Instituto Politécnico Nacional
Centro de Investigación en Computación

Diseño de un "Laboratorio de Malware para uso Académico y de Investigación" ALMA *(Advanced Laboratory for Malware Analysis)* Requerimientos, funcionalidades, etc.

Asignatura

Análisis y Detección de Malware

Versión 1.1

13 de noviembre del 2022

1 Control de versiones del documento

Fecha	Ver.	Comentarios	Autor (es)
05/nov./2022	1.0	Versión inicial (estructura del documento).	Raúl Acosta Bermejo (RAB)
09/nov./2022		Captura de contenido y nuevas secciones.	RAB
13/nov./2022	1.1	Captura de imágenes, tablas y nuevas ideas de contenido.	RAB
		Versión final	

2 Tabla de Contenido

3 Tabla de Figuras

4 Tabla de Tablas

5 Resumen

El presente documento tiene como finalidad, explicar de manera general cómo funciona una Laboratorio de Malware para poder construir uno desde cero con un uso académico y de investigación.

6 Introducción

En esta sección se describen los conceptos generales que se deben conocer para comprender el trabajo que se quiere realizar y el cual se describe en la última sección de objetivos.

6.1 Malware y Analistas de malware

Malware o "software malicioso" es un término amplio que describe cualquier programa o código malicioso que es dañino para los sistemas. El malware es hostil, intrusivo e intencionadamente desagradable ya que intenta invadir, dañar o deshabilitar ordenadores, sistemas informáticos, redes, tabletas y dispositivos móviles, a menudo asumiendo el control parcial de las operaciones de un dispositivo.

Los malware más comunes son:

1. Un virus es malware que se adjunta a otro programa y, cuando se ejecuta —normalmente sin que lo advierta el usuario—, se replica modificando otros programas del ordenador e infectándolos con sus propios bits de código.
2. Los gusanos son un tipo de malware similar a los virus, que se replica por sí solo con el fin de diseminarse por otros ordenadores en una red, normalmente provocando daños y destruyendo datos y archivos.
3. Un troyano, o caballo de Troya, es uno de los tipos de malware más peligrosos. Normalmente se presenta como algo útil para engañar al usuario. Una vez que está en el sistema, los atacantes que se ocultan tras el troyano obtienen acceso no autorizado al ordenador infectado. Desde allí, los troyanos se pueden utilizar para robar información financiera o instalar amenazas como virus y ransomware.
4. El ransomware es un tipo de malware que bloquea el acceso del usuario al dispositivo o cifra sus archivos y después lo fuerza a pagar un rescate para devolvérselos. El ransomware se ha reconocido como el arma preferida de los delincuentes informáticos porque exige un pago rápido y provechoso en criptomoneda de difícil seguimiento. El código que subyace en el ransomware es fácil de obtener a través de mercados ilegales en línea y defenderse contra él es muy difícil.
5. El rootkit es un tipo de malware que proporciona al atacante privilegios de administrador en el sistema infectado. Normalmente, también se diseña de modo que permanezca oculto del usuario, de otro software del sistema y del propio sistema operativo.
6. Un keylogger o registrador de pulsaciones de teclas, es malware que graba todas las pulsaciones de teclas del usuario, almacena la información recopilada y se la envía al atacante, que busca

información confidencial, como nombres de usuario, contraseñas o detalles de la tarjeta de crédito.

7. El adware es un software no deseado diseñado para mostrar anuncios en su pantalla, normalmente en un explorador. Suele recurrir a un método subrepticio: bien se hace pasar por legítimo, o bien se adosa a otro programa para engañar al usuario e instalarse en su PC, tableta o dispositivo móvil.
8. El spyware es malware que observa las actividades del usuario en el ordenador en secreto y sin permiso, y se las comunica al autor del software.
9. Los exploits son un tipo de malware que aprovecha los errores y vulnerabilidades de un sistema para que el creador del exploit pueda asumir el control. Los exploits están vinculados, entre otras amenazas, a la publicidad maliciosa, que ataca a través de un sitio legítimo que descarga contenido malicioso inadvertidamente desde un sitio peligroso. A continuación, el contenido dañino intenta instalarse en el ordenador tras una descarga involuntaria.

Analista de Malware

Es alguien que evalúa las amenazas de ciberseguridad para un empleador. Estos profesionales a menudo pasan gran parte de su tiempo aprendiendo sobre los tipos más comunes de virus, malware y otras tecnologías dañinas, así como sobre cómo minimizar su impacto negativo en una red.

En general, el analista de malware proporciona experiencia en software dañino, a menudo llamado malware, y recursos relacionados utilizados por piratas informáticos o cualquier otra persona que intente dañar un sitio web o red. Esto implica monitorear cuidadosamente la seguridad existente y encontrar las mejores herramientas nuevas para mejorar la capacidad de la red para manejar las amenazas.

Con frecuencia, los analistas de malware examinan bots, gusanos, troyanos y otros tipos de malware para descubrir cómo funcionan. Estos profesionales crean respuestas que mantendrán la red segura. Estos pueden incluir la detección preventiva con detectores de antivirus o malware, así como el mantenimiento de firewalls o los esfuerzos humanos para aislar y destruir el malware.

El papel de un analista de malware incluye realizar ingeniería inversa. La ingeniería inversa es el proceso de buscar un producto de software compilado y descubrir cómo está hecho. Los analistas de malware pueden hacer esto con malware para generar las respuestas más efectivas.

Por otro lado, es importante ubicar el rol del Analista de Malware en un entorno de trabajo real. Uno de los entornos más frecuentes es el de trabajar de forma coordinada con un SOC (*Security Operations Center*), es decir, un equipo de especialistas de ciberseguridad encargados de monitorear una red para prevenir o detectar ataques. Este entorno de trabajo (véase la **Figura 1**) puede llegar a ser bastante complejo dependiendo de la cantidad de usuarios y la diversidad de activos con los que cuente.

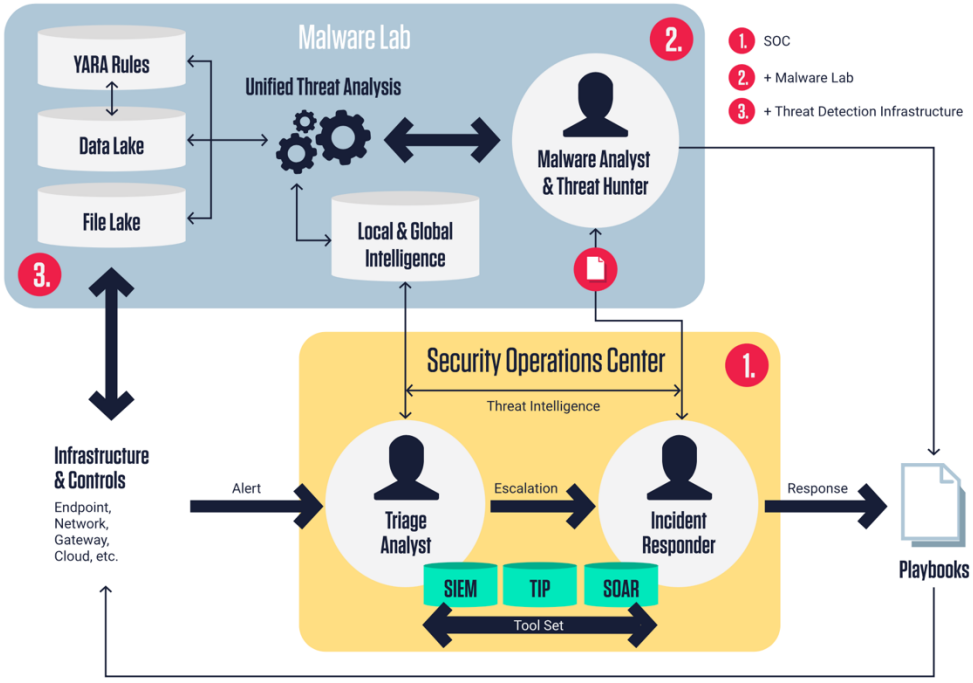

Figura 1. Entorno de trabajo del Analista de Malware. **Fuente**. Falta referencia.

Hay muchos sitios con cursos e información para que los analistas de malware se entrenen, por ejemplo:

1. Malware Traffic Analysis
    a. https://www.malware-traffic-analysis.net/index.html

Existen varias certificaciones posibles para un analista de malware, algunas de las mejores son:

1. SANS
   Ofrece una lista muy amplia de certificaciones en ciberseguridad que si bien son caras, son las mejores a nivel mundial.
    a. https://www.sans.org/cyber-security-courses/reverse-engineering-malware-malware-analysis-tools-techniques/
    b. https://www.giac.org/certifications/reverse-engineering-malware-grem/
2. Academias de Hackeo Ético.
    a. https://ethicalhackersacademy.com/products/certified-malware-analyst
    b. https://www.isacindia.org/isac-certified-reverse-engineer-and-malware-analyst/

6.2 Indicadores

6.2.1 IoA

Los Indicadores de ataque (IoA – *Indicators of Attack*, por sus siglas en inglés) se refieren al conjunto de evidencias que ofrecen pistas de una actividad **sospechosa** y en el mejor de los escenarios adelantarse a un eventual ataque en curso.

6.2.2 IoC

Un Indicador de compromiso (IoC – *Indicator Of Compromise*) es un conjunto de datos sobre un objeto o una actividad que indica acceso no autorizado al equipo (compromiso de datos). Es una evidencia en una computadora que indica que la seguridad de la red **ha sido violada**.

Dado que actualmente la detección de malware se hace con base en comportamientos y no firmas, las empresas crean bases de datos de IoC:

1. Kaspersky
    a. https://support.kaspersky.com/KESWin/11.7.0/es-MX/213408.htm.
2. INQuest Labs
    a. https://labs.inquest.net/

Tanto los IoA como los IoC se clasifican:

1. Tráfico de datos inusual.
2. Cambios en la actividad de los usuarios privilegiados.
3. Fallas de autenticación (varios intentos).
4. Actividad inusual en las bases de datos.
5. Solicitudes de acceso a archivos importantes (varios intentos).
6. Cambios de configuración sospechosos.
3. Señales de actividad DDoS.
4. Tráfico web con conductas sospechosas.

La medición de la mayoría de estos indicadores no es exacta y es por eso que se usan algoritmos de *Machine Learning*.

6.2.3 Vulnerabilidades

Las vulnerabilidades son las **brechas de seguridad presentes en cualquier software**. Si se explotan, estas fallas pueden permitir a los atacantes obtener acceso no autorizado a información confidencial o, en general, causar problemas que ponen en riesgo al dueño del activo.

Una vulnerabilidad es un fallo en un programa o sistema informático. Pero no cualquiera, sino un fallo de seguridad. Es necesaria esta distinción puesto que no todos los errores de programación derivan en fallos de seguridad. Un error en un programa puede llevar a que no funcione correctamente o que su comportamiento no sea el esperado, pero no todos estos tipos de problemas pueden considerarse fallos de seguridad. Según la capacidad de aprovecharse de este defecto, la vulnerabilidad será más o menos grave.

Las vulnerabilidades son difíciles de gestionar. Se descubren decenas día a día, y clasificarlas es una tarea compleja. Para ello, la organización MITRE (www.mitre.org) creó el sistema CVE el cual es imprescindible hoy en día para estandarizar las vulnerabilidades y para poder referenciarlas y conocer su gravedad de forma objetiva. Además, este permite responder, siempre que sea posible, a todos los parámetros que definen la vulnerabilidad.

El CVE (*Common Vulnerabilities and Exposures*) es un estándar que se encarga de identificar unívocamente a las vulnerabilidades mediante un identificador único (DNI) cuyo formato es una cadena formada por las letras CVE, seguido del año en el que se asignó el código a la vulnerabilidad, seguido de un número de cuatro cifras, por ejemplo, CVE-2011-1234.

La detección de ciertas vulnerabilidades se puede hacer mediante la búsqueda de patrones (de cadenas o secuencias de bytes). La herramienta más usada es YARA y las empresas crean sus propias reglas de búsqueda (no públicas) como parte de su servicio de detección de malware, sin embargo, existen varios sitios con reglas YARA gratuitas que pueden ser utilizadas.

1. ReversingLabs
    a. https://github.com/reversinglabs/reversinglabs-yara-rules

En un entorno real de producción es sumamente importante tener la lista de reglas YARA actualizadas y/o crear nuevas.

6.3 Laboratorio de Malware

El objetivo general de un Laboratorio de Malware es identificar el comportamiento de un malware y esto puede hacerse de dos formas: i) teniendo acceso directo a la muestra de malware, o ii) obteniendo de forma indirecta el comportamiento, por ejemplo, obteniendo y analizando las comunicaciones realizadas o diversas métricas (por ejemplo, de rendimiento). En ambos casos el análisis puede ser de dos tipos explicados a continuación.

6.3.1 Análisis estático

El análisis estático de malwares es un conjunto de técnicas que permiten estudiar, prever y observar el funcionamiento de este tipo de softwares **sin necesidad de ejecutarlos**. El análisis se hace por medio de la revisión del código fuente del archivo y la identificación de elementos maliciosos en el mismo. De este modo, un analista puede hacerse una idea de las tareas que ejecuta el malware en un sistema **sin correr el riesgo de infectar el ordenador** con este.

6.3.2 Análisis dinámico

Los análisis dinámicos son realizados en máquinas virtuales preparadas para ejecutar malwares dañinos y demostrar su funcionamiento en tiempo real. Se diferencian del análisis estático de malwares en que en los análisis dinámicos **se ejecuta el software malicioso** y **se le permite que ejecute tareas dañinas en un entorno virtual controlado**.

Sin embargo, aquí no termina el proceso de análisis. Después de esta fase, sigue la ingeniería inversa del virus. La ingeniería inversa es un proceso por medio del cual los analistas de malware ejecutan el software con su código fuente ensamblado, para poder ver el paso a paso de cada una de sus tareas. Este método combina las dos fases anteriores y ofrece un análisis detallado acerca del funcionamiento del virus.

La ingeniería inversa es diferente al análisis estático de malwares en cuanto a que se ejecuta el programa a la vez que se estudia su código fuente. Es decir, se observan las miles de peticiones que hace el software de manera detenida y, por lo tanto, son análisis que pueden tomar varias semanas o meses. La principal desventaja de esta fase es su duración, que podría ser más larga de lo necesario para reparar las acciones de un virus.

6.3.3 Trabajos similares

Algunos desarrollos previos de Laboratorios de Malware o servicios de análisis de malware son:

1. Comerciales
   Con versión gratuita limitada
   a. VirusTotal
   b. Hybrid Analysis
      i. Empresa: Crowdstrike.
   c. ANY.RUN
2. Gratuitos
   a. Cuckoo
   b. MultiScanner

Algunos trabajos similares de Máquinas Virtuales para análisis de malware son:

1. FLARE VM
2. REMnux.

Un análisis comparativo de estos trabajos se presenta en la **Tabla 1**.

Sistema	Licencia	Tipos de Archivos	Análisis Estático	Análisis Dinámico	Machine Learning	??
Cuckoo	OS				X	
MultiScanner	OS					
VirusTotal	$	EXE, URLs			✓	
Hybrid Analysis	$					
ANY.RUN	$					
	IPN	Sólo EXE				

Tabla 1. Comparativa de trabajos similares.

6.4 Objetivos y alcance

Objetivo general

Construir un Laboratorio de Malware que realice el análisis automático de cierto tipo de muestras de malware y genere una serie de documentos que contengan los resultados del análisis. Los análisis que se podrán realizar serán de varios tipos y dependerán del tipo de la muestra de malware y serán configurables. El sistema tiene como propósito la formación de estudiantes como analistas de malware por lo que es importante el manejo adecuado de las evidencias y el procesamiento de múltiples muestras de malware.

Al sistema se le denominará ALMA (***A**dvanced **L**aboratory for **M**alware **A**nalysis*).

Objetivos específicos

OE-1. Realizar la construcción del laboratorio de manera incremental tanto en su funcionalidad como en su rendimiento.
   a. Para la cantidad de muestra empezar por cientos, luego miles y al final millones.
   b. Para el tipo de archivo se empezará con ejecutables de un solo procesador y sistema operativo.

OE-2. Dado el gran número de opciones posibles de análisis, el diseño debe contemplar la posibilidad de "agregar funcionalidades" nuevas.
   a. Para el caso de máquinas virtuales se empezará con VirtualBox y luego QEMU.
   b. Para la programación utilizar primero comandos y luego librerías, por ejemplo, para el control de las máquinas virtuales.

OE-3. Gestionar de forma adecuada la diversidad de opciones de entrada y salida
   a. Se usarán archivos de configuración, por ejemplo, archivos Json para las máquinas virtuales.
   b. Se usarán archivos de resultados, por ejemplo, archivos de texto y/o Json para la información resultante de los análisis.

OE-4. Interacción con otros sistemas de análisis
   a. Se podrá complementar el análisis de las muestras con los resultados de otros sistemas de antivirus, por ejemplo, el VirusTotal.

OE-5. Orientar la generación de resultados a la detección de malware mediante técnicas de Máquinas de Aprendizaje.
   a. Generará los N-grams de ciertos resultados como los Opcode, Syscall o API calls.

Alcance

1. Solo se analizarán muestras de malware de ejecutables en formato ELF para Linux.
2. Sólo se utilizará un tipo hipervisor: VirtualBox de tipo 1 (requiere un sistema operativo).

3. Sólo se realizarán experimentos con **50** muestras de malware.
4. Sólo se realizarán hasta **5** tipos de análisis estáticos diferentes y **2** de análisis dinámicos a las muestras de malware.
5. El sistema NO realizará la Detección de malware y no se incluye el desarrollo de algoritmos para ello. Sin embargo, si generará los datos que serían los insumos de un algoritmo de detección (por ejemplo, el uso de *Machine Learning*).
6. El sistema NO implementará un esquema de Control de Acceso, es decir, de creación de usuarios, grupos, roles y sus permisos.

7 Especificación de requerimientos

En esta sección se describe el funcionamiento del Laboratorio de Malware que se quiere construir utilizando diferentes elementos típicos del Desarrollo de Software. Los elementos son presentados siguiendo una secuencia que inicia con los requerimientos más abstractos y terminando con la especificación más concreta y detallada posible.

7.1 Requerimientos

Si bien el sistema no utilizará un esquema de roles, la totalidad de las funciones del sistema pueden dividirse en dos grupos:

1. Las de administración del sistema (ADM).
2. Las de un usuario que lleva a cabo el análisis de las muestras de malware (MLW).

Esta clasificación se incluye en la información de las siguientes secciones cuando aplica.

Hay varios formas de clasificar los requerimientos pero en las siguientes secciones se utilizarán sólo dos tipos: funcionales y no funcionales.

7.1.1 Funcionales

Los requerimientos funcionales de un sistema, son aquellos que describen cualquier actividad que este deba realizar, en otras palabras, el comportamiento o función particular de un sistema o software cuando se cumplen ciertas condiciones. Por lo general, estos deben incluir funciones desempeñadas por pantallas específicas, descripciones de los flujos de trabajo a ser realizadas por el sistema y otros requerimientos de negocio.

No	Identificador	Nombre	Descripción
1	RF-01	Hipervisores ADM	Se podrán usar varios tipos y marca. Por ejemplo, los hipervisores de tipo 1 (requiere un SO), de tipo 2 (conocidos como *baremetal* y no

			requieren un SO), o emuladores (de procesadores como Qemu).
			Poder utilizar más de una marca.
	RF-01.0		Lista de operaciones de MV.
	RF-01.1		VirtualBox
	RF-01.2		QEMU
	RF-01.3		VMware
2	RF-02	Análisis Estático MLW	Analiza una muestra de malware sin ejecutarla.
	RF-02.0		Generar los datos de identificación del malware: valores hash y su metadata (tipo de archivo).
	RF-02.1		Determinar si el archivo usa alguna técnica de ofuscación: comprimido, antidebugging.
	RF-02.2		Obtener las cadenas.
	RF-02.3		Obtener las librerías que usa y su tipo.
	RF-02.4		Obtener la lista de OPCODEs.
	RF-02.5		Realizar el cálculo de varias métricas del código binario.
	RF-02.6		Buscar vulnerabilidades usando Patrones/YARA
3	RF-03	Análisis Dinámico MLW	Analiza una muestra de malware ejecutándola en un ambiente controlado.
	RF-03.0		Ejecutar archivos de 64, 32, y 16 bits. Determinar requerimientos del kernel y su ver.
	RF-03.1		Generar los OPCODEs.
	RF-03.2		Generar los Syscall.
	RF-03.3		Generar los API calls.
	RF-03.4		Generar el volcado de memoria del proceso(s)
	RF-03.5		Generar la lista de archivos utilizados.
	RF-03.6		Generar la lista de conexiones de red usadas.
	RF-03.7		Capturar los paquetes de red transmitidos.
	RF-03.8		Buscar vulnerabilidades usando Patrones/YARA
	RF-03.9-X		Generar los Ngrams del X (RF-03.1).
4	RF-04	Grafos MLW	Construir un grafo a partir de la información obtenida en los requerimientos previos.
	RF-04.1		Construir el FCG (*Flow Call Graph*).
	RF-04.2		Construir el CFG (*Call Flow Graph*).
	RF-04.3		Construir el: Data Flow Graph Finite State Machine. Cadena de Markov
5	RF-05	Configuración ADM	El sistema debe poder funcionar en diferentes escenarios que pueden ser elegidos o cambiados dinámicamente.
	RF-05.1		Validar la estructura de los archivos Json y/o cargar la información en la BD.
	RF-05.2		Cambiar la dirección IP por default de las MV.

6	RF-06	GUI ADM y MLW	Se tendrá una interfaz gráfica de usuario que permite visualizar toda la información de forma jerárquica y clasificada.
	RF-06.1 RF-06.2		Se usarán pestañas para mostrar la información. Se usarán tablas de 2 columnas para mostrar las duplas (llave, valor).
	FUNCIONES	**AVANZADAS**	
7	RF-07	Patrones estáticos MLW	Se analizarán las muestras de malware para buscar patrones estáticos.
	RF-07.1 RF-07.2		Se usará la herramienta YARA. Se integrarán un repositorio de archivos de reglas públicas.
8	RF-08	API Rest MLW	Se podrá interactuar con el sistema de forma remota usando un API de tipo Rest.
	RF-08.1 RF-08.2 RF-08.3		Se usará un API key para validar las operaciones. APIs para subir muestras de malware. APIs para descargar los resultados.
9	RF-09	Sistemas externos MLW	Integración de funcionalidades de sistemas externos.
	RF-09.1		VirusTotal
10	RF-10	Interacción GUI MLW	Se simula la interacción de una persona con el sistema.
	RF-10.1		Movimiento del mouse, uso del teclado.
11	RF-11	Formatos de Archivos MLW	Se podrán analizar otros tipos de archivos.
	RF-11.1 RF-11.2 RF-11.3		Imágenes: overlay, steganography. Documentos de office. Documentos de lectura: PDF.

Tabla 2. Lista de Requerimientos funcionales.

Para poder saber el comportamiento específico de varios de los requerimientos, es necesario una descripción más amplia que puede requerir desde uno o varios párrafos, hasta un documento independiente lo cual está fuera del alcance de este documento . Por ejemplo, para el RF-03.6 la lista puede contener:

1. La dirección destino.
2. El protocolo y puestos utilizados.
3. La estadística de la cantidad de información transmitida.

7.1.2 No funcionales

Los requerimientos no funcionales representan cualidades, características, y restricciones de la aplicación o sistema que se esté desarrollando. Los requerimientos funcionales suelen denominarse atributos de

calidad de un sistema y algunos ejemplos de ellos son la seguridad, usabilidad y otros medibles en tiempo de ejecución como la escalabilidad.

Suelen presentar dificultades en su definición dado que su conformidad o no conformidad podría ser sujeto de libre interpretación, por lo cual es recomendable acompañar su definición con criterios de aceptación que se puedan medir.

No	Identificador	Nombre	Descripción
1	RNF-01	Seguridad	Establecer las políticas necesarias para evitar ataques y/o la propagación de las muestras de malware fuera del sistema.
	RNF-01.1 RNF-01.2		Definir controles de acceso tanto locales como remotos. .
2	RNF-02	Rendimiento	Para tener un rendimiento adecuado se deberá usar tecnologías que permitan explotar los recursos de la infraestructura. La medición del tiempo de respuesta de los análisis estáticos no debe rebasar un minuto.
	RNF-02.1 RNF-02.2		Crear procesos/hilos. Utilizar un esquema de gestión de tareas (posiblemente basado en colas).
3	RNF-03	Control de Errores	
4	RNF-04	Portabilidad	
5	RNF-05	Escalabilidad	
6	RNF-06	Disponibilidad	
7	RNF-07	Mantenibilidad	
8	RNF-08	Interfaces	Externas (con otros sistemas), de Usuario, etc.

Tabla 3. Lista de Requerimientos No funcionales.

7.2 Procesos de Negocio

No	Identificador	Nombre	Descripción
1	PN-01	Gestión de Máquinas virtuales	Librería de MV. Se encarga del CRUD de MV.
2	PN-02	Gestión de experimentos	Cantidad de tareas y su avance para cada experimento.
3	PN-03	Control y monitoreo del Análisis estático	

4	PN-04	Control y monitoreo del Análisis dinámico	
5	PN-05	Gestión de Resultados	

Tabla 4. Lista de Procesos de Negocio.

7.3 Reglas de Negocio

Toda organización trabaja de acuerdo con un conjunto de políticas, leyes y estándares. Parte de la reglamentación es externa y está documentada formalmente. La normativa interna puede estar formalizada o puede consistir en un conjunto de convenciones. Las **reglas del negocio** están conformadas tanto por la reglamentación externa e interna, las normas formales y las convenciones.
Las **reglas del negocio** describen las políticas, normas, operaciones, definiciones y restricciones presentes en una organización y que son de vital importancia para alcanzar los objetivos misionales.

No	Identificador	Nombre	Descripción
1	RN-01		
2	RN-02		
3	RN-03		
4	RN-04		
5	RN-05		

Tabla 5. Lista de Reglas de Negocio.

7.4 Casos de Uso

Un Caso de Uso (CU) está formado por los siguientes elementos:

1. Actores: principal y secundarios.
2. La descripción de las actividades
    a. Identificando el flujo principal (normal) y los sub-flujos (alternos o excepciones).
    b. Los flujos paralelos.
    c. Los procedimientos
3. Entradas y salidas: cualquier tipo de artefacto (ej. archivos).

4. Precondiciones y postcondiciones.
5. Diagrama de CU
6. Reglas de Negocio

No	Identificador	Nombre	Descripción
1	CU-01	Máquinas virtuales	Gestiona la librería de MV, es decir realiza el CRUD de MV.
	CU-01.1		Dar de alta una nueva MV.
	CU-01.2		Dar de baja una MV.
2	CU-02		
3	CU-03		
4	CU-04		
5	CU-05		

Tabla 6. Lista de Reglas de Negocio.

7.5 Matrices de Trazabilidad

Estas matrices permiten verificar que se cumple con los objetivos y esto se realiza obteniendo las relaciones que deben existir de la especificación del sistema en un alto nivel de abstracción (objetivos) hasta la especificación de bajo nivel (casos de uso).

REQ\ Objetivos Específicos	OE-1	OE-2	OE-03
RF-01	✓		
RNF-01		✓	

Tabla 7. Requerimientos vs Objetivos específicos.

REQ\ CU	CU-01	CU-02	CU-03	CU-04	CU-05	CU-06	CU-07	CU-08	CU-09	CU-10
RF-01	✓									

RNF-01	✓

Tabla 8. Requerimientos vs Casos de Uso.

8 Diseño del sistema ALMA

8.1 Modelo de operación

El modelo de operación de laboratorio se basa en el concepto de experimento el cual define:

1. Cuantas y cuales muestras de malware se utilizarán.
2. Que tipos de análisis estáticos y dinámicos se realizarán.
3. Los diferentes elementos de configuración que permitan definir con precisión los análisis y el ambiente de ejecución.

Entrada	Caja Negra	Salida
Muestra(s) de malware Archivos de configuración	Laboratorio de Malware	Archivos de resultados

Figura 2. Modelo de operación del sistema.

La finalidad del análisis es:

1. Poder identificar la familia de malware de la muestra de malware.
2. Realizar una descripción técnica (lo más detallada posible) del comportamiento del malware. Para esta descripción se puede utilizar:
   a. La matriz de ATT&CK de MITRE.
   b. Asociar los IoA con las TTP (Tácticas, Técnicas y Procedimiento) de la matriz.
   c. Identificar las vulnerabilidades y/o su clasificación según el CVE.

8.2 Arquitectura simplificada

8.2.1 Componentes tecnologías de software

1. Hipervisor

   a. VirtualBox, versión x.y.z
2. Base de datos
   a. MySql, versión x.y.z
3. Framework
   a. Django, versión x.y.z
   b. Programación en Python del módulo central del sistema usando el modelo MVC.
4. Librerías
   a. Libvirt, versión x.y.z
   b. Gestión de las MV usando un API.

8.2.2 Módulos del sistema

El sistema está conformado por:

1. Componentes. Elementos de software que ya existen y solo se instalan y configuran.
2. Módulos. Elementos que son creados (programados) para tener la funcionalidad deseada.
   a. Los módulos serán creados con el lenguaje y el framework elegido.

El sistema está compuesto por **5** módulos descritos a continuación.

1. Módulo central
   a. Es el encargado de coordinar todas las actividades del sistema en sus diferentes capas, módulos y componentes.
2. Módulo de gestión de máquinas virtuales.
   a. Es el encargado de poder administrar la librería de máquinas virtuales.
   b. Las operaciones que debe poder hacer con las máquinas virtuales son: encender, apagar, clonar, suspender, restaurar un snapshot.
   c. Este módulo está dividido en 2 partes: la administración y la ejecución del hipervisor.
3. Módulo de gestión de tareas.
4. Módulo de análisis de malware
   a. Este módulo es capaz de crear tareas específicas que realizan el análisis de una muestra de malware.
   b. Este módulo debe ser configurable para poder tener un catálogo de análisis editable, es decir, agregar, eliminar, o editar elementos de análisis.
   c. Algunos análisis requieren varios elementos para funcionar, por ejemplo, la detección de vulnerabilidades mediante patrones usando YARA. Esto requiere un repositorio de reglas.
5. Módulo GUI o de Resultados

IMAGEN

Figura 3. Arquitectura del software.

MV-3 Análisis dinámico
MVoCON-2 Análisis estático
MVoCON-1. Sistema central
Capa básica: SO + Hipervisor + Módulo

IMAGEN

Figura 4. Arquitectura del sistema.

8.2.3 Algoritmos principales

Algoritmo de Análisis de muestras

```
1  Crear_experimento (lista de muestras)
2  PARA cada muestra de malware
3     Determinar el tipo de muestra de malware.
4     SI es posible realizar el análisis (tipo, almacenamiento, etc.)
5        Preparar la infraestructura (si es necesario) / Error: decidir si
6  continuar o no
7                    Crear o clonar MV, encenderlas y restaurarlas
8     Crear lista de tareas del análisis estático
9     Crear lista de tareas del análisis dinámico
         # Algunas requieren tareas previas / dependencias.
         # Ejemplo: Para los Ngrams se requiere que termine antes la tarea de,
         # por ejemplo, obtener opcodes.
10 Pedir confirmación de inicio al cliente
      # Esto informa que si se puede y que no.
11 Ejecutar las tareas y esperar a que terminen: secuencial o paralelo
      # Esto implica gestionar también los errores.
12 Compilar resultados y/o mostrarlos
```

Figura 5. Algoritmo de análisis de muestras.

Algoritmo de Monitoreo de proceso muestra

La idea es tener algo similar a la aplicación de Windows conocida como “Process Monitor” pero con ciertos cambios, por ejemplo, sin interfaz gráfica y sin la necesidad de la interacción de un usuario.

Para sistemas tipo UNIX (caso de Linux), el problema principal a resolver es que hay muchos comandos para monitoreo, pero no son capaces de hacerlo para un proceso, lo cual obliga a capturar todo

(ineficiente) y luego esperar a que use un recurso (dispositivo de red) para filtrar la información (esto puede que no capture el comportamiento inicial).

```
1  Crear proceso genérico suspendido (ruta del ejecutable)
2  Ejecutar proceso genérico y capturar el PID
3  Preparar estructuras de datos
4  Ejecutar procesos de monitoreo con el PID
      Archivos utilizados (comando de Linux lsof)
      Syscalls utilizados (comando de Linux strace)
      Paquetes de red transmitidos (crear una red de monitoreo - namespace)
5  Ejecutar proceso suspendido de la muestra malware
```

Figura 6. Algoritmo de monitoreo de proceso muestra.

8.2.4 Máquinas Virtuales para el análisis / Ambiente controlado

Estas máquinas virtuales son usadas para realizar el análisis estático y dinámico de las muestras, es por eso que son construidas bajo el concepto de Ambiente Controlado. Un ambiente controlado es aquel que monitorea y limita varias actividades en el sistema operativo.

En particular estas MV se caracterizan por:

1. Usar agentes que monitorean toda la actividad del malware.
2. Controlan que actividades puede realizar el malware.
    a. Estos controles pueden ser internos o externos, por ejemplo, un firewall. Por default no debe establecer comunicación externa.
    b. No deben permitir cambios a la configuración.

Para el caso de las máquinas virtuales que harán el análisis del malware se contempla:

1. Monitorea el proceso que ejecuta el malware y además:
    a. Monitorea los procesos e hilos que cree y ejecute.
    b. Monitorea los diferentes mecanismos de comunicación entre procesos (IPC) que use de acuerdo con el sistema operativo.
2. Monitorear las comunicaciones de red.
   Debe poder identificar:
    a. Las direcciones fuente y destino. Pueden ser tanto de IP como de HTTP.
    b. Los protocolos y puertos utilizados.
    c. L captura de los paquetes transmitidos.
3. Monitorear la interacción con el sistema de archivos.
   Debe poder identificar:
    a. Los archivos creados, eliminados, o modificados.

b. El cambio de permisos de archivos.

8.3 Diagramas UML

8.3.1 Diagrama de componentes

IMAGEN

Figura 7. Diagrama de componentes.

8.3.2 Diagramas de secuencias

IMAGEN

Figura 8. Diagrama de secuencia.

8.3.3 Diagramas de la Base de Datos

Diagrama ER o Relacional

IMAGEN

Figura 9. Diagrama de la BD.

Diccionario de datos

Tabla: Nombre

No	tipo de dato	nombre del campo	propiedades	Descripción
1	unsigned int	id	Llave principal	
2				
3				

Tabla 9. Diccionario de la tabla Nombre.

8.3.4 Diagramas de la GUI

En esta sección se presentan los *wireframes* (Marcos de alambre) de la Interfaz Gráfica de Usuario.

Las principales vistas del sistema son:

Identificador	Nombre	Descripción
VIS-1	Inicio del sistema.	
VIS-2	Librería de MV.	

Tabla 10. Lista de vistas del sistema.

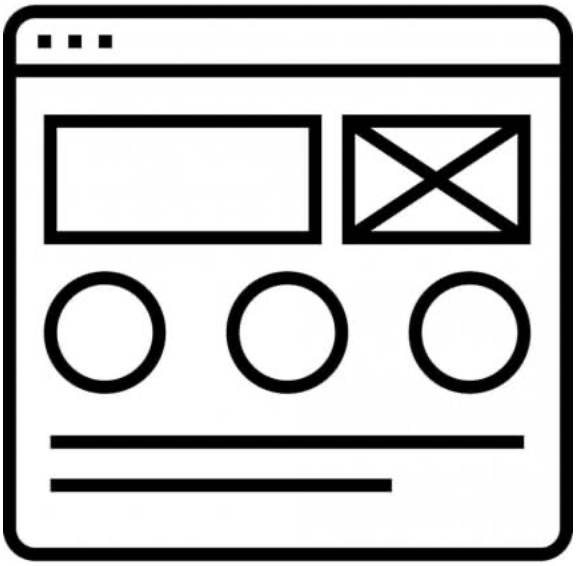

Figura 10. Vista de inicio del sistema. VIS-1.

9 Plan de Pruebas

9.1 Matriz de pruebas

CU\ CP	CP-01	CP-02	CP-03	CP-04	CP-05	CP-06	CP-07	CP-08	CP-09	CP-10
CU-01	✓									
CU-02										
CU-03										
CU-04		✓								
CU-05										
CU-06										

CU-07	
CU-08	
CU-09	
CU-11	

Tabla 11. Casos de Uso vs Casos de Prueba.

9.2 Casos de prueba

9.2.1 Funcionales

El.

No	Identificador	Nombre	Descripción
1	CP-01	Hipervisores	Verificar el uso de los diferentes tipos de hipervisores. Este caso de prueba corrsponde con RF-01.
	CP-01.0		Probar la interfaz de las operaciones.
	CP-01.1		VirtualBox
	CP-01.2		QEMU
	CP-01.3		VMware
2	CP-02	Análisis estático	
	CP-02.1		Uso de un archivo ejecutables creados.
	CP-02.2		Uso de muestras de malware.
3	CP-03	Análisis dinámico	
	CP-03.1		Uso de un archivo ejecutables creados.
	CP-03.2		Uso de muestras de malware.

Tabla 12. Casos de Prueba funcionales.

9.2.2 No Funcionales

El.

No	Identificador	Nombre	Descripción
1	CP-01		
	CP-01.0		
	CP-01.1		
	CP-01.2		
	CP-01.3		
2			
3			

4	
5	

Tabla 13. Casos de Prueba No funcionales.

10 Conclusiones

11 Glosario de términos

Concepto	Descripción
CRUD	Acrónimo usado en bases de datos que representan las operaciones que se pueden realizar: Create, Read, Update, Delete La operación de lectura puede dividirse en dos, la de leer un registro completo o la de listar un resumen de los registros.

12 Bibliografía

URLs
Laboratorios
https://www.virustotal.com
https://www.hybrid-analysis.com/
https://any.run/
https://cuckoosandbox.org/
https://github.com/mitre/multiscanner

Maquinas
https://www.mandiant.com/resources/blog/flare-vm-update

13 Anexos

13.1 Estructura de archivos

El laboratorio de malware utilizará una estructura de la información tal que le permita evolucionar a diferentes herramientas tecnológicas sin afectar el funcionamiento de un conjunto de funcionalidades que forman parte del núcleo del sistema. La estructura propuesta de la información es la siguiente.

No	1er Nivel	2do Nivel	3er Nivel	Descripción
1	config			Contiene la información de la configuración del sistema.
2		vm		Contiene los archivos Json de las máquinas virtuales.
2		keys		Contiene los archivos de las llaves para una conexión remota sin password a las máquinas virtuales.
3	vm			Contiene toda la información de las máquinas virtuales.
4		vm_#		Cada carpeta contiene los archivos de una máquina virtual. Las MV tienen un identificador único.
5	exp	exp_#		Cada carpeta contiene los archivos de un experimento.
			config	Configuración del experimento
			results	Archivos de los resultados del experimento. Cada archivo tendrá un nombre único de acuerdo a un catálogo y podrán ser de texto sin formato (txt) o estructurados (json): 1. strings.json 2. headers.json
6	repo	yara		Repositorio de reglas YARA.

Tabla 14. Estructura de archivos del sistema.

13.2 Archivos Json

En esta sección se presenta la estructura propuesta de los archivos Json a usar. La estructura se apega a la definición estándar que contempla:

1. Tipos básicos: cadenas, enteros, booleanos.
2. Objetos. secuencia de duplas (separadas por comas) cada una formada por un primer campo denominado llave y un segundo llamado valor (separados por dos puntos ":").
3. Arreglos de los elementos anteriores.

13.2.1 Configuraciones diversas

Base de datos

```
{
  nombre: “alma_config”,
  user: “root”,
  password: “12345678”
  puerto: 3306
}
```

Cuentas de sistemas

```
[
  { sistema: “VirusTotal”,
    user: “root”,
    password: “12345678” },
  { Los campos se repiten }
]
```

13.2.2 Máquinas virtuales

```
{
  ubicacion: /ruta/subdir_mv,
  hipervisor: virtualbox|qemu|...,
  sistema_operativo: “Windows 10”
  nombre: “Win-10-1”
  usuarios: [
    { user: “root”, password: “12345678”},
    { user: “alma”, password: “@Soul2023”}
  ],
  direcciones_red: [ “dir1”, “dir2”, “dir3”]
  remote: { puerto:2220, protocolo:ssh }
  snapshot: “name1”
}
```

13.2.3 Análisis estático

Por cada análisis estático que se realice habrá un archivo Json que especifica que cosas se realizarán y en algunos casos como.

```
{
  hashes: 0|1,
  string: 0|1,
```

```
  headers: 0|1,
  libraries: 0|1,
  asm: 0 | [{objdump, radare2}],
  opcodes: 0|1
  ngrams: {opcode:[1,2,3] }
}
```

13.2.4 Análisis dinámico

Por cada análisis dinámico que se realice habrá un archivo Json que especifica que cosas se realizarán y en algunos casos como.

```
{
  time: unsigned int        tiempo dado en minutos
  opcodes: 0|1,
  syscall: 0|1,
  api_calls: 0|1,
  dump: 0|1,

  files: 0|1,
  conections: 0|1,
  pcap: 0|1,
  os: {
       windows_registry: 0|1,
  }
  ngrams: {opcode:[1,2,3], syscall:[4,5], apicall:[6]}
}
```

13.3 Lista de herramientas

Con la intención de dar una idea de las posibles herramientas a utilizar en las máquinas virtuales que realizan el análisis del malware, a continuación, se presenta una lista resumida.

No	Tipo	Windows	Linux	MacOS
1	Syscall	StraceNT Dr Strace		
2	APIcall			
2				
3	Opcodes		objdump radare2	
4				
5	Volcado de Memoria	Dr. Memory		

Tabla 15. Estructura de archivos del sistema.

Para una lista más actualizada y completa, véase el archivo de Excel anexo denominado ALMA-herramientas.

========== ESTA ES LA ÚLTIMA PÁGINA ==========