ]]; then
        if [[ -n "$elaStatus" ]]; then
```

```
          # If the container exists, but it is not running: remove it
          docker rm malice-elastic &>/dev/null
        fi
      else
        # If only this container is running, stop it and remove it
        docker stop malice-elastic &>/dev/null
        docker rm malice-elastic &>/dev/null
      fi

      if [[ "$kibStatus" != "running" ]]; then
        if [[ -n "$kibStatus" ]]; then
          # If the container exists, but it is not running: remove it
          docker rm malice-kibana &>/dev/null
        fi
      else
        # If only this container is running, stop it and remove it
        docker stop malice-kibana &>/dev/null
        docker rm malice-kibana &>/dev/null
      fi

      if ! malice elk &>/dev/null; then
        err "Cannot start Malice ELK Docker containers"
        return 1
      fi
    fi
  }
```

Se revisan los contenedores asociados a los motores de antivirus activados. Si alguno existiese, es porque no se ha eliminado correctamente después del último escáner, por lo que se fuerza su eliminación:

```
  ######################################
  # Check and clean up antivirus Docker containers
  # Globals:
  #   None
  # Arguments:
  #   None
  # Returns:
  #   0 --> Success
  #   1 --> Error
  #-------------------------------------
  checkAVContainers() {
    local avContList=""
    local avCont=""
    local avContStatus=""

    # Get names of all antivirus containers
    avContList="$(malice plugin list 2>/dev/null)"

    if [[ $? -ne 0 ]]; then
      err "Cannot list Malice plugins"
      return 1
    else
      # If a container exists, force the removal
      for avCont in $avContList; do
        avContStatus="$(docker  inspect  -f  {{.State.Status}}  $avCont
2>/dev/null)"
        if [[ -n "$avContStatus" ]]; then
```

```
          docker rm -f $avCont &>/dev/null
        fi
      done
    fi
  }
```

4.5.6 Gestión del cierre del programa – Señales

Se capturan ciertas señales del sistema para eliminar los motores de antivirus insertados en los contenedores de Docker y los subprocesos que estén ejecutándose antes de salir por completo. Tanto si es un error como si el usuario cancela la ejecución del programa se llevan a cabo estas acciones:

```
    # Control exit properly
    trap 'manageExit "$avResDir" "$nmPIDs"' SIGHUP SIGINT SIGQUIT
SIGTERM
```

4.6 Licencia del programa

El programa desarrollado se distribuye bajo la licencia GNU GPL v3. En las siguientes líneas se facilita una breve descripción de dicha licencia.

La Licencia Pública General de GNU es una licencia libre, sin derechos para software y otros tipos de trabajos.

Las licencias para la mayoría del software y otros trabajos prácticos están diseñadas para suprimir la libertad de compartir y modificar los trabajos. Por el contrario, la Licencia Pública General de GNU pretende garantizar su libertad para compartir y modificar todas las versiones de un programa -- y garantizar que siga siendo software libre para todos sus usuarios. Nosotros, la Fundación para el Software Libre, utilizamos la Licencia Pública General de GNU para la mayoría de nuestro software; se aplica también a cualquier otro trabajo publicado de esta manera por sus autores. También puede aplicarlo a sus programas.

Cuando hablamos de software libre, nos referimos a la libertad, no al precio. Nuestras Licencias Públicas Generales están diseñadas para asegurar que tiene la libertad de distribuir copias de software libre (y cobrar por ellas si lo desea), de recibir el código fuente o de poder obtenerlo si lo desea, de modificar el software o usar partes del mismo en nuevos programas libres y de saber que puede hacer estas cosas.

Para conseguir más información sobre la licencia y sus términos se pueden seguir los siguientes enlaces:

- Información general en castellano: https://www.gnu.org/licenses/licenses.es.html
- Licencia original en inglés: https://www.gnu.org/licenses/gpl.html

Figura 4-3. Logo licencia GNU GPL v3

5 VALIDACIÓN Y RESULTADOS

Tell me is something eluding you sunshine?
Is this not what you expected to see?

Roger Waters, Pink Floyd

Las pruebas llevadas a cabo han sido desarrolladas para verificar si se ha logrado solucionar el problema a resolver, ese que motiva la realización de este trabajo. Por dicho motivo es importante recordar lo recogido en el primer capítulo, Introducción, antes de comenzar de lleno a analizar los resultados. En última instancia lo que se buscaba era crear un entorno Open Source y estable que fuese capaz de automatizar el ciclo de tratamiento de las muestras.

5.1 Objetivo del Proyecto – Problema Inicial a Resolver

Se conciben las pruebas como herramientas para validar si el desarrollo expuesto en el capítulo anterior, Diseño de la Solución, cumple con los tres pilares fundamentales del problema a resolver: Open Source, estabilidad y automatización.

Por otra parte, se tienen en cuenta otros aspectos menores relacionados con decisiones de diseño que la solución desarrollada debe ofrecer. Se enumeran a continuación:

- Flexibilidad para elegir qué tipos de análisis se quieren ejecutar:

  - Estático + Dinámico (Combinar ambas técnicas para aprovechar las ventajas de cada una)
  - Sólo estático
  - Sólo dinámico
- Independencia de los métodos y productos usados para extraer las muestras. De esta forma, se aporta libertad para facilitar la integración con infraestructuras que ya cuenten con dispositivos IDS/IPS disponibles o productos capaces de extraer los archivos del tráfico de red.

5.2 Requisitos

En base a los objetivos descritos anteriormente se generaron al inicio del proyecto los requisitos (RQ) a verificar:

- Licencia – Open Source (L):
  - RQ-L-001: La solución usará la licencia GNU GPL v3.
- Estabilidad (E):
  - RQ-E-001: La solución hará uso del concepto de programación defensiva.
  - RQ-E-002: La solución gestionará de forma propia las salidas estándar y de error del sistema Linux.
  - RQ-E-003: La solución gestionará los errores y las excepciones que surjan durante su ejecución.
  - RQ-E-004: La solución gestionará la creación y terminación de subprocesos.
  - RQ-E-005: La solución gestionará la ejecución y eliminación de contenedores de Docker.
  - RQ-E-006: La solución gestionará el cierre del programa haciendo uso de las Señales del sistema Linux.
- Automatización (A):
  - RQ-A-001: La solución automatizará el envío en tiempo real a los analizadores de malware de las muestras extraídas de la red.
  - RQ-A-002: La solución será independiente del producto usado para extraer las muestras del tráfico de red.
  - RQ-A-004: La solución usará los diferentes tipos de analizadores de malware según la configuración establecida incluyendo las siguientes opciones: estático + dinámico, sólo estático y sólo dinámico.
  - RQ-A-004: La solución automatizará la generación en tiempo real de eventos en el gestor de información de amenazas con los resultados de los análisis de malware.

5.3 Riesgos Potenciales

De los requisitos a verificar, se desprenden los siguientes riesgos potenciales (RS) asociados a este desarrollo:

- Licencia – Open Source (L):
  - RS-L-001: La solución no se distribuye bajo la licencia especificada, GNU GPL v3.
- Estabilidad (E):
  - RS-E-001: El código de la solución distribuida no implementa las medidas pertenecientes al concepto de programación defensiva.

- RS-E-002: La solución distribuida no hace uso de funciones propias para escribir mensajes adaptados por las salidas del sistema Linux.
- RS-E-003: El código de la solución distribuida no captura los errores y excepciones que puedan surgir durante la ejecución.
- RS-E-004: La solución distribuida no termina todos los subprocesos antes de finalizar la ejecución del proceso padre.
- RS-E-005: La solución distribuida no elimina todos los contenedores de Docker utilizados antes de finalizar la ejecución.
- RS-E-006: La solución distribuida no captura las Señales del sistema Linux para gestionar el cierre del programa.

- Automatización (A):
  - RS-A-001: La solución distribuida no envía en tiempo real las muestras extraídas de la red a los analizadores de malware.
  - RS-A-002: El código de la aplicación distribuida hace uso de métodos de integración con dispositivos de red específicos para obtener las muestras extraídas del tráfico.
  - RS-A-003: La solución distribuida no permite la configuración de una o varias de las siguientes opciones de analizadores de malware: estático + dinámico, sólo estático y sólo dinámico.
  - RS-A-004: La solución distribuida no genera en tiempo real eventos en el gestor de información de amenazas con los resultados de los análisis de malware.

Como se puede comprobar en la siguiente tabla, los identificadores de los riesgos siguen la numeración del requisito al que están asociados.

Tabla 5-1. Relación requisitos-riesgos

Requisito ID	Riesgo ID
RQ-L-001	RS-L-001
RQ-E-001	RS-E-001
RQ-E-002	RS-E-002
RQ-E-003	RS-E-003
RQ-E-004	RS-E-004
RQ-E-005	RS-E-005
RQ-E-006	RS-E-006
RQ-A-001	RS-A-001
RQ-A-002	RS-A-002
RQ-A-003	RS-A-003
RQ-A-004	RS-A-004

5.4 Análisis de Riesgos

El análisis de los riesgos de la solución está compuesto por dos bloques principales. Se comienza clasificándolos en función de la prioridad que requiere cada uno a la hora de ser controlado y validado. Una vez clasificados, se procede a tomar medidas para poder mitigarlos en base a los métodos de control definidos.

5.4.1 Clasificación de Riesgos

Para llegar a obtener la prioridad asociada de todos los riesgos, se utiliza una combinación de cuatro diferentes parámetros junto con tres niveles de valores posibles por cada uno de ellos. A continuación, se encuentran las definiciones y la relación que hay entre estos y la prioridad de los riesgos.

- Severidad: Impacto en la seguridad, calidad y funcionalidades de la solución.
- Probabilidad: Posibilidad de que ocurra el riesgo potencial.
- Clase: Severidad x Probabilidad
- Detectabilidad: Posibilidad de identificar la presencia del riesgo antes de que tenga impacto sobre la solución.

Tabla 5-2. Clasificación de riesgos: Clase

		Probabilidad		
		Baja	Media	Alta
Severidad	Alta	2	1	1
	Media	3	2	1
	Baja	3	3	2

Tabla 5-3. Clasificación de riesgos: Prioridad

		Detectabilidad		
		Alta	Media	Baja
Clase	1	Media	Alta	Alta
	2	Baja	Media	Alta
	3	Baja	Baja	Media

Se parte de la definición de los requisitos y de los riesgos en las secciones anteriores para evaluar la naturaleza de los mismos. En la siguiente tabla se puede apreciar con detalle los valores tomados para cada parámetro y el resultado final de la clasificación.

Tabla 5-4. Clasificación de riesgos

ID	Severidad	Probabilidad	Clase	Detectabilidad	Prioridad
RS-L-001	Alta	Baja	2	Baja	Alta
RS-E-001	Alta	Media	1	Baja	Alta
RS-E-002	Baja	Media	3	Media	Baja
RS-E-003	Media	Media	2	Baja	Alta
RS-E-004	Media	Media	2	Media	Media
RS-E-005	Media	Media	2	Media	Media
RS-E-006	Media	Media	2	Media	Media
RS-A-001	Alta	Baja	2	Media	Media
RS-A-002	Alta	Baja	2	Baja	Alta
RS-A-003	Alta	Baja	2	Media	Media
RS-A-004	Alta	Baja	2	Media	Media

Para terminar con la clasificación, se facilita un gráfico a modo de sumario recogiendo el número de riesgos asociados a cada nivel de prioridad.

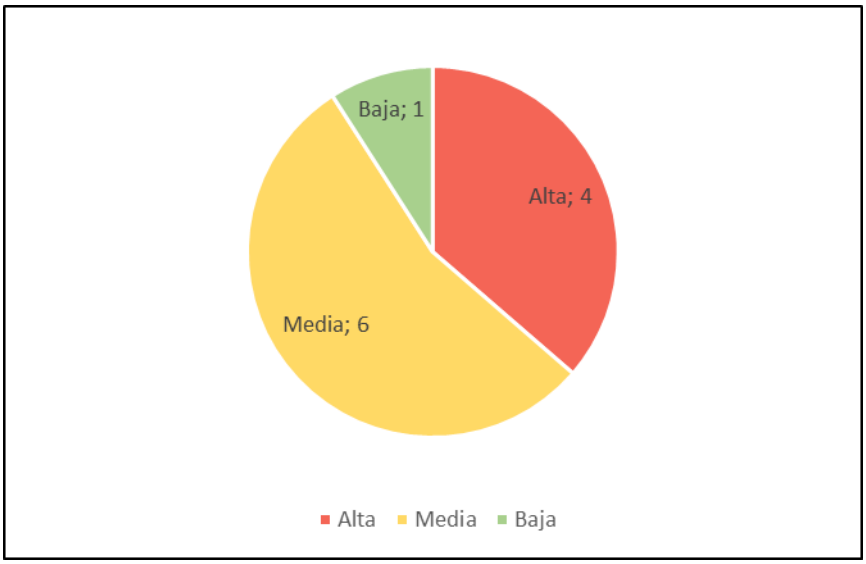


Figura 5-1. Prioridad de los riesgos: Sumario

5.4.2 Control de Riesgos

Con la clasificación de la sección anterior ya realizada, es momento de buscar formas de mitigar los riesgos. Antes de comenzar a desarrollar medidas y controlarlos, es necesario definir los métodos de control. Estos son usados como base sobre la que partir a la hora de diseñar las diferentes mitigaciones.

- Test (T):

  - Este método conlleva la ejecución de una prueba en la que el resultado obtenido muestre el cumplimiento de uno o más requisitos. Se definirán condiciones específicas para cada procedimiento de prueba a ejecutar.

- Inspección (I):
  - Por una parte, este método sirve para validar aquellos requisitos en los que realizar un test no es posible porque la comprobación solo puede hacerse a través de una inspección visual (Ej.: validar que cierta documentación o licencia está disponible). Por otra, es útil para realizar una comprobación en profundidad de código cuando la detectabilidad del riesgo asociado al requisito es baja.

En la siguiente tabla se encuentran cada una de las medidas ideadas, junto con el método en el que se basan, y la asociación de éstas con la prioridad de su correspondiente riesgo.

Tabla 5-5. Control de Riesgos

Riesgo ID	Riesgo Prioridad	Control Método	Control Medida
RS-L-001	Alta	Inspección	Comprobar en el código de la solución que existe un aviso legal sobre la licencia utilizada incluyendo una copia de la misma o una referencia para conseguirla.
RS-E-001	Alta	Inspección	Comprobar en el código de la solución que se han implementado cada una de las medidas descritas en la sección 4.4.1 de este documento.
RS-E-002	Baja	Test	Comprobar que todos los mensajes enviados por las salidas del sistema Linux utilizan el mismo formato establecido por la solución.
RS-E-003	Alta	Inspección	Comprobar en el código de la solución que se han implementado las medidas descritas en la sección 4.5.3 de este documento.
RS-E-004	Media	Test	Comprobar que no queda ninguno de los subprocesos en ejecución después del cierre del programa, proceso padre.
RS-E-005	Media	Test	Comprobar que no queda ninguno de los contenedores de Docker utilizados después del cierre del programa.
RS-E-006	Media	Test	Comprobar que el programa inicia la limpieza de subprocesos y contenedores Docker ante la recepción de las señales del sistema Linux.
RS-A-001	Media	Test	Comprobar que las muestras se envían a los analizadores de malware en cuanto son extraídas.
RS-A-002	Alta	Inspección	Comprobar en el código de la solución que se han implementado las medidas descritas en la sección 4.5.1 de este documento para obtener las muestras extraídas de la red.
RS-A-003	Media	Test	Comprobar que se pueden configurar las 3 opciones de analizadores de malware.
RS-A-004	Media	Test	Comprobar que los resultados de los analizadores de malware se envían al gestor de información de amenazas en cuanto son generados.

Como se puede apreciar, para cada riesgo con prioridad alta se ha optado por realizar una inspección en profundidad del código fuente. Este tipo de tarea exhaustiva requiere muchos más recursos que la realización de un test, por lo que se ha reservado en exclusiva para controlar aquellos riesgos altamente prioritarios.

5.5 Validación de la Solución

Esta sección incluye la información necesaria para comprobar que la solución se ajusta a los objetivos con los que fue diseñada. Los casos y procedimientos de pruebas descritos son suficientes para validar todos los requisitos y, por tanto, la solución. Todos y cada uno de ellos cuentan con un identificador único que se utiliza para generar a posteriori la matriz de trazabilidad.

5.5.1 Tareas de la sesión de validación

Las siguientes tareas se llevan a cabo durante la sesión de validación:

- Desplegar el escenario de las pruebas con los sistemas configurados y la comunicación entre los mismos disponible.
- Comprobar que todos los elementos que componen las pruebas están configurados adecuadamente.
- Ejecutar los procedimientos.
- Comparar los resultados obtenidos con los resultados esperados.

5.5.2 Criterio de aceptación

Como punto de partida, el criterio de aceptación para la validación de cada uno de los requisitos se basa en la comparación entre el resultado obtenido al ejecutar los procedimientos y los resultados esperados. Si ambos son iguales, se puede considerar como validado el/los requisito/s asociado/s. Un hecho a tener en cuenta, aunque pueda parecer trivial, es que los resultados esperados deben ser definidos con el objetivo de cumplir con los requisitos y con anterioridad a la fase de ejecución de los procedimientos.

El criterio de aceptación se detalla por separado para cada uno de los diferentes procesos a ejecutar.

5.5.3 Escenario de las Pruebas

El escenario en el que se lleva a cabo la validación es el descrito en la sección 3.3, Modelo para pruebas. En la siguiente figura se ilustra la disposición tanto de las redes como de los sistemas utilizados.

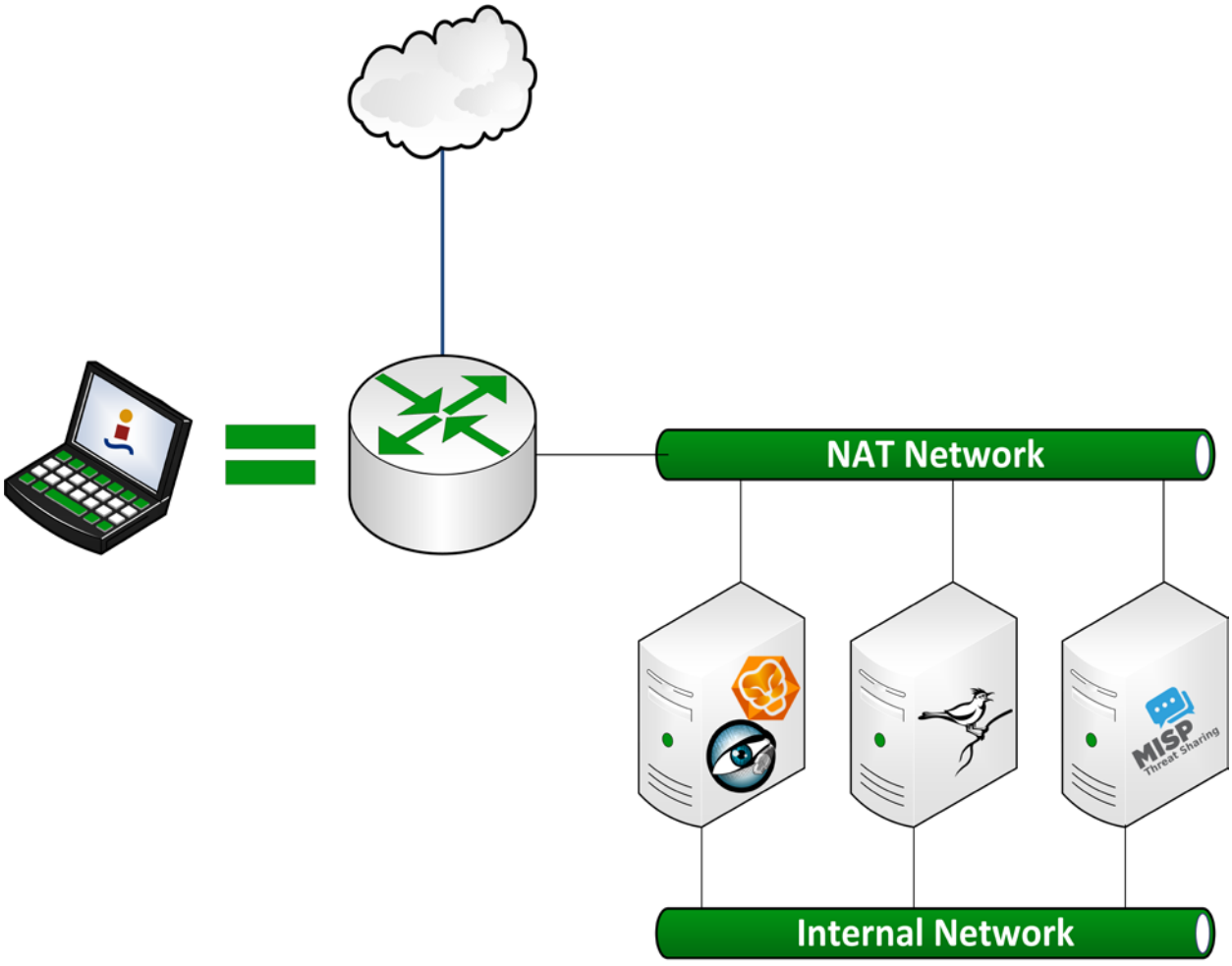


Figura 5-2. Escenario de las pruebas

5.5.4 Formato de los casos y procedimientos

Cada prueba, formada por un caso y uno o varios procedimientos, se presenta en una tabla como la siguiente:

Tabla 5-6. Formato de los casos

Id. Caso	*PR-<Método de control>-<DD>* *Método de control: T (Test) o I (Inspección).* *DD es un valor decimal de dos dígitos único dentro de su método de control.* *Ejemplo: PR-T-01*
Objetivo	*Objetivo de la prueba*
Descripción	*Descripción de la actividad y la configuración de la prueba.*
Id. Req.	*Hace referencia a los requisitos que se validan con la prueba, todos los requisitos que validen los procedimientos pertenecientes a este caso.*
Prerrequisitos	*Si hay algún prerrequisito adicional para la prueba, se menciona aquí. Ej.: Una configuración específica sólo para la prueba.*

Tabla 5-7. Formato de los procedimientos

Id. Procedimiento	*PR-<Método de control>-<DD>-<DDD>* *Método de control: T (Test) o I (Inspección).* *DD es un valor decimal de dos dígitos único dentro de su método de control.* *DD es un valor decimal de tres dígitos único dentro de su Caso.* *Ejemplo: PR-T-01-001*
Id. Req.	*Hace referencia a los requisitos que se validan con el procedimiento.*
Prerrequisitos	*Si hay algún prerrequisito adicional para el procedimiento, se menciona aquí. Ej.: Una configuración específica sólo para el procedimiento*
Criterio de aceptación	*Condición bajo la que el procedimiento se considera exitoso o fallido.*
Resultado del procedimiento	Exitoso ☐ Fallido ☐

Resultado del caso	Exitoso ☐	Fallido ☐

5.5.5 Casos, procedimientos y resultados

Un prerrequisito general derivado de las tareas descritas en la sección 5.5.1 es que el escenario de las pruebas, Figura 5-2, y la solución desarrollada, Anexo E: ninjaMind, estén desplegados por completo y debidamente configurados.

5.5.5.1 PR-I-01

Tabla 5-8. Caso PR-I-01

Id. Caso	PR-I-01
Objetivo	Este caso valida que existe un aviso legal sobre la licencia utilizada, GNU GPL v3, incluyendo una copia de la misma o una referencia para conseguirla.
Descripción	Inspección del código fuente completo de la solución.
Id. Req.	RQ-L-001
Prerrequisitos	Ninguno adicional.

5.5.5.1.1 PR-I-01-001

Tabla 5-9. Procedimiento PR-I-01-001

Id. Procedimiento	PR-I-01-001
Id. Req.	RQ-L-001
Prerrequisitos	Ninguno adicional.
Criterio de aceptación	El procedimiento debe ser considerado como exitoso si se encuentra en el código fuente: - Un aviso legal sobre la licencia GNU GPL v3 (o una versión posterior). - Una copia completa de la licencia GNU GPL v3 (o una versión posterior) o una referencia para obtenerla.
Resultado del procedimiento	Exitoso ☒ Fallido ☐

Resultado del caso	Exitoso ☒ Fallido ☐

5.5.5.2 PR-I-02

Tabla 5-10. Caso PR-I-02

Id. Caso	PR-I-02
Objetivo	Este caso valida que la solución hace uso del concepto de programación defensiva a lo largo de todo el código fuente.
Descripción	Inspección del código fuente completo de la solución.
Id. Req.	RQ-E-001
Prerrequisitos	Ninguno adicional.

5.5.5.2.1 PR-I-02-001

Tabla 5-11. Procedimiento PR-I-02-001

Id. Procedimiento	PR-I-02-001
Id. Req.	RQ-E-001
Prerrequisitos	Ninguno adicional.
Criterio de aceptación	El procedimiento debe ser considerado como exitoso si se encuentran implementadas todas las medidas descritas en la sección 4.4.1 a lo largo del código fuente completo de la solución.
Resultado del procedimiento	Exitoso ☒ Fallido ☐

Resultado del caso	Exitoso ☒	Fallido ☐

5.5.5.3 PR-I-03

Tabla 5-12. Caso PR-I-03

Id. Caso	PR-I-03
Objetivo	Este caso valida que la aplicación gestiona los errores y las excepciones que surgen durante su ejecución.
Descripción	Inspección del código fuente completo de la solución.
Id. Req.	RQ-E-003
Prerrequisitos	Ninguno adicional.

5.5.5.3.1 PR-I-03-001

Tabla 5-13. Procedimiento PR-I-03-001

Id. Procedimiento	PR-I-03-001	
Id. Req.	RQ-E-003	
Prerrequisitos	Ninguno adicional	
Criterio de aceptación	El procedimiento debe ser considerado como exitoso si se encuentran implementadas todas las medidas descritas en la sección 4.5.3 a lo largo del código fuente completo de la solución.	
Resultado del procedimiento	Exitoso ☒	Fallido ☐

Resultado del caso	Exitoso ☒	Fallido ☐

5.5.5.4 PR-I-04

Tabla 5-14. Caso PR-I-04

Id. Caso	PR-I-04
Objetivo	Este caso valida que la solución es independiente del producto usando para extraer las muestras del tráfico de red.
Descripción	Inspección del código fuente completo de la solución.
Id. Req.	RQ-A-002
Prerrequisitos	Ninguno adicional.

5.5.5.4.1 PR-I-04-001

Tabla 5-15. Procedimiento PR-I-04-001

Id. Procedimiento	PR-I-04-001
Id. Req.	RQ-A-002
Prerrequisitos	Ninguno adicional.

Criterio de aceptación	El procedimiento debe ser considerado como exitoso si se encuentra implementada en el código fuente de la solución la medida descrita en la sección 4.5.1 cuando se obtienen las muestras a analizar.
Resultado del procedimiento	Exitoso ☒ Fallido ☐

Resultado del caso	Exitoso ☒ Fallido ☐

5.5.5.5 PR-T-01

Tabla 5-16. Caso PR-T-01

Id. Caso	PR-T-01
Objetivo	Este caso valida que la solución: - Gestiona de forma propia las salidas del sistema Linux - Gestiona la creación y terminación de subprocesos. - Gestiona la ejecución y eliminación de contenedores de Docker. - Gestiona el cierre del programa haciendo uso de las señales del sistema Linux. - Automatiza el envío de muestras en tiempo real a los analizadores de malware. - Es configurable a la hora de elegir qué tipos de analizadores de malware usar en cada ejecución incluyendo al menos estas opciones: estático + dinámico, sólo estático y sólo dinámico. - Automatiza el envío en tiempo real de los resultados de los análisis de malware al gestor de amenazas.
Descripción	Se ejecuta la solución tres veces utilizando las siguientes configuraciones de los analizadores de malware: - Estático + Dinámico - Estático - Dinámico Al realizar estas tres ejecuciones se analiza el comportamiento del programa poniendo especial atención en los siguientes aspectos: - Mensajes enviados por las salidas del sistema Linux. - Subprocesos durante y después de la ejecución del programa. - Contenedores de Docker durante y después de la ejecución del programa. - Reacción al recibir señales del sistema Linux. - Envío de las muestras a los analizadores de malware configurados. - Envío de los resultados de los análisis de malware al gestor de información de amenazas.
Id. Req.	RQ-E-002 RQ-E-004 RQ-E-005 RQ-E-006 RQ-A-001 RQ-A-003 RQ-A-004
Prerrequisitos	Ninguno adicional.

5.5.5.5.1 PR-T-01-001

Tabla 5-17. Procedimiento PR-T-01-001

Id. Procedimiento	PR-T-01-001
Id. Req.	RQ-E-002
Prerrequisitos	Ninguno adicional.

Criterio de aceptación	El procedimiento debe ser considerado como exitoso si todos los mensajes enviados por las salidas del sistema utilizan el mismo formato establecido por la solución.
Resultado del procedimiento	Exitoso ☒ Fallido ☐

5.5.5.5.2 PR-T-01-002

Tabla 5-18. Procedimiento PR-T-01-002

Id. Procedimiento	PR-T-01-002
Id. Req.	RQ-E-004
Prerrequisitos	Ninguno adicional.
Criterio de aceptación	El procedimiento debe ser considerado como exitoso si no queda ninguno de los subprocesos en ejecución después del cierre del programa, proceso padre.
Resultado del procedimiento	Exitoso ☒ Fallido ☐

5.5.5.5.3 PR-T-01-003

Tabla 5-19. Procedimiento PR-T-01-003

Id. Procedimiento	PR-T-01-003
Id. Req.	RQ-E-005
Prerrequisitos	Ninguno adicional.
Criterio de aceptación	El procedimiento debe ser considerado como exitoso si no queda ninguno de los contenedores de Docker utilizados después del cierre del programa.
Resultado del procedimiento	Exitoso ☒ Fallido ☐

5.5.5.5.4 PR-T-01-004

Tabla 5-20. Procedimiento PR-T-01-004

Id. Procedimiento	PR-T-01-004
Id. Req.	RQ-E-006
Prerrequisitos	Ninguno adicional.
Criterio de aceptación	El procedimiento debe ser considerado como exitoso si el programa inicia la limpieza de subprocesos y contenedores Docker ante la recepción de las señales del sistema Linux.
Resultado del procedimiento	Exitoso ☒ Fallido ☐

5.5.5.5.5 PR-T-01-005

Tabla 5-21. Procedimiento PR-T-01-005

Id. Procedimiento	PR-T-01-005
Id. Req.	RQ-A-001
Prerrequisitos	Ninguno adicional.
Criterio de aceptación	El procedimiento debe ser considerado como exitoso si las muestras se envían a los analizadores de malware en cuanto son extraídas.
Resultado del procedimiento	Exitoso ☒ Fallido ☐

5.5.5.5.6 PR-T-01-006

Tabla 5-22. Procedimiento PR-T-01-006

Id. Procedimiento	PR-T-01-006
Id. Req.	RQ-A-003
Prerrequisitos	Ninguno adicional.
Criterio de aceptación	El procedimiento debe ser considerado como exitoso si se pueden configurar las 3 opciones de analizadores de malware.
Resultado del procedimiento	Exitoso ☒ Fallido ☐

5.5.5.5.7 PR-T-01-007

Tabla 5-23. Procedimiento PR-T-01-007

Id. Procedimiento	PR-T-01-007
Id. Req.	RQ-A-004
Prerrequisitos	Ninguno adicional
Criterio de aceptación	El procedimiento debe ser considerado como exitoso si los resultados de los analizadores de malware se envían al gestor de información de amenazas en cuanto son generados.
Resultado del procedimiento	Exitoso ☒ Fallido ☐

Resultado del caso	Exitoso ☒ Fallido ☐

Fin de la sesión de validación

5.6 Aceptación de la Solución

Tras haber realizado el proceso completo de análisis de los riesgos y validación de la solución, se tiene toda la información necesaria para poder decidir si la aplicación distribuida en esta entrega cumple con los objetivos del proyecto. En otras palabras, confirmar si el problema inicial a resolver que motiva este trabajo queda resuelto.

En la siguiente tabla se muestra el resumen con los resultados de cada uno de los puntos anteriores. De izquierda a derecha se puede apreciar de forma rápida el proceso que se ha seguido con cada requisito durante su validación y el resultado final de ésta.

Tabla 5-24. Aceptación de la solución: Matriz de trazabilidad

Requisito ID	Riesgo ID	Riesgo Prioridad	Control Método	Control Medida	Validación Procedimiento	Validación Resultado
RQ-L-001	RS-L-001	Alta	Inspección	Comprobar en el código de la solución que existe un aviso legal sobre la licencia utilizada incluyendo una copia de la misma o una referencia para conseguirla.	PR-I-01-001	Exitoso
RQ-E-001	RS-E-001	Alta	Inspección	Comprobar en el código de la solución que se han implementado cada una de las medidas descritas en la sección 4.4.1 de este documento.	PR-I-02-001	Exitoso
RQ-E-002	RS-E-002	Baja	Test	Comprobar que todos los mensajes enviados por las salidas del sistema Linux utilizan el mismo formato establecido por la solución.	PR-T-01-001	Exitoso
RQ-E-003	RS-E-003	Alta	Inspección	Comprobar en el código de la solución que se han implementado las medidas descritas en la sección 4.5.3 de este documento.	PR-I-03-001	Exitoso
RQ-E-004	RS-E-004	Media	Test	Comprobar que no queda ninguno de los subprocesos en ejecución después del cierre del programa, proceso padre.	PR-T-01-002	Exitoso
RQ-E-005	RS-E-005	Media	Test	Comprobar que no queda ninguno de los contenedores de Docker utilizados después del cierre del programa.	PR-T-01-003	Exitoso
RQ-E-006	RS-E-006	Media	Test	Comprobar que el programa inicia la limpieza de subprocesos y contenedores Docker ante la recepción de las señales del sistema Linux.	PR-T-01-004	Exitoso
RQ-A-001	RS-A-001	Media	Test	Comprobar que las muestras se envían a los analizadores de malware en cuanto son extraídas.	PR-T-01-005	Exitoso
RQ-A-002	RS-A-002	Alta	Inspección	Comprobar en el código de la solución que se han implementado las medidas descritas en la sección 4.5.1 de este documento para obtener las muestras extraídas de la red.	PR-I-04-001	Exitoso
RQ-A-003	RS-A-003	Media	Test	Comprobar que se pueden configurar las 3 opciones de analizadores de malware.	PR-T-01-006	Exitoso
RQ-A-004	RS-A-004	Media	Test	Comprobar que los resultados de los analizadores de malware se envían al gestor de información de amenazas en cuanto son generados.	PR-T-01-007	Exitoso

Como punto final de este proceso, se muestra a continuación un gráfico a modo de sumario sobre los resultados de la validación.

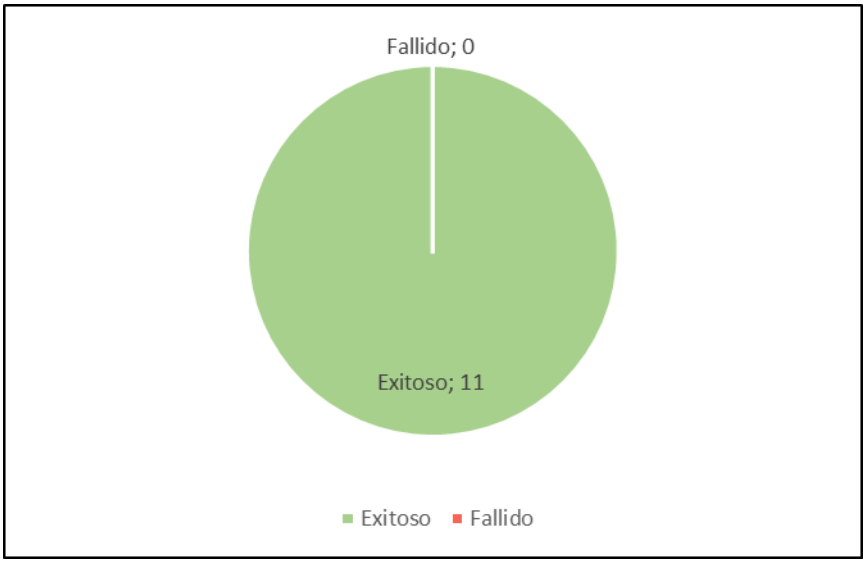


Figura 5-3. Validación de los requisitos: Sumario

Cada uno de los requisitos ha sido validado con éxito. Por lo tanto, se puede afirmar que esta entrega de la solución cumple con todos los requisitos y que todos los riesgos han sido controlados en función de su prioridad. Además, de este hecho se deduce que cada uno de los objetivos del proyecto se ven satisfechos y que se ha resuelto el problema inicial.

6 LÍNEAS DE MEJORA

When at last the work is done, don't sit down.
It's time to dig another one.

Roger Waters, Pink Floyd

Desde un primer momento, la idea era usar los conocimientos adquiridos y el entorno diseñado como base para desarrollar una futura solución Open Source. Un producto apto para ser usado en el ámbito empresarial. Una vez alcanzados los objetivos propuestos, no se acaba el trabajo, comienza el trazado de la hoja de ruta.

6.1 Funcionalidades de la solución desarrollada

Para aumentar la utilidad de la solución, se considera interesante implementar las siguientes evoluciones sobre el funcionamiento de ninjaMind explicado en apartados anteriores.

- Extraer también las URLs desde los sensores de red e incluirlas en el flujo de análisis. Por una parte, comparándolas con feeds de listas negras debidamente actualizados y por otra, analizando el comportamiento al acceder a ellas dentro de la sandbox.
- Dar la opción de configurar los parámetros desde un archivo además de poder usar los argumentos en la línea de comandos como sucede ahora mismo.

- Implementar nuevas posibilidades de configuración:
    - Indicar la ruta hacia un fichero de log para dejar constancia de los detalles de la ejecución. Esta configuración debe ser independiente de que se haya activado o no la opción para que se muestren los mensajes por la salida estándar.
    - Indicar el número mínimo de análisis de antivirus que deben concluir con “infectado” para crear un evento en MISP. Por tanto, si se activa el análisis de comportamiento, también se establecería el número mínimo de resultados indicando “infectado” que evitaría reenviar el archivo o la URL a Cuckoo Sandbox.
- Usar la futura API de Malice, que está actualmente en desarrollo, para poder interactuar mejor con la información generada por los motores de antivirus.
- Indicar en los eventos creados en MISP la relación entre el número de resultados que indiquen “infectado” y el total de análisis antivirus realizados a la muestra. Ej.: “1 / 13”.

6.2 Funcionalidades de las herramientas del entorno

Además, para mejorar integración de las herramientas utilizadas con el entorno se pretende:

- Crear un nuevo módulo para Malice que consulte el registro de muestras almacenado en MISP y devuelva “infectado” si encuentra una coincidencia. Es decir, se trata de que la comunicación entre MISP y Malice no sea unidireccional y éste último pueda realimentarse de toda la información sobre amenazas disponibles en la instancia de MISP.
- Modificar los módulos tanto de “reporting” como de “processing” que posee Cuckoo Sandbox relacionados con MISP. Esta modificación tiene como objetivo poder utilizar más parámetros de PyMISP en los archivos de configuración de Cuckoo Sandbox. De esta manera se conseguiría añadir más flexibilidad a la hora de personalizar la integración entre ambos productos. Ejemplos de ello serían: el nivel de distribución y la etiqueta del evento, la lista de direcciones IP y URLs almacenadas en MISP que no se quieren procesar, etc.

6.3 Arquitectura

El principal cambio es introducir Apache Kafka como hilo conductor en el intercambio de información consiguiendo así separar completamente las funciones de sensor de red y de analizador de malware.

Por una parte, integrar un productor de Apache Kafka en cada sensor para que envíe las URLs y los archivos codificados en base64 al broker.

Por otra parte, tener un consumidor de Apache Kafka en cada instancia de analizador que reciba del broker las URLs y archivos decodificando estos últimos para ejecutar los análisis correspondientes.

De esta forma se consigue flexibilidad a la hora de decidir en qué partes de la red desplegar qué funciones y escalabilidad a la hora de decidir cuantas instancias de cada tipo usar en paralelo. Dicha evolución iría enfocada a despliegues en infraestructuras con un gran volumen de tráfico y/o con alta probabilidad de encontrar muestras potencialmente peligrosas.

6.4 Apariencia

En el ámbito empresarial, es una práctica habitual realizar una prueba de concepto de una solución y sus alternativas antes de decidirse por implantar una de ellas. Durante la realización de estas pruebas, se evalúa principalmente su utilidad. Ante una posible equidad en los resultados, algunos de los aspectos a tener en

cuenta para decantarse finalmente por una opción u otra son la facilidad de uso para los operadores y la sensación de estar ante un producto “maduro”.

Teniendo en cuenta este hecho, se plantea modificar las diferentes interfaces gráficas para ofrecer una imagen homogénea de la solución. En definitiva, mostrar una identidad propia incluyendo nombre, eslogan, logotipo y colores.

7 CONCLUSIONES

The time is gone, the song is over.
Thought I'd something more to say.

Roger Waters, Pink Floyd

Durante la realización de este trabajo, he puesto en práctica una amplia gama de conocimientos adquiridos durante el Grado de Ingeniería de las Tecnologías de Telecomunicación. En concreto, los relacionados con el departamento de Ingeniería Telemática. Ésta fue una de las razones por las que me sentí tan motivado al plantear la organización del proyecto; debería aunar conocimientos sobre redes, sobre sistemas operativos y sobre programación.

La ciberseguridad es un ámbito multidisciplinar y así se ha hecho constar a lo largo del trabajo. Estar familiarizado con la distribución del sistema de capas OSI, por ejemplo, es esencial. De ahí que la idea inicial fuese un punto de partida perfecto para abrir el abanico de materias de las que nutrirme al comenzar mi investigación.

Aspectos de vital importancia en el mundo laboral se han visto potenciados debido a los retos afrontados para conseguir los objetivos propuestos. Un ejemplo de ello es la mejora de la autonomía, la capacidad de autocrítica y, sobre todo, la capacidad de resolver problemas a la hora de sacar el trabajo adelante.

El paquete formado por el entorno y la solución diseñados cumple con las expectativas iniciales. Si se modifica éste en los puntos detallados en la hoja de ruta y se adapta a la infraestructura de cada organización interesada, se podría comercializar ofreciendo los servicios profesionales necesarios para su instalación, configuración, puesta a punto y/o administración.

7.1 Malice – Malware Analysis Framework

Se encuentra en una fase temprana de desarrollo. Aun así, ya aporta una serie de funcionalidades muy interesantes como se ha visto a lo largo del proyecto y una estabilidad suficiente como para ser usado de forma intensiva.

Mención especial hay que hacer a la hoja de ruta de Malice. Por una parte, se va a ir ampliando el número de módulos soportados para poder analizar archivos simultáneamente con un mayor número de motores de antivirus.

Por otra parte, la futura API va a ser de enorme utilidad de cara a automatizar la ejecución de escáneres. La ampliación de la documentación disponible también facilitará el desarrollo de módulos propios para cubrir necesidades concretas de nuestro entorno.

En definitiva, es una herramienta muy interesante con una evolución prometedora.

7.2 Cuckoo Sandbox

Tiene una gran comunidad de usuarios detrás y un desarrollo muy maduro. Se aplican de forma ágil numerosas mejoras de rendimiento y corrección de errores con actualizaciones menores. Mientras, se van acumulando nuevas funcionalidades para ser liberadas en grandes actualizaciones.

Dentro de las mejoras incorporadas en la última actualización de la rama estable, dos se han usado extensamente a lo largo de este trabajo. Por un lado, el nuevo enfoque sobre los procedimientos usados para configurar el Cuckoo Host, Cuckoo Working Directory, que facilita en gran medida el proceso de actualización y de probar diferentes configuraciones dentro de la misma instancia. Por otro lado, los módulos de procesado y reporte para habilitar el intercambio de información de amenazas con MISP.

Con las modificaciones indicadas en capítulos anteriores relativas a las medidas “Anti-AntiVM” es una perfecta alternativa a las sandboxs de productos propietarios, incluso para entornos productivos.

7.3 MISP (Malware Information Sharing Platform)

Su desarrollo se encuentra respaldado por importantes organizaciones encargadas de gestionar información sobre amenazas y dar respuesta ante incidentes de seguridad. Un claro ejemplo de ello es CIRCL (Computer Incident Response Center Luxembourg), que además de ofrecer formación sobre la herramienta ha desarrollado la librería oficial de Python para interactuar con la API de MISP.

A lo largo de este trabajo se ha observado que la capacidad de integración con todo tipo de productos para intercambiar la información sobre las amenazas es su punto fuerte. Se podría decir que, si alguna herramienta exporta o importa este tipo de información, es posible conectarla con MISP.

Cualquier entorno dedicado al análisis de malware debería contar con un producto que centralice toda la información de las diferentes herramientas para gestionarlas con eficiencia y que además sea capaz de importar más datos de otras organizaciones de confianza.

BIBLIOGRAFÍA

- **Steve Piper.** *Definitive Guid to Next-Generation Threat Protection.* 2013. ISBN 978-0-9888233-1-0
- **Dr. Eric Cole.** *Advanced Persistent Threat.* 2013. ISBN 978-1-59749-949-1
- **Richard Bejtlich.** *The Practice of Network Security Monitoring.* 2013. ISBN 978-1-59327-509-9
- **Michael Hale Ligh, Steven Adair, Blake Hartstein y Matthew Richard.** *Malware Analyst's Cookbook.* 2011. ISBN 978-1-118-00336-7
- **Digit Oktavianto y Iqbal Muhardianto.** *Cuckoo Malware Analysis.* 2013. ISBN 978-1-78216-923-9
- Advanced Alert Analysis – Online Course. Por: *FireEye Training.* 2016.
- **David A. Mundie y David M. McIntire.** *The MAL: A Malware Analysis Lexicon.* 2013. Disponible en: http://www.sei.cmu.edu/reports/13tn010.pdf
- **Manuel Egele, Theodoor Scholte, Engin Kirda y Christopher Kruegel.** *A Survey on Automated Dynamic Malware Analysis Techniques and Tools.* Disponible en: https://seclab.nu/static/publications/acm2012survey.pdf
- *Anatomy of Advanced Persistent Threats.* En: FireEye Resources. Disponible en: https://www.fireeye.com/current-threats/anatomy-of-a-cyber-attack.html
- **Paul Poputa-Clean.** Automated Defense Using Threat Intelligence to Augment Security. En: *SANS Institute - InfoSec Reading Room.* 2015. Disponible en: https://www.sans.org/reading-room/whitepapers/threats/automated-defense-threat-intelligence-augment-35692
- **Greg Farnham.** Tools and Standards for Cyber Threat Intelligence Projects. En: *SANS Institute - InfoSec Reading Room.* 2013. Disponible en: https://www.sans.org/reading-room/whitepapers/warfare/tools-standards-cyber-threat-intelligence-projects-34375
- **Matthew Toussain.** Home-Field Advantage Using Indicators of Compromise to Hunt Down the Advanced Persistent Threat. En: *SANS Institute - InfoSec Reading Room.* 2014. Disponible en: https://www.sans.org/reading-room/whitepapers/detection/home-field-advantage-indicators-compromise-hunt-down-advanced-persistent-threat-35462
- **Hun-Ya Lock.** Using IOC (Indicators of Compromise) in Malware Forensics. En: *SANS Institute - InfoSec Reading Room.* 2013. Disponible en: https://www.sans.org/reading-

room/whitepapers/forensics/ioc-indicators-compromise-malware-forensics-34200

- *Sophisticated Indicators for the Modern Threat Landscape: An Introduction to OpenIOC. Disponible en: http://openioc.org/resources/An_Introduction_to_OpenIOC.pdf*
- **Antonio López.** *United Against Cyberthreats: Information Sharing.* En: CERTSI – Blog. 2016. Disponible en: https://www.certsi.es/en/blog/united-against-cyberthreats-information-sharing
- **José Ortiz.** Deployment of a Flexible Malware Sandbox Environment Using Open Source Software. En: *SANS Institute - InfoSec Reading Room.* 2015. Disponible en: https://www.sans.org/reading-room/whitepapers/incident/deployment-flexible-malware-sandbox-environment-open-source-software-36207
- **Christopher O'Brien.** Automated Network Defense through Threat Intelligence and Knowledge Management. En: *SANS Institute - InfoSec Reading Room.* 2016. Disponible en: https://www.sans.org/reading-room/whitepapers/detection/automated-network-defense-threat-intelligence-knowledge-management-36572
- **Dilshan Keragala.** Detecting Malware and Sandbox Evasion Techniques. En: *SANS Institute - InfoSec Reading Room.* 2016. Disponible en: https://www.sans.org/reading-room/whitepapers/forensics/detecting-malware-sandbox-evasion-techniques-36667
- **George Khalil.** An Overview to Forensic Enterprise Architecture Design. En: *SANS Institute - InfoSec Reading Room.* 2016. Disponible en: https://www.sans.org/reading-room/whitepapers/forensics/ids-file-forensics-35952
- **Yaser Mansour.** An Early Malware Detection, Correlation, and Incident Response System with Case Studies. En: *SANS Institute - InfoSec Reading Room.* 2014. Disponible en: https://www.sans.org/reading-room/whitepapers/detection/early-malware-detection-correlation-incident-response-system-case-studies-34485
- **Stephen Deck.** Extracting Files from Network Packet Captures. En: *SANS Institute - InfoSec Reading Room.* 2015. Disponible en: https://www.sans.org/reading-room/whitepapers/forensics/extracting-files-network-packet-captures-36562
- Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies – RFC2045. En: *IETF – Request For Comments.* 1996. Disponible en: https://tools.ietf.org/html/rfc2045
- Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. En: *IETF – Request For Comments.* 1996. Disponible en: https://tools.ietf.org/html/rfc2046
- **Lauri Palkmets, Cosmin Ciobanu, Yonas Leguesse y Christos Sidiropoulos.** Building artifact handling and analysis environment. En: *European Union Agency for Network and Information Security – Topics.* 2014. Disponible en: https://www.enisa.europa.eu/topics/trainings-for-cybersecurity-specialists/online-training-material/documents/building-artifact-handling-and-analysis-environment-toolset
- **Cynthia Wagner, Alexandre Dulaunoy, Gérard Wagener y Andras Iklody.** *MISP: The Design and Implementation of a Collaborative Threat Intelligence Sharing Platform.* 2016.
- **Deloitte Bélgica, Jo De Muynck y Dr. Silvia Portesi.** Cyber Security Information Sharing: An Overview of Regulatory and Non-regulatory Approaches. En: *European Union Agency For Network And Information Security – Publications.* 2015. Disponible en: https://www.enisa.europa.eu/publications/cybersecurity-information-sharing

ÍNDICE DE CONCEPTOS

GLOSARIO

ANEXO A: DOCKER COMMUNITY EDITION

Las guías y procedimientos de este anexo están probados para Ubuntu en las versiones de Trusty en adelante.

Instalación de Docker Community Edition

Se opta por la instalación desde el repositorio oficial de Docker, ya que facilita la gestión de futuras actualizaciones.

Actualizar el índice de paquetes:

```
$ sudo apt-get update
```

Instalar paquetes adicionales sólo necesarios en Ubuntu Trusty 14.04 y no en versiones posteriores:

```
$ sudo apt-get install linux-image-extra-$(uname -r) linux-image-extra-
virtual
```

Instalar los paquetes que permiten usar repositorios sobre HTTPS:

```
$ sudo apt-get install apt-transport-https ca-certificates curl software-
properties-common
```

Añadir la clave oficial de Docker:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
add –
```

Verificar que el ID de la clave es 9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88:

```
$ sudo apt-key fingerprint 0EBFCD88

[…]

pub   4096R/0EBFCD88 2017-02-22

      Key fingerprint = 9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF
CD88

uid                  Docker Release (CE deb) <docker@docker.com>

sub   4096R/F273FCD8 2017-02-22

[…]
```

Establecer el repositorio de la versión estable de Docker:

```
$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

Actualizar el índice de paquetes con el nuevo repositorio configurado:

```
$ sudo apt-get update
```

Instalar la última versión estable de Docker Community Edition:

```
$ sudo apt-get install docker-ce
```

Verificar que se ha instalado correctamente Docker:

```
$ sudo docker run hello-world
```

Para poder usar Docker como un usuario no administrador se incluye dentro del grupo correspondiente. Es necesario cerrar la sesión y volver a abrirla para que se apliquen los cambios:

```
$ sudo groupadd docker

$ sudo usermod -aG docker $USER
```

Docker se configura automáticamente para ejecutarse al inicio si se usa `upstart`, pero es necesario habilitar esta opción si se usa `systemd`:

```
$ sudo systemctl enable docker
```

Cambiar el controlador de almacenamiento de Docker a Overlay2

Hacer un backup de `/var/lib/docker` teniendo el servicio de Docker detenido:

```
$ cp -au /var/lib/docker /var/lib/docker.bk
```

Modificar el archivo `/etc/docker/daemon.json` o crearlo si no existe añadiendo el siguiente contenido:

```
{
  "storage-driver": "overlay2"
}
```

Comprobar que los cambios se han efectuado de forma correcta después de haber arrancado Docker de nuevo:

```
$ docker info

[…]

Storage Driver: overlay2

[…]
```

ANEXO B: MALICE - MALWARE ANALYSIS FRAMEWORK

Las guías y procedimientos de este anexo están probados para Ubuntu en las versiones de Trusty en adelante.

Instalación de Malice - Malware Analysis Framework

Es necesario tener instalado Docker antes de seguir.

El desarrollo de Malice está aún en fase beta, por lo que se opta por instalar desde un binario precompilado de la versión probada. A la hora de actualizar, se elimina el binario siguiendo el procedimiento indicado en el siguiente apartado para después volver a instalar la versión deseada.

Descargar el binario de la versión 0.3.11:

```
$ wget
https://github.com/maliceio/malice/releases/download/v0.3.11/malice_0.3.1
1_linux_amd64.tar.gz -O /tmp/malice.tar.gz
```

Extraer en la ruta adecuada para su posterior ejecución:

```
$ sudo tar -xzf /tmp/malice.tar.gz -C /usr/local/bin/
```

Desinstalación de Malice - Malware Analysis Framework

Eliminar binario:

```
$ sudo rm /usr/local/bin/malice
```

Eliminar configuraciones y muestras comprimidas de los archivos analizados:

```
$ rm -rf ~/.malice
```

Configuración Inicial Malice - Malware Analysis Framework

Listar todos los plugins indicando de qué tipo es cada uno:

```
$ malice plugin list –-all –-detail
```

Eliminar los plugins que no sean de tipo antivirus o identificador de firmas maliciosas, como por ejemplo los analizadores de metadatos y cadenas de caracteres:

```
$ malice plugin remove [plugin_name]
```

Otra forma de realizar el paso anterior es modificando el archivo de configuración `~/.malice/plugins/plugins.toml` dejando en él solo los plugins que nos interesen. De igual manera, se debe acceder a dicho archivo para habilitar los plugins `[enabled = true]` que por defecto aparecen deshabilitados. En este caso, un ejemplo del mismo es:

```
  [[plugin]]
    enabled = true
    name = "fileinfo"
    description = "ssdeep/TRiD/exiftool"
    category = "metadata"
    image = "malice/fileinfo"
    repository = "https://github.com/malice-plugins/fileinfo.git"
    build = false
    mime = "*"
    env = ["MALICE_TIMEOUT"]

  [[plugin]]
    name = "nsrl"
    enabled = true
    category = "intel"
    description = "NSRL Database Hash Search"
    image = "malice/nsrl:sha1"
    repository = "https://github.com/malice-plugins/nsrl.git"
    build = false
    apikey = ""
    mime = "hash"
    hashtypes = ["sha1"]
    cmd = "lookup"
    env = ["MALICE_TIMEOUT"]
    Installed = false

  [[plugin]]
    name = "virustotal"
    enabled = true
    category = "intel"
```

```
    description = "VirusTotal - files scan and hash lookup"
    image = "malice/virustotal"
    repository = "https://github.com/malice-plugins/virustotal.git"
    build = false
    apikey                                                                  =
"2539516d471d7beb6b28a720d7a25024edc0f7590d345fc747418645002ac47b"
    mime = "hash"
    hashtypes = ["md5", "sha1", "sha256"]
    cmd = "lookup"
    env = ["MALICE_VT_API", "MALICE_TIMEOUT"]
    Installed = false

  [[plugin]]
    name = "totalhash"
    enabled = true
    category = "intel"
    description = "#totalhash - hash lookup"
    image = "malice/totalhash"
    repository = "https://github.com/malice-plugins/totalhash.git"
    build = false
    apikey = ""
    mime = "hash"
    hashtypes = ["sha1"]
    cmd = ""
    env = ["MALICE_TH_USER", "MALICE_TH_KEY", "MALICE_TIMEOUT"]
    Installed = false

  [[plugin]]
    name = "shadow-server"
    enabled = true
    category = "intel"
    description = "ShadowServer - hash lookup"
    image = "malice/shadow-server"
    repository = "https://github.com/malice-plugins/shadow-server.git"
    build = false
    apikey = ""
    mime = "hash"
    hashtypes = ["md5", "sha1"]
    cmd = ""
    env = ["MALICE_TIMEOUT"]
    Installed = false

  [[plugin]]
    name = "team-cymru"
    enabled = true
    category = "intel"
    description = "TeamCymru - hash lookup"
    image = "malice/team-cymru"
    repository = "https://github.com/malice-plugins/team-cymru.git"
    build = false
    apikey = ""
    mime = "hash"
    hashtypes = ["md5", "sha1"]
    cmd = ""
    env = ["MALICE_TIMEOUT"]
    Installed = false

  [[plugin]]
```

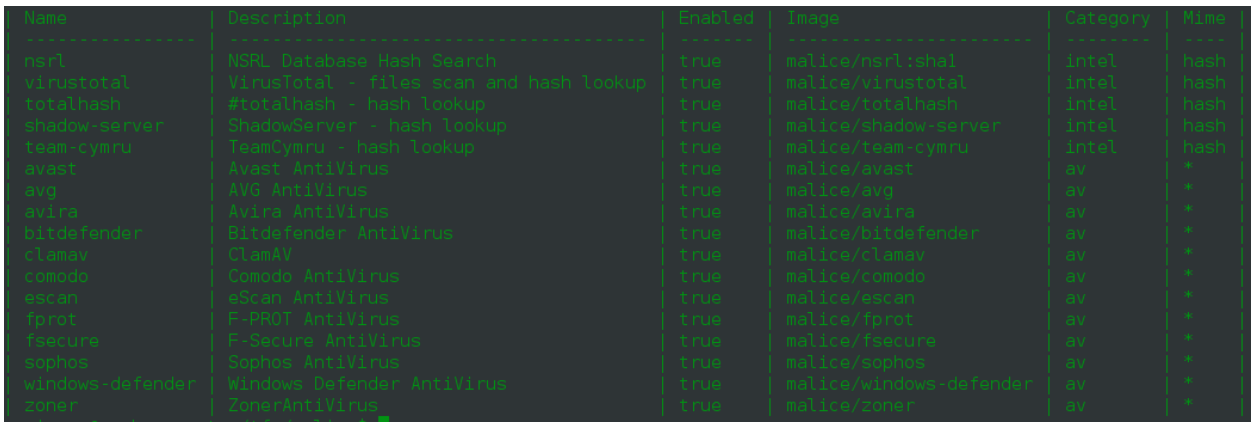
```
    name = "avast"
    enabled = true
    category = "av"
    description = "Avast AntiVirus"
    image = "malice/avast"
    repository = "https://github.com/malice-plugins/avast.git"
    build = false
    apikey = ""
    mime = "*"
    cmd = ""
    env = ["MALICE_TIMEOUT"]
    Installed = false

  [[plugin]]
    name = "avg"
    enabled = true
    category = "av"
    description = "AVG AntiVirus"
    image = "malice/avg"
    repository = "https://github.com/malice-plugins/avg.git"
    build = false
    apikey = ""
    mime = "*"
    cmd = ""
    env = ["MALICE_TIMEOUT"]
    Installed = false

  [[plugin]]
    name = "avira"
    enabled = true
    category = "av"
    description = "Avira AntiVirus"
    image = "malice/avira"
    repository = "https://github.com/malice-plugins/avira.git"
    build = false
    apikey = ""
    mime = "*"
    cmd = ""
    env = ["MALICE_TIMEOUT"]
    Installed = false

  [[plugin]]
    name = "bitdefender"
    enabled = true
    category = "av"
    description = "Bitdefender AntiVirus"
    image = "malice/bitdefender"
    repository = "https://github.com/malice-plugins/bitdefender.git"
    build = false
    apikey = ""
    mime = "*"
    cmd = ""
    env = ["MALICE_TIMEOUT"]
    Installed = false

  [[plugin]]
    name = "clamav"
    enabled = true
```

```
    category = "av"
    description = "ClamAV"
    image = "malice/clamav"
    repository = "https://github.com/malice-plugins/clamav.git"
    build = false
    apikey = ""
    mime = "*"
    cmd = ""
    env = ["MALICE_TIMEOUT"]
    Installed = false

  [[plugin]]
    name = "comodo"
    enabled = true
    category = "av"
    description = "Comodo AntiVirus"
    image = "malice/comodo"
    repository = "https://github.com/malice-plugins/comodo.git"
    build = false
    apikey = ""
    mime = "*"
    cmd = ""
    env = ["MALICE_TIMEOUT"]
    Installed = false

  [[plugin]]
    name = "escan"
    enabled = true
    category = "av"
    description = "eScan AntiVirus"
    image = "malice/escan"
    repository = "https://github.com/malice-plugins/escan.git"
    build = false
    apikey = ""
    mime = "*"
    cmd = ""
    env = ["MALICE_TIMEOUT"]
    Installed = false

  [[plugin]]
    name = "fprot"
    enabled = true
    category = "av"
    description = "F-PROT AntiVirus"
    image = "malice/fprot"
    repository = "https://github.com/malice-plugins/fprot.git"
    build = false
    apikey = ""
    mime = "*"
    cmd = ""
    env = ["MALICE_TIMEOUT"]
    Installed = false

  [[plugin]]
    name = "fsecure"
    enabled = true
    category = "av"
    description = "F-Secure AntiVirus"
```

```
    image = "malice/fsecure"
    repository = "https://github.com/malice-plugins/fsecure.git"
    build = false
    apikey = ""
    mime = "*"
    cmd = ""
    env = ["MALICE_TIMEOUT"]
    Installed = false

  [[plugin]]
    name = "sophos"
    enabled = true
    category = "av"
    description = "Sophos AntiVirus"
    image = "malice/sophos"
    repository = "https://github.com/malice-plugins/sophos.git"
    build = false
    apikey = ""
    mime = "*"
    cmd = ""
    env = ["MALICE_TIMEOUT"]
    Installed = false

  [[plugin]]
    name = "windows-defender"
    enabled = true
    category = "av"
    description = "Windows Defender AntiVirus"
    image = "malice/windows-defender"
    repository      =      "https://github.com/malice-plugins/windows-
defender.git"
    build = false
    apikey = ""
    mime = "*"
    cmd = ""
    env = ["MALICE_TIMEOUT"]
    Installed = false

  [[plugin]]
    name = "zoner"
    enabled = true
    category = "av"
    description = "ZonerAntiVirus"
    image = "malice/zoner"
    repository = "https://github.com/malice-plugins/zoner.git"
    build = false
    apikey = ""
    mime = "*"
    cmd = ""
    env = ["MALICE_TIMEOUT"]
    Installed = false
```

Tras estos cambios, el listado de plugins disponibles debería ser el siguiente:

```
$ malice plugin list –-all –-detail
```



Name	Description	Enabled	Image	Category	Mime
nsrl	NSRL Database Hash Search	true	malice/nsrl:sha1	intel	hash
virustotal	VirusTotal - files scan and hash lookup	true	malice/virustotal	intel	hash
totalhash	#totalhash - hash lookup	true	malice/totalhash	intel	hash
shadow-server	ShadowServer - hash lookup	true	malice/shadow-server	intel	hash
team-cymru	TeamCymru - hash lookup	true	malice/team-cymru	intel	hash
avast	Avast AntiVirus	true	malice/avast	av	*
avg	AVG AntiVirus	true	malice/avg	av	*
avira	Avira AntiVirus	true	malice/avira	av	*
bitdefender	Bitdefender AntiVirus	true	malice/bitdefender	av	*
clamav	ClamAV	true	malice/clamav	av	*
comodo	Comodo AntiVirus	true	malice/comodo	av	*
escan	eScan AntiVirus	true	malice/escan	av	*
fprot	F-PROT AntiVirus	true	malice/fprot	av	*
fsecure	F-Secure AntiVirus	true	malice/fsecure	av	*
sophos	Sophos AntiVirus	true	malice/sophos	av	*
windows-defender	Windows Defender AntiVirus	true	malice/windows-defender	av	*
zoner	ZonerAntiVirus	true	malice/zoner	av	*

Figura B-1. Plugins de Malice instalados

Algunos de los módulos necesitan una licencia para poder analizar como, por ejemplo, Avast o una clave para la API como, por ejemplo, VirusTotal. Si no se van a configurar estos parámetros, se deben eliminar los plugins afectados para que no generen avisos y errores durante la ejecución de Malice.

Actualizar todos los plugins:

```
$ malice plugin update –-all
```

Arrancar la interfaz de usuario web:

```
$ malice elk
```

Si se obtiene un mensaje de error “timeout” al tratar de conectar con Elasticsearch, eliminamos los contenedores que se acaban de crear y configuramos el parámetro asociado a la memoria virtual:

```
INFO[0001] Elasticsearch Container Started                env=development
ip=localhost name="/malice-elastic" port=[9200]

INFO[0001] Waiting for Elasticsearch to come online.
server="http://localhost:9200" timeout=20

ERRO[0021] connecting to elasticsearch timed out          timeout=20

ERRO[0021] Get http://localhost:9200/: dial tcp [::1]:9200: getsockopt:
connection refused

ERRO[0021] Get http://localhost:9200/: dial tcp [::1]:9200: getsockopt:
connection refused

$ docker rm –f malice-elastic malice-kibana
```

Cambiar el parámetro `vm.max_map_count` y volver a intentarlo:

```
$ echo "vm.max_map_count=262144" | sudo tee -a /etc/sysctl.conf

$ sudo sysctl -w vm.max_map_count=262144

$ malice elk
```

Generar datos de prueba para almacenarlos en Elasticsearch ejecutando el análisis de un archivo, por ejemplo, el propio binario de Malice:

```
$ malice scan malice_0.3.11_linux_amd64.tar.gz
```

Acceder a la instancia de Kibana por primera vez yendo a http://localhost, introducir “malice” en “Index name or pattern” y hacer click en “Create”:

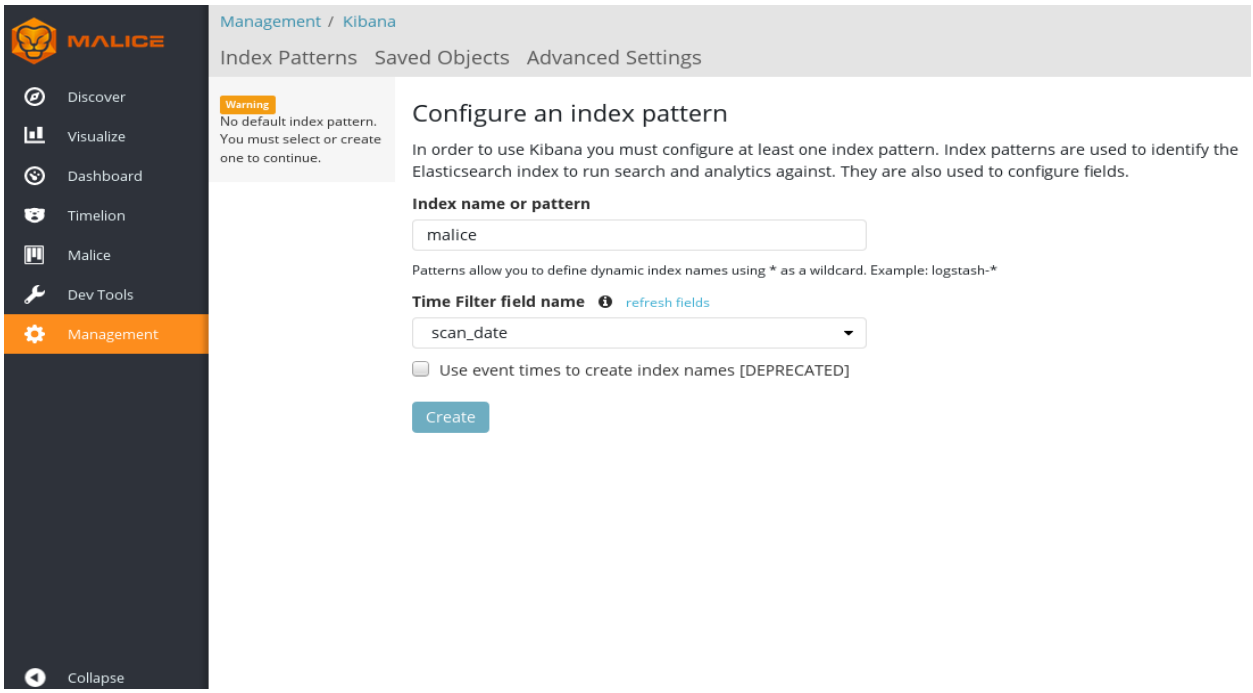


Figura B-2. Configuración de Malice-Kibana primer acceso

Ir a la pestaña “Discover” para comprobar que se encuentra allí el resultado del análisis anterior:

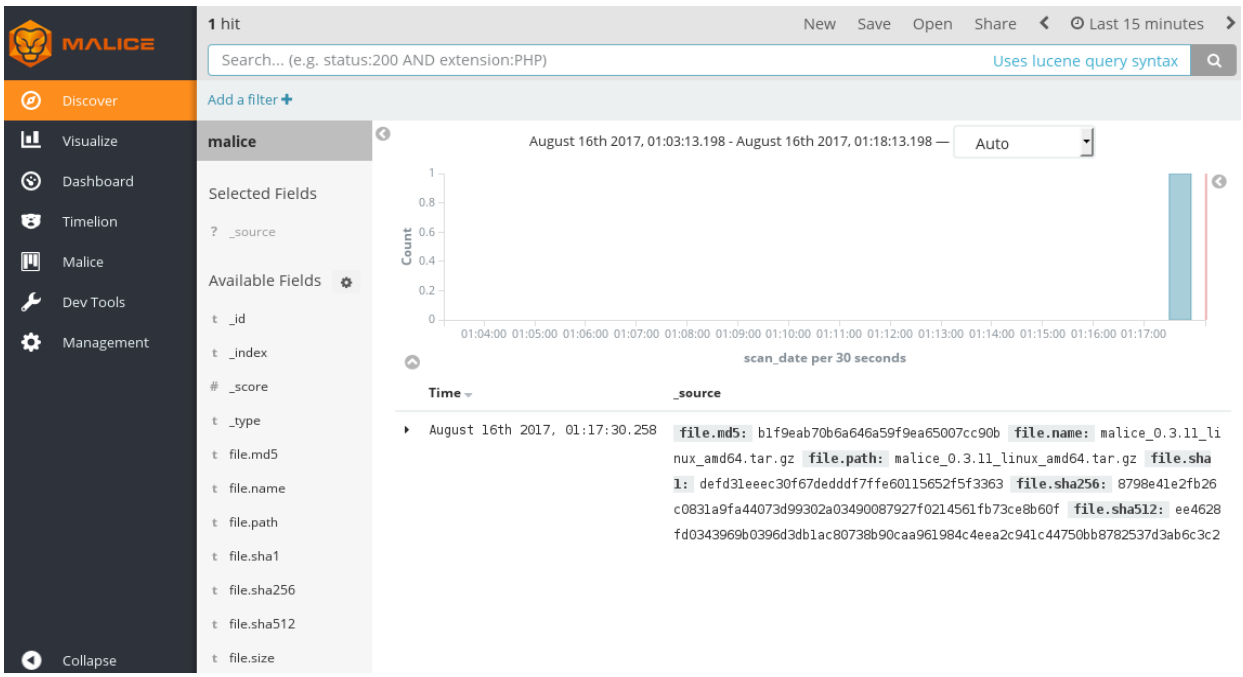


Figura B-3. Ver análisis de prueba en Malice-Kibana primer acceso

ANEXO C: CUCKOO SANDBOX 2.0

Las guías y procedimientos de este anexo están probados para Debian en las version Jessie. Se instalarán las versiones de las dependencias más actualizadas posibles teniendo en cuenta la compatibilidad entre ellas.

Instalación de Cuckoo Sandbox Host

Actualizar el índice de paquetes y el sistema operativo:

```
$ sudo apt-get update && sudo apt-get upgrade && sudo apt-get dist-upgrade
```

Reiniciar el sistema:

```
$ sudo reboot
```

Instalar los paquetes de Python y librerías necesarias para la instalación:

```
$ sudo apt-get install python python-pip python-dev libffi-dev libssl-dev

$ sudo apt-get install python-virtualenv python-setuptools

$ sudo apt-get install libjpeg-dev zlib1g-dev swig
```

Instalar MongoDB para poder utilizar la interfaz web basada en Django:

```
$ sudo apt-get install mongodb
```

Instalar PostgreSQL, base de datos recomendada:

```
$ sudo apt-get install postgresql libpq-dev
```

Instalar Yara 3.4.0:

```
$ sudo apt-get install autoconf libtool libjansson-dev libmagic-dev
libssl-dev

$ wget https://github.com/virustotal/yara/archive/v3.4.0.tar.gz gz -O
yara-3.4.0.tar.gz

$ tar -zxf yara-3.4.0.tar.gz

$ cd yara-3.4.0/

$ ./bootstrap.sh

$ ./configure --with-crypto --enable-cuckoo --enable-magic

$ make

$ sudo make install
```

Validar la instalación de Yara:

```
$ yara –v

yara 3.4.0
```

Instalar la extensión Yara-python:

```
cd yara-python/

$ python setup.py build

$ sudo python setup.py install
```

Validar la instalación de Yara-python:

```
$ pip show yara-python

---

Name: yara-python

Version: 3.4.0

Location: /usr/local/lib/python2.7/dist-packages

Requires:
```

Instalar SSDeep 2.13:

```
$ wget http://sourceforge.net/projects/ssdeep/files/ssdeep-2.13/ssdeep-
2.13.tar.gz/download -O ssdeep-2.13.tar.gz
```

```
$ tar -zxf ssdeep-2.13.tar.gz

$ cd ssdeep-2.13

$ ./configure

$ make

$ sudo make install
```

Validar la instalación de SSDeep:

```
$ ssdeep –V

2.13
```

Instalar PyDeep:

```
$ sudo pip install pydeep
```

Validar la instalación de PyDeep:

```
pip show pydeep

---

Name: pydeep

Version: 0.2

Location: /usr/local/lib/python2.7/dist-packages

Requires:
```

Instalar VirtualBox 5.1:

```
$ echo deb http://download.virtualbox.org/virtualbox/debian $(lsb_release
-cs) contrib | sudo tee -a /etc/apt/sources.list.d/virtualbox.list

$ wget -q https://www.virtualbox.org/download/oracle_vbox_2016.asc -O- |
sudo apt-key add -

$ sudo apt-get update

$ sudo apt-get install virtualbox-5.1
```

Instalar tcpdump:

```
$ sudo apt-get install tcpdump
```

Configurar tcpdump para que se pueda ejecutar sin ser usuario root:

```
$ sudo apt-get install libcap2-bin
```

```
$ sudo setcap cap_net_raw,cap_net_admin=eip /usr/sbin/tcpdump
```

Validar la configuración de tcpdump:

```
$ /sbin/getcap /usr/sbin/tcpdump

/usr/sbin/tcpdump = cap_net_admin,cap_net_raw+eip
```

Instalar Volatility 2.5:

```
$ sudo pip install openpyxl

$ sudo pip install ujson

$ sudo pip install pycrypto

$ sudo pip install distorm3

$ sudo pip install pytz

$ wget
https://github.com/volatilityfoundation/volatility/archive/2.5.tar.gz -O
volatility-2.5.tar.gz

$ tar -xzf volatility-2.5.tar.gz

$ cd volatility-2.5.tar.gz

$ python setup.py build

$ sudo python setup.py install
```

Validar la instalación de Volatility:

```
$ python vol.py –h

Volatility Foundation Volatility Framework 2.5

Usage: Volatility - A memory forensics analysis platform.

[…]
```

Instalar SWIG. Necesaria para que Cuckoo instale automáticamente la dependencia M2Crypto durante su instalación:

```
$ sudo apt-get install swig
```

Crear el usuario para Cuckoo Sandbox:

```
$ sudo adduser cuckoo
```

Agregar el usuario de Cuckoo Sandbox al grupo de VirtualBox:

```
$ sudo usermod -a -G vboxusers cuckoo
```

Instalar Cuckoo Sandbox 2.0.3 usando el usuario `cuckoo`. Se recomienda usar un entorno virtual para evitar problemas con las versiones de las dependencias y no necesitar privilegios de superusuario para actualizar Cuckoo o para añadir nuevos paquetes.

Sin entorno virtual:

```
$ pip install -U pip setuptools

$ pip install -U cuckoo
```

Con entorno virtual:

```
$ virtualenv venv

$ . venv/bin/activate

(venv)$ pip install -U pip setuptools

(venv)$ pip install -U cuckoo
```

Crear la interfaz de tipo “host-only” para las máquinas virtuales de los “guests”

```
$ sudo vboxmanage hostonlyif create

$ sudo vboxmanage hostonlyif ipconfig vboxnet0 --ip 192.168.13.1
```

Crear un “Cuckoo Working Directory” para generar la configuración base de la instancia. Se deberá usar el usuario `cuckoo` para realizar el siguiente paso:

```
$ cuckoo –d
```

Instalación y configuración de Cuckoo Sandbox Guests

En este apartado se describen los procedimientos para crear máquinas virtuales con una configuración estándar. Se recomienda usar un sistema lo más parecido posible al que queremos proteger incluyendo aplicaciones instaladas. Para más información sobre cómo evitar que el malware detecte que está en una máquina virtual, acudir al apartado relacionado con Cuckoo Sandbox en capítulos anteriores.

Creación de Cuckoo Guests manualmente

Es necesario ejecutar los siguientes pasos con el usurio cuckoo.

Crear la máquina virtual:

```
$ vboxmanage createvm --name "win7x64" --ostype Windows7_64 –register

$ cd "VirtualBox VMs/win7x64/"
```

```
$ vboxmanage modifyvm "win7x64" --memory 1000 --acpi on --boot1 dvd --
nic1 nat

$ vboxmanage createhd --filename "win7x64.vdi" --size 256000

$ vboxmanage storagectl "win7x64" --name "IDE Controller" --add ide --
controller PIIX4

$ vboxmanage storageattach "win7x64" --storagectl "IDE Controller" --port
0 --device 0 --type hdd --medium "win7x64.vdi"

$ vboxmanage storageattach "win7x64" --storagectl "IDE Controller" --port
0 --device 1 --type dvddrive --medium
/home/cuckoo/en_windows_7_professional_with_sp1_x64_dvd_u_677056.iso

$ vboxmanage modifyvm "win7x64" --nic1 hostonly

$ vboxmanage modifyvm "win7x64" --hostonlyadapter1 vboxnet0
```

Instalar el sistema operativo:

```
$ vboxmanage startvm "win7x64"
```

Una vez que termine el proceso de instalación mediante la interfaz gráfica, configurar la red:

```
IP: 192.168.13.23

Máscara de red: 255.255.255.0

Puerta de enlace predeterminada: 192.168.13.1

DNS primario: 8.8.8.8

DNS secundario: 8.8.4.4
```

Deshabilitar las actualizaciones automáticas, el firewall de Windows y el UAC (User Account Control).

Instalar aplicaciones como Adobe Reader, Microsoft Office, Firefox, java, etc. Para encontrar versiones anteriores de ciertas aplicaciones: [http://www.oldapps.com/]

Descargar la versión correspondiente de `Python` desde: [https://www.python.org/downloads/release/python-2711/]

```
$ wget https://www.python.org/ftp/python/2.7.11/python-2.7.11.msi
```

Descargar `Python Imaging Library` para Windows:

```
$ wget http://effbot.org/downloads/PIL-1.1.7.win32-py2.7.exe
```

Copiar el agente de Cuckoo a la carpeta donde se hayan descargados los archivos anteriores cambiando su extensión por `.pyw`:

```
$ cp /home/cuckoo/.cuckoo/agent/agent.py /home/cuckoo/Downloads/agent.pyw
```

Pasar los archivos descargados junto con el agente de Cuckoo al Guest:

```
$ python -m SimpleHTTPServer
```

Desde el Guest acceder a http://192.16813.1:8000 y descargar los archivos.

Instalar Python y PIL

Colocar el agente de Cuckoo en la carpeta de arranque al inicio, en Windows 7:

```
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup
```

Reiniciar la máquina virtual.

Ejecutar el agente de Cuckoo, `agent.pyw`, como administrador.

Realizar un snapshot de la máquina virtual con la máquina arrancada y el agente corriendo:

```
$ vboxmanage snapshot "win7x64" take "snapshot1" --pause

$ vboxmanage controlvm "win7x64" poweroff

$ vboxmanage snapshot "win7x64" restorecurrent
```

Creación de Cuckoo Guests mediante VMCloak

Instalar VMCloak 0.4.4:

```
$ wget https://github.com/jbremer/vmcloak/archive/0.4.4.tar.gz -O
vmcloak-0.4.4.tar.gz

$ tar -xzf vmcloak-0.4.4.tar.gz

$ cd vmcloak-0.4.4/

$ sudo pip install -r requirements.txt

$ sudo python setup.py install
```

Crear un punto de montaje para la imagen del SO a utilizar. En este ejemplo será Windows 7 Professional x64:

```
$ sudo mkdir -p /mnt/win7x64

$ sudo mount -o loop,ro
/home/cuckoo/en_windows_7_professional_with_sp1_x64_dvd_u_677056.iso
/mnt/win7x64/
```

Es necesario ejecutar `vmcloak` con el usuario `cuckoo` para evitar posibles problemas de permisos en las máquinas virtuales durante los análisis. Para ello hay que incluir al usuario `cucko` en el archivo `sudoers` antes de seguir con los siguientes pasos.

Crear la máquina virtual con `vmcloak` y finalizar la instalación del sistema operativo. Configurar IP estática y máscara acordes con las creadas anteriormente y comprobar conectividad con el Cuckoo Host (192.168.13.1):

```
$ sudo -E vmcloak init --win7x64 --iso-mount /mnt/win7/ --cpus 1 --
ramsize 1024 --vm-visible -d win7x64
```

Instalar algunas aplicaciones básicas en la máquina virtual:

```
$ sudo -E vmcloak install --vm-visible –d win7x64 adobe9 wic pillow
dotnet40 firefox_41 java7 silverlight5 pil chrome iexplore
```

Hacer un snapshot de la máquina virtual:

```
$ sudo -E vmcloak snapshot --vm-visible –d win7x64 snapshot1
192.168.13.23
```

Configuración inicial Cuckoo Host

Asegurar que todo el contenido del directorio del usuario cuckoo le pertenece:

```
$ sudo chown -R cuckoo:cuckoo /home/cuckoo
```

La configuración de Cuckoo Sanbox es altamente personalizable, por lo que en las siguientes indicaciones se tratan únicamente los parámetros esenciales para su funcionamiento básico.

Editar el archivo `cuckoo.conf`:

```
$ vim /home/cuckoo/.cuckoo/conf/cuckoo.conf
```

```
  […]
  # Specify the name of the machinery module to use, this module will
  #  define  the  interaction  between  Cuckoo  and  your  virtualization
software
  # of choice.
  machinery = virtualbox
  #  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 = yes
  […]
  [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 = 192.168.13.1
```

```
  […]
```

Editar el archivo `auxiliary.conf`:

```
$ vim /home/cuckoo/.cuckoo/conf/auxiliary.conf
```

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

  # Specify the path to your local installation of tcpdump. Make sure
this
  # path is correct.
  tcpdump = /usr/sbin/tcpdump
  […]
```

Editar el archivo `virtualbox.conf`:

```
$ vim /home/cuckoo/.cuckoo/conf/virtualbox.conf
```

```
  […]
  # Default network interface.
  interface = vboxnet0

  # 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 = win7x64

  [win7x64]
  # Specify the label name of the current machine as specified in your
  # VirtualBox configuration.
  label = win7x64

  # 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.
  ip = 192.168.13.23

  # (Optional) Specify the snapshot name to use. If you do not specify
a snapshot
  # name, the VirtualBox MachineManager will use the current snapshot.
  # Example (Snapshot1 is the snapshot name):
  snapshot = snapshot1
```

```
  […]
```

Editar el archivo `processing.conf`:

```
$ vim /home/cuckoo/.cuckoo/conf/processing.conf
```

```
  […]
  [memory]
  # Create a memory dump of the entire Virtual Machine. This memory dump will
  # then be analyzed using Volatility to locate interesting events that can be
  # extracted from memory.
  enabled = yes
  […]
```

Obtener la lista de perfiles disponibles para Volatility:

```
$ vol.py --info |grep Profiles -A48

Volatility Foundation Volatility Framework 2.5

Profiles

--------

VistaSP0x64     - A Profile for Windows Vista SP0 x64

VistaSP0x86     - A Profile for Windows Vista SP0 x86

VistaSP1x64     - A Profile for Windows Vista SP1 x64

VistaSP1x86     - A Profile for Windows Vista SP1 x86

VistaSP2x64     - A Profile for Windows Vista SP2 x64

VistaSP2x86     - A Profile for Windows Vista SP2 x86

Win10x64        - A Profile for Windows 10 x64

Win10x86        - A Profile for Windows 10 x86

Win2003SP0x86   - A Profile for Windows 2003 SP0 x86

Win2003SP1x64   - A Profile for Windows 2003 SP1 x64

Win2003SP1x86   - A Profile for Windows 2003 SP1 x86

Win2003SP2x64   - A Profile for Windows 2003 SP2 x64

Win2003SP2x86   - A Profile for Windows 2003 SP2 x86

Win2008R2SP0x64 - A Profile for Windows 2008 R2 SP0 x64

Win2008R2SP1x64 - A Profile for Windows 2008 R2 SP1 x64
```

```
Win2008SP1x64   - A Profile for Windows 2008 SP1 x64

Win2008SP1x86   - A Profile for Windows 2008 SP1 x86

Win2008SP2x64   - A Profile for Windows 2008 SP2 x64

Win2008SP2x86   - A Profile for Windows 2008 SP2 x86

Win2012R2x64    - A Profile for Windows Server 2012 R2 x64

Win2012x64      - A Profile for Windows Server 2012 x64

Win7SP0x64      - A Profile for Windows 7 SP0 x64

Win7SP0x86      - A Profile for Windows 7 SP0 x86

Win7SP1x64      - A Profile for Windows 7 SP1 x64

Win7SP1x86      - A Profile for Windows 7 SP1 x86

Win81U1x64      - A Profile for Windows 8.1 Update 1 x64

Win81U1x86      - A Profile for Windows 8.1 Update 1 x86

Win8SP0x64      - A Profile for Windows 8 x64

Win8SP0x86      - A Profile for Windows 8 x86

Win8SP1x64      - A Profile for Windows 8.1 x64

Win8SP1x86      - A Profile for Windows 8.1 x86

WinXPSP1x64     - A Profile for Windows XP SP1 x64

WinXPSP2x64     - A Profile for Windows XP SP2 x64

WinXPSP2x86     - A Profile for Windows XP SP2 x86

WinXPSP3x86     - A Profile for Windows XP SP3 x86


Address Spaces

--------------
[…]
```

Editar el archivo `memory.conf`:

```
$ vim /home/cuckoo/.cuckoo/conf/memory.conf
```

```
[…]
# Basic settings
[basic]
# Profile to avoid wasting time identifying it
guest_profile = Win7SP1x64
[…]
```

Editar el archivo `reporting.conf`:

```
$ /home/cuckoo/.cuckoo/conf/reporting.conf
```

```
[…]
# Enable creation of report.html?
html = yes
[…]
[mongodb]
enabled = yes
host = 127.0.0.1
port = 27017
db = cuckoo
store_memdump = yes
paginate = 100
# MongoDB authentication (optional).
username =
password =
[…]
```

Arranque inicial Cuckoo Sandbox

Descargar la versión actualizada de todas las “Cuckoo Signatures”, las reglgas de Yara, etc.:

```
$ cuckoo community
```

Lanzar los siguientes comandos en diferentes shells.

Ejecutar el analizador de Cuckoo Sandbox en modo debug:

```
$ cuckoo –d
```

Ejecutar la interfaz gráfica de Cuckoo Sandbox, aplicación web:

```
$ cuckoo web
```

Ejecutar el servicio REST API de Cuckoo Sandbox:

```
$ cuckoo api
```

Integración con MISP

Cuckoo Sandbox 2.0 hace uso de la REST API de MISP a través de dos módulos de los tipos “processing” y “reporting”.

Editar el archivo `processing.conf`:

```
$ vim /home/cuckoo/.cuckoo/conf/processing.conf
```

```
  […]
  [misp]
  enabled = yes
  url = http://misp.etsi.us
  apikey = your_api_key

  # Maximum amount of IOCs to look up (hard limit).
  maxioc = 100
  […]
```

Editar el archivo `reporting.conf`:

```
$ vim /home/cuckoo/.cuckoo/conf/reporting.conf
```

```
  […]
  [misp]
  enabled = yes
  url = http://misp.etsi.us
  apikey = your_api_key

  # The various modes describe which information should be submitted to
MISP,
  # separated by whitespace. Available modes: maldoc ipaddr hashes url.
  mode = maldoc ipaddr hashes url
  […]
```

La API Key se puede obtener en la interfaz web de MISP después de haber iniciado sesión usando un usuario con permisos de automatización. Dentro del menú “Event Actions” hacer click en “Automation”:

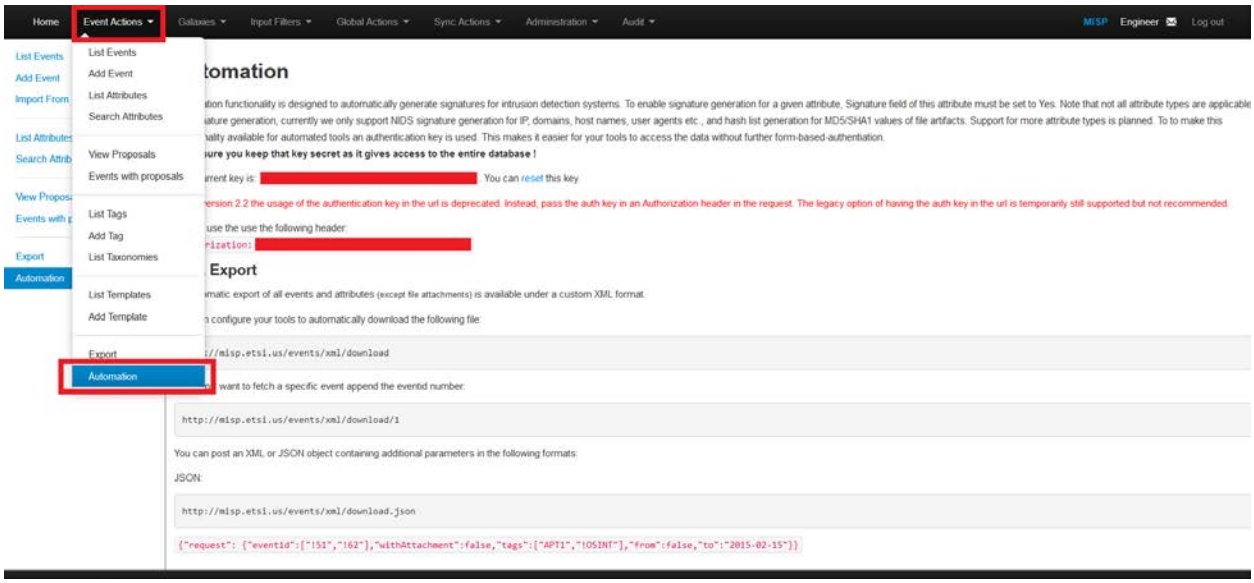


Figura C-1. Obtener API key de MISP

ANEXO D: BRO NETWORK SECURITY MONITOR

Configuración: Extraer archivos del tráfico de red

Editar el archivo `local.bro`:

```
$ sudo vim /opt/bro/share/bro/site/local.bro
```

```
[…]
# File Extraction
@load file-extraction
[…]
```

Crear el archivo `__load.bro__` en la ruta `/opt/bro/share/bro/file-extraction/`:

```
$ sudo vim /opt/bro/share/bro/file-extraction/__load__.bro
```

```
@load ./extract
```

Crear el archivo `extract.bro` en la ruta `/opt/bro/share/bro/file-extraction/`:

```
$ sudo vim /opt/bro/share/bro/file-extraction/extract.bro
```

```
global ext_map: table[string] of string = {
    ["text/plain"] = "txt",
    ["text/html"] = "html",
    ["text/json"] = "json",
    ["image/jpeg"] = "jpg",
```

```
      ["image/png"] = "png",
      ["image/x-ms-bmp"] = "bmp",
      ["image/gif"] = "gif",
      ["application/xml"] = "xml",
      ["application/x-dosexec"] = "exe",
      ["application/x-msi"] = "msi",
      ["application/x-msdownload"] = "dll",
      ["application/pdf"] = "pdf",
      ["application/msword"] = "doc",
      ["application/vnd.ms-excel"] = "xls",
      ["application/vnd.ms-powerpoint"] = "ppt",
      ["application/vnd.openxmlformats-
officedocument.wordprocessingml.document"] = "docx",
      ["application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet"] = "xlsx",
      ["application/vnd.openxmlformats-
officedocument.presentationml.presentation"] = "pptx",
      ["application/java-archive"] = "jar",
      ["application/x-java-applet"] = "jar",
      ["application/x-java-jnlp-file"] = "jnlp",
      ["application/x-shockwave-flash"] = "swf",
      ["application/javascript"] = "js",
      ["application/zip"] = "zip",
      ["application/x-rar-compressed"] = "rar",
      ["application/x-7z-compressed"] = "7z",
      ["application/x-tar"] = "tar",
      ["application/x-gzip"] = "gz",
      ["application/x-bzip"] = "bz",
      ["application/x-bzip2"] = "bz2",
      ["text/mspg-legacyinfo"] = "msi",
      ["application/x-ole-storage"] = "msi",
  } &default ="";

  event file_sniff(f: fa_file, meta: fa_metadata)
      {
      if ( ! meta?$mime_type || meta$mime_type != "application/x-
dosexec" && meta$mime_type != "application/x-msi" && meta$mime_type !=
"application/x-msdownload" && meta$mime_type != "application/pdf" &&
meta$mime_type != "application/msword" && meta$mime_type !=
"application/vnd.ms-excel" && meta$mime_type != "application/vnd.ms-
powerpoint" && meta$mime_type != "application/vnd.openxmlformats-
officedocument.wordprocessingml.document" && meta$mime_type !=
"application/vnd.openxmlformats-officedocument.spreadsheetml.sheet" &&
meta$mime_type != "application/vnd.openxmlformats-
officedocument.presentationml.presentation" && meta$mime_type !=
"application/java-archive" && meta$mime_type != "application/x-java-
applet" && meta$mime_type != "application/x-java-jnlp-file" &&
meta$mime_type != "application/x-shockwave-flash" && meta$mime_type !=
"application/javascript" && meta$mime_type != "application/zip" &&
meta$mime_type != "application/x-rar-compressed" && meta$mime_type !=
"application/x-7z-compressed" && meta$mime_type != "application/x-tar"
&& meta$mime_type != "application/x-gzip" && meta$mime_type !=
"application/x-bzip" && meta$mime_type != "application/x-bzip2" &&
meta$mime_type != "text/mspg-legacyinfo" && meta$mime_type !=
"application/x-ole-storage" && meta$mime_type != "application/octet-
stream" && meta$mime_type != "application/x-msdos-program" )
          return;
```

```
      local ext = "";

      if ( meta?$mime_type )
          ext = ext_map[meta$mime_type];

      local fname = fmt("/opt/bro/extractedFiles/%s-%s.%s", f$source,
f$id, ext);
      Files::add_analyzer(f,                    Files::ANALYZER_EXTRACT,
[$extract_filename=fname]);
      }
```

Cargar la nueva configuración:

```
$ sudo broctl deploy
```

ANEXO E: NINJAMIND

A continuación, se facilita una copia completa del código del programa:

```
 #!/bin/bash

 # ninjaMind monitors the specified directory to perform malware
analyses of all new files.
 # Copyright (C) 2017  Manuel Martín Gutiérrez
 #
 # This program is free software: you can redistribute it and/or
modify
 # it under the terms of the GNU General Public License as published
by
 # the Free Software Foundation, either version 3 of the License, or
 # (at your option) any later version.
 #
 # This program is distributed in the hope that it will be useful,
 # but WITHOUT ANY WARRANTY; without even the implied warranty of
 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 # GNU General Public License for more details.
 #
 # You should have received a copy of the GNU General Public License
 #    along    with    this    program.         If    not,    see
<http://www.gnu.org/licenses/>.

 ######################################################################
 ###                          ninjaMind                             ###
 ###                            v 0.1                               ###
 ###                           09/2017                              ###
 ######################################################################
 ###                   Manuel Martín Gutiérrez                      ###
 ###                                                                ###
 ######################################################################
```

```
  ###################################
  ###     Global Variables      ###
  ###################################

  readonly programName=${0##*/}
  readonly programVersion="0.1"
  quietMode=""


  #######################################
  ###           Functions           ###
  #######################################

  #######################################
  # Print help/usage
  # Globals:
  #   programName
  # Arguments:
  #   None
  # Returns:
  #   None
  #--------------------------------------
  usage() {
                                                                    cat
<<- EOF
                                                                    Usag
e:   $programName   -e   <dir>   [-a   <dir>   [-m   <url[:port]>]]   [-b
<url[:port]>] [-q]

                                                                    Prog
ram monitors the specified directory to perform malware analyses of
                                                                    all
new files.

                                                                    Malw
are analyses can be static (antivirus), dynamic (behavior) or both.
                                                                    At
least one type is required. If static and dynamic are set, behavioral
                                                                    anal
ysis is only available for undetected files by antiviruses.

                                                                    It
is  possible  to  create  an  event  in  the  specified  MISP  (Malware
Information
                                                                    Shar
ing  Platform)  instance  when  one  of  the  antiviruses  detects  an
infection.

                                                                    Opti
ons:
                                                                      -e
<dir>           Extracted files from network traffic directory
                                                                    -a
<dir>           Malice Multi-engine Antivirus results directory
                                                                    -m
<url[:port]>    Malware Information Sharing Platform API URL

Only if -a option is set
```

```
                                                                    -b
<url[:port]>    Cuckoo Sanbodx API URL
                                                                    -q
Quiet mode
                                                                    -v
Show version
                                                                    -h
Show this message

                                                                  Exam
ples:

Run static and dynamic analysis with MISP integration:

$ $programName -e ./files/ -a ./results/ -m http://misp.etsi.us \\

-b http://cuckoo.etsi.us:1301

Run static and dynamic analysis:

$ $programName -e ./files/ -a ./results/ -b http://cuckoo.etsi.us:1301

Run static analysis with MISP integration:

$ $programName -e ./files/ -a ./results/ -m http://misp.etsi.us

Run static analysis:

$ $programName -e ./files/ -a ./reesults/

Run dynamic analysis:

$ $programName -e ./files/ -b http://cuckoo.etsi.us:1301
                                                                  EOF
 }

 ######################################
 # Print ninjaMind logo
 # Globals:
 #   None
 # Arguments:
 #   None
 # Returns:
 #   None
 #--------------------------------------
 logo() {
                                                                  cat
<<- EOF

_        _              _           _

_ __ (_)_ __  (_) __ _  /\/\ (_)_ __   __| |
                                                                    |
'_ \| | '_ \ | |/ _\` |/    \| | '_ \ / _\` |
                                                                    |
| | | | | | || | (_| / /\/\ \ | | | | (_| |

|_| |_|_|_| |_|/ |\__,_\/    \/_|_| |_|\__,_|
```

```
There is |__/ nothing so stable as change

                                                                    EOF
 }

 ######################################
 # Print error messages along with date and time
 # Globals:
 #   quietMode
 # Arguments:
 #   1 Error message to print
 # Returns:
 #   None
 #--------------------------------------
 err() {
   if [[ -z "$quietMode" ]]; then
     echo "[ERR][$(date +'%Y/%m/%d-%H:%M:%S%z')]: $1" >&2
   fi
 }

 ######################################
 # Print information messages along with date and time
 # Globals:
 #   quietMode
 # Arguments:
 #   1 Information message to print
 # Returns:
 #   None
 #--------------------------------------
 inf() {
   if [[ -z "$quietMode" ]]; then
    echo "[INF][$(date +'%Y/%m/%d-%H:%M:%S%z')]: $1"
   fi
 }

 ######################################
 # Create a new task in Cuckoo Sandbox uploading extracted file
 # Globals:
 #   None
 # Arguments:
 #   1 Cuckoo Sandbox API URL
 #   2 Path to extracted file
 # Returns:
 #   0 --> Success
 #   1 --> Error
 #--------------------------------------
 sendFileCuckoo() {
   local fileName=""
   fileName="${2##*/}"

   curl -F file=@${2} "$1"tasks/create/file &>/dev/null

   if [[ $? -ne 0 ]]; then
     err "Cannot send file to Cuckoo Sandbox: $fileName"
     return 1
   else
     inf "Cuckoo Sandbox - File sent: $fileName"
```

```
    fi
  }

  ######################################
  # Create a new event in MISP uploading malware sample
  # Globals:
  #   mispApiKey
  # Arguments:
  #   1 MISP instance API URL
  #   2 Path to extracted file
  #   3 Malice detection details
  # Returns:
  #   0 --> Success
  #   1 --> Error
  #-------------------------------------
  createEventMISP() {
    local fileName=""
    local fileB64=""
    local fileInfo=""
    local fileComment=""

    if [[ -z "$mispApiKey" ]]; then
      err "mispApiKey global variable is unset or empty"
      return 1
    else
      fileName="${2##*/}"
      fileB64="$(base64 $2)"
      fileInfo="$(echo -n "[MALWARE][$(date +'%Y/%m/%d-%H:%M:%S%z')]")"
      fileComment="$3"

      # Data must be read from stdin because argument list is too long
      curl --header "Accept: application/json" --header "Content-type:
application/json" --header "Authorization: $mispApiKey" --request POST
--data @- "$1"events/upload_sample/ &>/dev/null << CURL_DATA
  {"request":{"files": [{"filename": "$fileName", "data": "$fileB64"}],
"distribution":   0,   "category":"Payload   delivery",   "info"   :
"$fileInfo", "comment":"$fileComment"}}
  CURL_DATA

      if [[ $? -ne 0 ]]; then
        err "Cannot create MISP Event: $fileName"
        return 1
      else
        inf "MISP - Event created: $fileName"
      fi
    fi
  }

  ######################################
  # Parse Malice result file
  # Globals:
  #   None
  # Arguments:
  #   1 Path to Malice result file
  # Returns:
  #   0 --> Success
  #   1 --> Error
  #-------------------------------------
```

```
  parseMaliceResults() {
    local fileName=""
    fileName="${1##*/}"
    fileName="${fileName%.mr}"

    grep -B 3 "| true |" $1 2>/dev/null

    if [[ $? -ge 2 ]]; then
      err "Cannot parse Malice result: $fileName"
      return 1
    else
      return 0
    fi
  }

  ######################################
  # Check and clean up antivirus Docker containers
  # Globals:
  #   None
  # Arguments:
  #   None
  # Returns:
  #   0 --> Success
  #   1 --> Error
  #-------------------------------------
  checkAVContainers() {
    local avContList=""
    local avCont=""
    local avContStatus=""

    # Get names of all antivirus containers
    avContList="$(malice plugin list 2>/dev/null)"

    if [[ $? -ne 0 ]]; then
      err "Cannot list Malice plugins"
      return 1
    else
      # If a container exists, force the removal
      for avCont in $avContList; do
        avContStatus="$(docker  inspect  -f  {{.State.Status}}  $avCont
2>/dev/null)"
        if [[ -n "$avContStatus" ]]; then
          docker rm -f $avCont &>/dev/null
        fi
      done
    fi
  }

  ######################################
  # Check and start Malice ELK Docker containers
  # Globals:
  #   None
  # Arguments:
  #   None
  # Returns:
  #   0 --> Success
  #   1 --> Error
  #-------------------------------------
```

```
 checkMaliceELKContainers() {
   local elaStatus
   local kibStatus

   # Get containers information
   elaStatus="$(docker  inspect  -f  {{.State.Status}}  malice-elastic
2>/dev/null)"
   kibStatus="$(docker  inspect  -f  {{.State.Status}}  malice-kibana
2>/dev/null)"

   #Check containers information
   # If both containers are running, do not start Malice ELK
   if [[ "$elaStatus" != "running" || "$kibStatus" != "running" ]];
then
     if [[ "$elaStatus" != "running" ]]; then
       if [[ -n "$elaStatus" ]]; then
         # If the container exists, but it is not running: remove it
         docker rm malice-elastic &>/dev/null
       fi
     else
       # If only this container is running, stop it and remove it
       docker stop malice-elastic &>/dev/null
       docker rm malice-elastic &>/dev/null
     fi

     if [[ "$kibStatus" != "running" ]]; then
       if [[ -n "$kibStatus" ]]; then
         # If the container exists, but it is not running: remove it
         docker rm malice-kibana &>/dev/null
       fi
     else
       # If only this container is running, stop it and remove it
       docker stop malice-kibana &>/dev/null
       docker rm malice-kibana &>/dev/null
     fi

     if ! malice elk &>/dev/null; then
       err "Cannot start Malice ELK Docker containers"
       return 1
     fi
   fi
 }

 ######################################
 # Monitor spcified directory and send each new file to Cuckoo Sandbox
 # Globals:
 #   None
 # Arguments:
 #   1 Extracted files directory to monitor
 #   2 Cuckoo Sandbox API URL
 # Returns:
 #   0 --> Success
 #   1 --> Error
 #----------------------------------- #
 extrFilesMonCuckoo() {
   local extrFile=""
```

```
    ### NOTE: Follow "###" comments to manage exit properly inside
intofywait loops
    ### Use "trap" to capture "PIPE" BEFORE inotifywait loops
    trap ':' PIPE

    # Monitor specified directory
    inotifywait -m -r -q -e close_write -e moved_to --format "%w%f"
"$1" |
                while read extrFile; do
      ### Check errors: kill PIPE and then exit INSIDE inotifywait
loops
      # Send the file to Cuckoo Sandbox
      if ! sendFileCuckoo "$2" "$extrFile"; then
        kill -PIPE 0
        exit 1
      fi
    done

    ### Use "trap" to capture "PIPE" AFTER inotifywait loops
    trap PIPE
  }

  ######################################
  # Monitor specified directory and scan each new file with Malice
  # Globals:
  #   None
  # Arguments:
  #   1 Extracted files directory to monitor
  #   2 Directory to save malice results
  # Returns:
  #   0 --> Success
  #   1 --> Error
  #----------------------------------- #
  extrFilesMonMalice() {
    local extrFile=""
    local extrFileName=""
    local extrFileDir=""
    local malResDir=""

    ### NOTE: Follow "###" comments to manage exit properly inside
intofywait loops
    ### Use "trap" to capture "PIPE" BEFORE inotifywait loops
    trap ':' PIPE

    # Monitor specified directory
    inotifywait -m -r -q -e close_write -e moved_to --format "%w%f"
"$1" |
                while read extrFile; do
      extrFileName="${extrFile##*/}"
      extrFileDir="${extrFile%$extrFileName}"

      # Check if extracted file is in a subdirectory
      if [[ "$1" != "$extrFileDir" ]]; then
        # Create the same subtree in Malice results directory
        # Use the same subtree to save result file
        malResDir="$2${extrFileDir##$1}"
        mkdir -p "$malResDir" &>/dev/null
      else
```

```
        malResDir="$2"
      fi

      ### Check errors: kill PIPE and then exit INSIDE inotifywait
loops
      # Check Docker daemon
      if ! docker info &>/dev/null; then
        err "Unable to communicate with docker daemon"
        kill -PIPE 0
        exit 1
      # Check and start Malice ELK Docker containers
      elif ! checkMaliceELKContainers; then
        kill -PIPE 0
        exit 1
      # Check and clean up antivirus Docker containers
      elif ! checkAVContainers; then
        kill -PIPE 0
        exit 1
      # Scan the file with Malice
      elif ! malice scan "$extrFile" >"${malResDir}${extrFileName}.mr"
2>/dev/null; then
        err "Cannot scan files with Malice"
        kill -PIPE 0
        exit 1
      fi
    done

    ### Use "trap" to capture "PIPE" AFTER inotifywait loops
    trap PIPE
  }

  ######################################
  # Monitor specified directory and parse each new Malice result
  # Globals:
  #   None
  # Arguments:
  #   1 Malice results directory to monitor
  #   2 Extracted files directory
  #   3 Cuckoo Sandbox API URL
  #   4 MISP API URL
  # Returns:
  #   0 --> Success
  #   1 --> Error
  #-------------------------------------
  maliceResultsMon() {
    local malResult=""
    local malResultName=""
    local malResultDir=""
    local extrFileName=""
    local extrFileDir=""

    ### NOTE: Follow "###" comments to manage exit properly inside
intofywait loops
    ### Use "trap" to capture "PIPE" BEFORE inotifywait loops
    trap ':' PIPE

    # Monitor specified directory
```

```
    inotifywait -m -r -q -e close_write --format "%w%f" "$1" | while
read malResult; do
      ### Check errors: kill PIPE and exit INSIDE inotifywait loops
      malResultName="${malResult##*/}"
      malResultDir="${malResult%$malResultName}"
      extrFileName="${malResultName%.mr}"

      # Check if result file is in a subdirectory
      if [[ "$1" != "$malResultDir" ]]; then
        # Use the same subtree to locate extracted file
        extrFileDir="$2${malResultDir##$1}"
      else
        extrFileDir="$2"
      fi

      # Parse Malice Result
      malInfections="$(parseMaliceResults "$malResult")"
      if [[ $? -ne 0 ]]; then
        kill -PIPE 0
        exit 1
      elif [[ -n "$malInfections" ]]; then
        # Infection detected
        inf "AV - Infection detected: $extrFileName"
        if [[ -n "$4" ]]; then
          # Create an event if MISP option is set
          createEventMISP     "$4"      "${extrFileDir}${extrFileName}"
"$malInfections"
        fi
      elif [[ -n "$3" ]]; then
        # Infection undetected
        # Send file if Cuckoo Sandbox option is set
        sendFileCuckoo "$3" "${extrFileDir}${extrFileName}"
      fi
    done

    ### Use "trap" to capture "PIPE" AFTER inotifywait loops
    trap PIPE
  }

  ######################################
  #  Manage  exit  cleaning  up  Docker  containers  and  killing  all
subprocesses
  # Globals:
  #   None
  # Arguments:
  #   1 Malice results directory
  #   2 PIDs of the subprocesses
  # Returns:
  #   0 --> Success
  #   1 --> Error
  #-------------------------------------
  manageExit() {
    if [[ -n "$1" ]]; then
      checkAVContainers &>/dev/null
    fi

    kill "$2" &>/dev/null
    echo
```

```
  exit
}

#######################################
###             Main                ###
#######################################

main() {
  # Arguments variables
  local extFilDir=""
  local avResDir=""
  local mispURL=""
  local cuckooURL=""
  local hArg=""
  local vArg=""

  # List of subrocesses
  local nmPIDs=""

  # Check number of arguments
  if  [[ $# -eq 0 ]]; then
    logo
    usage
    exit 1
  fi

  # Extract options and their arguments into variables
  OPTIND=1
  while getopts ":e:a:m:b:qvh" opt; do
    case "$opt" in
      e)
        [[ "$OPTARG" != */ ]] && OPTARG="$OPTARG/"
        extFilDir="$OPTARG"
        ;;
      a)
        [[ "$OPTARG" != */ ]] && OPTARG="$OPTARG/"
        avResDir="$OPTARG"
        ;;
      m)
        [[ "$OPTARG" != */ ]] && OPTARG="$OPTARG/"
        mispURL="$OPTARG"
        ;;
      b)
        [[ "$OPTARG" != */ ]] && OPTARG="$OPTARG/"
        cuckooURL="$OPTARG"
        ;;
      q)
        quietMode="-q"
        ;;
      v)
        vArg="-v"
        ;;
      h)
        hArg="-h"
        break
        ;;
      :)
        err "Option -$OPTARG requires an argument"
```

```
                                                                    exit
1
          ;;
        \?)
          err "Invalid option: -$OPTARG"
          exit 1
          ;;
      esac
    done

    # Remove options that have already been handled from $@
    shift $((OPTIND -1))

    # Mark argument variables unchangeable
    readonly extFilDir
    readonly avResDir
    readonly mispURL
    readonly cuckooURL
    readonly quietMode
    readonly hArg

    # Check arguments
    [[ -z "$quietMode" ]] && logo

    if [[ -n "$hArg" ]]; then
      [[ -n "$quietMode" ]] && logo
      usage
    elif [[ -n "$vArg" ]]; then
      echo "Version: $programVersion"
    elif [[ -z "$extFilDir" ]]; then
      err "-e option is required"
      exit 1
    elif [[ ! -d "$extFilDir" ]]; then
      err "$extFilDir is not a directory"
      exit 1
    elif [[ ! (-r "$extFilDir" && -w "$extFilDir" && -x "$extFilDir")
]]; then
      err  "Unable  to  use  $extFilDir  -  User  $USER  must  have  rwx
permissions"
      exit 1
    elif [[ -z "$avResDir" ]]; then
      if [[ -z "$cuckooURL" ]]; then
        # There is not specified analyzer
        err "At  least  one  type  of  analysis  is  required:  -a  and/or  -b
options"
        exit 1
      else
        # File Analyzer: Cuckoo Sandbox
        inf "Extracted files location: $extFilDir"
        inf "Cuckoo Sandbox Host: $cuckooURL"
        inf "Running behavioral analysis..."
        extrFilesMonCuckoo "$extFilDir" "$cuckooURL" &
        nmPIDs="$nmPIDs $!"
      fi
    elif [[ ! -d "$avResDir" ]]; then
      err "$avResDir is not a directory"
      exit 1
```

```
    elif [[ ! (-r "$avResDir" && -w "$avResDir" && -x "$avResDir") ]];
then
      err "Unable to use $avResDir - User $USER must have rwx
permissions"
      exit 1
    elif [[ -z "$cuckooURL" ]]; then
      # File Analyzer:  Malice Multi-Antivirus
      inf "Extracted files location: $extFilDir"
      inf "Malice results location: $avResDir"
      [[ -n "$mispURL" ]] && inf "MISP instance: $mispURL"
      inf "Running multi-engine antivirus analysis..."

      extrFilesMonMalice "$extFilDir" "$avResDir" &
      nmPIDs="$nmPIDs $!"
      maliceResultsMon "$avResDir" "$extFilDir" "$cuckooURL" "$mispURL"
&
      nmPIDs="$nmPIDs $!"
    else
      # File Analyzers:  Malice Multi-Antivirus and Cuckoo Sandbox
      inf "Extracted files location: $extFilDir"
      inf "Malice results location: $avResDir"
      [[ -n "$mispURL" ]] && inf "MISP instance: $mispURL"
      inf "Cuckoo Sandbox Host: $cuckooURL"
      inf "Running multi-engine antivirus and behavioral analysis..."

      extrFilesMonMalice "$extFilDir" "$avResDir" &
      nmPIDs="$nmPIDs $!"
      maliceResultsMon "$avResDir" "$extFilDir" "$cuckooURL" "$mispURL"
&
      nmPIDs="$nmPIDs $!"
    fi

    # Control exit properly
    trap 'manageExit "$avResDir" "$nmPIDs"' SIGHUP SIGINT SIGQUIT
SIGTERM

    # Wait for all currently active child processes
    wait -n
  }

  main "$@"
```

ANEXO F: PRESUPUESTO

(hidden track)

Get a good job with more pay and you're okay.
Money, it's a gas.

Roger Waters, Pink Floyd

A continuación se detallan los costes económicos derivados de la realización de este trabajo. Se dividen en dos partidas: recursos humanos, un Ingeniero de Telecomunicaciones Junior y recursos materiales, el equipo informático necesario para realizar las tareas descritas en este documento. Se considera coste toda aquella inversión necesaria si se llevase a cabo fuera del ámbito académico.

Tabla F-1. Presupuesto del proyecto

Concepto	Cantidad	Precio/Unidad	Importe
Ingeniero de Telecomunicaciones Junior			**7800 €**
Investigación	100 horas	26 €/hora	2600 €
Diseño	50 horas	26 €/hora	1300 €
Implementación	100 horas	26 €/hora	2600 €
Experimentación	20 horas	26 €/hora	520 €
Documentación	30 horas	26 €/hora	780 €
Equipo de Trabajo			**2190 €**
Portátil (i7-7700HQ, 16 GB RAM)	1 Ud.	1965 €/Ud.	1965 €
Monitor 24”	1 Ud.	225 €/Ud.	225 €
TOTAL			**9990 €**

Don’t be another brick in the wall.

Manuel Martín González, 2011