UNIVERSIDAD INTERNACIONAL SEK

DIGITAL SCHOOL

Trabajo de fin de carrera titulado

"ANÁLISIS COMPARATIVO DE DIVERSAS METODOLOGÍAS DE DETECCIÓN DE MALWARE Y NIVEL DE EFICIENCIA EN SU DETECCIÓN"

Realizado por:

Ing. Geovanni Wladimir Pazmiño Salazar

Director del proyecto:

Ing. Joe Carrión Jumbo, PhD

Como requisito para la obtención del título de

MÁSTER EN CIBERSEGURIDAD

Quito, Septiembre del 2021

DECLARACIÓN JURAMENTADA

Yo, GEOVANNI WLADIMIR PAZMIÑO SALAZAR, con cédula de identidad 170771643-5, declaro bajo juramento que el proyecto desarrollado, es de mí autoría, no ha sido presentado previamente para ninguna calificación o grado profesional y que he citado debidamente las referencias bibliográficas incluidas en este documento.

Además, cedo mis derechos de propiedad intelectual sobre este trabajo, a la UNIVERSIDAD INTERNACIONAL SEK, como corresponde, de acuerdo a lo establecido por la Ley de Propiedad Intelectual, su reglamento y la normativa vigente.

1707716435
GEOVANNI
WLADIMIR
PAZMIÑO SALAZAR

Firmado digitalmente por 1707716435 GEOVANNI WLADIMIR PAZMIÑO SALAZAR
Fecha: 2021.10.20 13:19:14 -05'00'

Geovanni Wladimir Pazmiño Salazar

C.I: 1707716435

DECLARATORIA

El presente proyecto de investigación titulado:

“ANÁLISIS COMPARATIVO DE DIVERSAS METODOLOGÍAS DE DETECCIÓN DE MALWARE Y NIVEL DE EFICIENCIA EN SU DETECCIÓN”

Realizado por:

ING. GEOVANNI WLADIMIR PAZMIÑO SALAZAR

Como requisito para la obtención del título de:

MAGISTER EN CIBERSEGURIDAD

Dirigido por el Profesor

Ing. Joe Carrión Jumbo, PhD

Quien considera que constituye un trabajo original de su autor

Firmado electrónicamente por:
JOE LUIS CARRION

Ing. Joe Carrión Jumbo, PhD

DIRECTOR

PROFESORES INFORMANTES

Después de revisar el trabajo presentado, lo ha calificado como apto para su defensa oral ante el tribunal examinador.

Firmado electrónicamente por:
JOSE VINICIO FREIRE

Ing. José Freire Rumazo, Mg.

Digitally signed by
JOSE LUIS MEDINA BALSECA
EC

Ing. José Luis Medina Balseca, Mg.

Quito, septiembre de 2021

DEDICATORIA

El ser humano como sus actividades, es social y el compartir tiempo, espacio y esfuerzo para adquirir valores como: respeto, solidaridad, honestidad y ética, es su principal función.

Los sentimientos nos llevan a construir unidad familiar y la de nuestro entorno, es así, que existen muchas personas que brindaron sus palabras de aliento y con sus acciones, supieron guiarme durante mi formación académica a quienes les debo mucho.

Preguntaba a mi Hijo Nicolás, cuando era pequeño, ¿por qué no terminas de comer de manera simultánea tus alimentos? y su respuesta fue "Lo mejor, se deja para el último", ¡vaya!, que me ha marcado mis actividades.

A mi Madre, Prof. Elvita Salazar, por su esfuerzo, tenacidad y por edificarme como persona,

Para mi Padre, Dr. Edgar *Pazmiño* †, quien me acompañó en la formación profesional.

Para mi Hermano, Mag Lenin Darío Pazmiño †; que estuvo en todo momento como mi mejor amigo.

Para mis Hijos, Valentina Nicole y Nicolás Sebastián, quienes cambiaron mi vida, con los cuales comparto el grado de responsabilidad, proactividad y el amor que nos profesamos.

Para mi familia: *Inesita*†, Amadorcito†, Piedacita, Ernesto, *Jorge*†, primos y amigos, quiénes en diversos momentos, contribuyeron para la consecución de esta etapa.

Para la persona que, en aquellos momentos difíciles, estuvo a mi lado con la frase "Todo es pasajero" y quien dio equilibrio a mi vida, Alexandra Arosteguí.

A la Institución Internacional Sek, porque he sido parte de ella en todos sus frentes: Profesor: Padre de Familia y en esta última etapa, Alumno, durante 18 años; se fue modelando este proyecto, como fruto de su ideario, organización, visión y misión; que cambiaron el futuro, no sólo de quienes fuimos parte de ella, sino también de nuestras familias, generando Bienestar.

A todos ustedes, familia, amigos e institución educativa, a quienes entrego este proyecto de investigación.

AGRADECIMIENTO

Hay tantas cosas por agradecer,
Muchas más para reflexionar,
Con Docentes de calidad,
Compañeros al principio,
ahora amigos para siempre
Fuimos acogidos por el campus UISEK
Y entre noches extendidas hasta el amanecer,
Logramos tareas cumplir.
Agradecer a Christian, José Luis, Frankie y Fabián,
En el Primer Nivel
Verito, Antonio, Frankie y José Luis,
en el Segundo,
Joe, José, Juan Xavier y Fabián
En el Tercer Nivel.
Llegamos sin conocernos
y permaneceremos unidos hasta el final
Juan Carlos e Iván, Álvaro y David
Wagner y Joel, Freddy y Benjamín
Jaime y Daniel, Edwin y Lorena
Eli y Yo.
Todos conjugaron para a la meta, llegar
Felicitaciones amigos.
Es hora qué desborden sus familias de alegría
Que se esculpa en nuestro espíritu
el esfuerzo de hoy, el mejorar de mañana
Y el disfrutar siempre de él
Es hora de sonreír; la sonrisa,
es la mayor riqueza para entregar.
Es el escudo del alma para no sufrir
y la medicina para sanar
Felicidades
Primera promoción de Ciberseguridad
25/07/2020

Destacar la guía proporcionada por mi Tutor Ing. Joe Luis Carrión, PhD. para la consecución de este proyecto.

Mi reconocimiento especial a mi Amigo y Profesor, Christian Pazmiño, precursor de este camino, quién me invito a la carrera e hizo que me inscribiera, fue el inicio y que hoy culmina.

A mi Padre Celestial, que nos ha permitido caminar, lograr metas y conseguir objetivos, brindándonos las oportunidades para alcanzar fortaleza, sabiduría y bienestar.

Resumen:

El presente proyecto está orientado a realizar una revisión sistemática de investigaciones publicadas, enmarcadas en tres metodologías para la detección de malware: procedimientos que se ejecutan internamente, interdependencia, y la tendencia actual y que son las siguientes: Hooking (Inyección de código) para evitar la ejecución de archivos o rutinas .exe o.dll, que podría incluir un malware (código malicioso), Redes Bayesianas, método clasificador de código (maligno o benigno), secuencia de bytes binario, necesario como data para procesarse en redes neuronales de aprendizaje profundo e Ingeniería inversa, para obtener información de la estructura, familia a la que pertenece o variantes, dependencia de procesos y funcionamiento de aplicaciones. Finalmente, se obtuvo un registro cuantificado de: la interacción entra las metodologías, sus procesos y métodos y la tendencia actual para su aplicación.

Palabra claves: Malware (código malicioso), Detección de Malware (Malware Detection), Metodologías (Methologies), Redes Bayesianas Bayesians net, Ingeniería inversa, Llamadas al sistema API, Hooking, Secuencia de código binario. CIS (CENTER INTERNET SECURITY)

Abstract

This project is aimed at a systematic review of applied research, framed in three methodologies for the malware detection, the procedures that are executed within each one, the interdependence to be executed, and the current trend.

These are the following: Hooking (code injection) to prevent the execution of files or routines, exe, dll, that could include malware (malicius code), Bayesians Networks Code classifiers method (malignant or bening), binary secuence of vites necessary to be processed in neural systems of Deep learning and reverse engineering, to obtain information on the family structure to which it belongs or variants, process dependency and operation. Finally a quantified record was obtained of: the interaction between the methodologies, their processes and methods and the current trend for their application.

ÍNDICE GENERAL DE CONTENIDOS

CAPÍTULO I

INTRODUCCIÓN

1.1. Problema de Investigación

El acceso masivo a la tecnología en comunicaciones ha permitido una interacción entre usuarios, para compartir mensajes y datos; casi siempre de manera libre, espontánea y fácil.

Sin embargo, esto genera riesgos que determinan que dicho contenido sea vulnerable y esté expuesta, dependiendo del uso al cual se oriente. La necesidad de proteger los activos de información de una empresa ante la inminente posibilidad de robo por varias razones, han promovido la investigación para encontrar procesos y metodologías que: protejan, prevengan, establezcan políticas de seguridad, incorporación de equipos para monitorear brechas de seguridad, herramientas que protejan la integridad, confiabilidad y disponibilidad de la información, así como también de la infraestructura que la soporta.

1.1.1. Planteamiento del Problema.

Actualmente se han desarrollado muchas plataformas y equipos para blindar la información ante amenazas y cuyas funciones principales son las siguientes: escanear vulnerabilidades, cuantificación de riesgos, análisis de datos, adquisición y configuración de dispositivos e infraestructura de redes (firewalls), y como parte de la gobernanza de seguridad, está establecer políticas de seguridad (normas para uso de equipos y datos), plan de recuperación ante desastres (DRPs) y plan de continuidad de negocio (BCPs), para garantizar un nivel de operatividad de las empresas en cuanto a la seguridad organizacional y de la información.

Existen muchos procesos de ataques como el Phishing, Ingeniería social, ataque al firmware, exploits, backdoors, y mucho más, mediante MALWARE (software malicioso), que de acuerdo a la Figura 2, destaca un 27 % de ataques realizados, solamente superado por Phishing (29% correo electrónico).

Este es el motivo por el cual se propone analizar las siguientes metodologías para detectar MALWARE: Redes Bayesianos (Clasificadores), Ingeniería inversa Análisis estático y dinámico (ambiente controlado) y por llamadas al Sistema Operativo (Hooking) mediante una revisión sistemática.

La interacción y la integración de servicios y dispositivos móviles como: tabletas, teléfonos celulares, Smart TV, laptops, plataformas de servicios, etc., exigen la protección de datos personales y empresariales, mediante políticas y hábitos en relación al servicio de Internet y aprovechamiento de lo que se puede explorar y encontrar en la red.

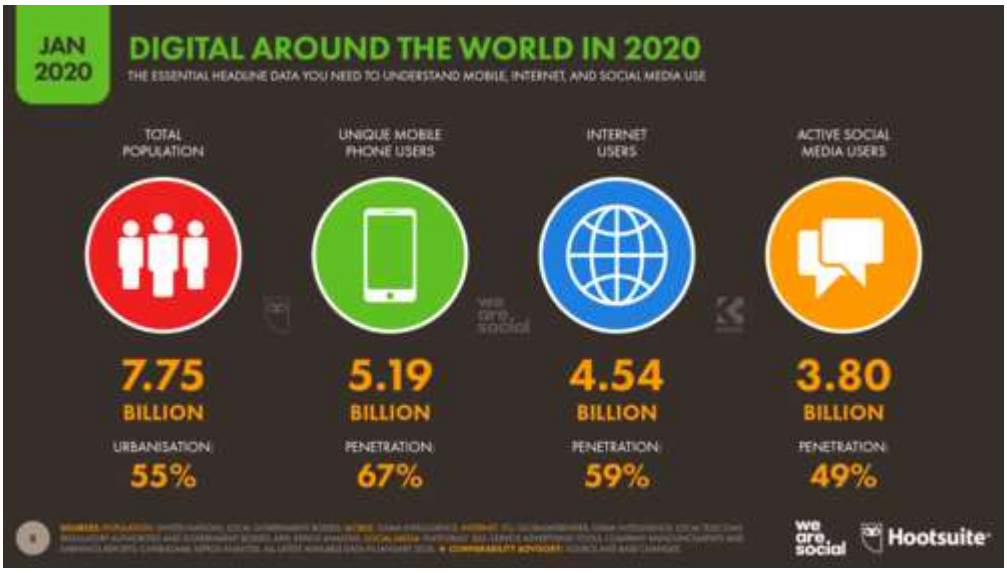

Figura 1Usuarios de Internet enero 2020
Fuente: Digital2020, (Kemp, 2020)

Como se puede apreciar en la Figura 1, el crecimiento de usuarios al Internet y medios tecnológicos, cada vez aumenta y esto se debe a la facilidad de contar con dispositivos inteligentes que incorporan estos servicios.

Es muy común que las personas se conecten a redes y portales de servicio, sobre todo inalámbricas, sin seguridades, donde el servicio de Wi–Fi, es gratuito a cambio de datos de los usuarios, convirtiéndose en blancos perfectos para sufrir ataques y vulneración de derechos.

A nivel de empresas privadas e instituciones públicas, en las cuales se debe preservar los activos (todo aquello que tiene valor para la empresa), es vital impedir el acceso de personas no autorizadas; siendo primordial la autenticación

y la autorización para evitar: la fuga, alteración de información y amenazas a la infraestructura; y garantizar el funcionamiento normal de las mismas.

Invertir en seguridad es costoso y es la principal razón para que, la mayoría de las empresas queden abandonadas a la suerte, sin salvaguardar lo fundamental: la privacidad de las personas, integridad de la información y la solidez de la infraestructura.

Se debe contribuir a garantizar la seguridad personal e institucional y a la vez, aprovechar los recursos y servicios que ofrece la tecnología para ser aplicadas.

"Engaños por correo electrónico, descargas de software pirata y publicidad invasiva con códigos maliciosos son algunas de las principales modalidades de ataques cibernéticos que vivió América Latina en el 2019"(El Tiempo, 2019)

Según (El Tiempo, 2019), ataques por malware móvil hasta el 2019, se reportaron cerca de 6,4 ataques por segundo, considerando que se cuenta con: aplicaciones financieras, estatales, correo electrónico y redes sociales, han demostrado lo vulnerables que son, a pesar de las aplicaciones para reducir los riesgos que existen.

Los equipos del Internet de las Cosas (IoT), como son tabletas, televisores inteligentes SmartTV, electrodomésticos inteligentes, etc., son dispositivos que están expuestos y que pueden ser aprovechados para propagar código malicioso (malware) a través de correos electrónicos.(*TENDENCIAS 2016 (IN) SECURITY EVERYWHERE*, ESET (Enjoy Safer Technology), pp. 1-73)

La falta de capacitación sobre seguridad de la información y la importancia de la privacidad de los datos que utilizan, tanto instituciones públicas como privadas, han minimizado las consecuencias de no tomar medidas de prevención. Los últimos acontecimientos políticos, en los que se denunció la exposición de datos de los ecuatorianos en el Internet, inducen a pensar que no hay conocimiento y peor aún concientización sobre los riesgos y consecuencias que significa tener libre la información y de la necesidad de eliminarlos.

De acuerdo con la Figura 2, se observa que el tipo de ataque más utilizado es el Phishing seguido por Malware/Ransomware y denegación de servicios, los

cuales se propagan en una organización a través de los trabajadores y las áreas más susceptibles como son: infraestructura de TI, aplicaciones legacy y dispositivos conectados a Internet (itResearch, 2020).

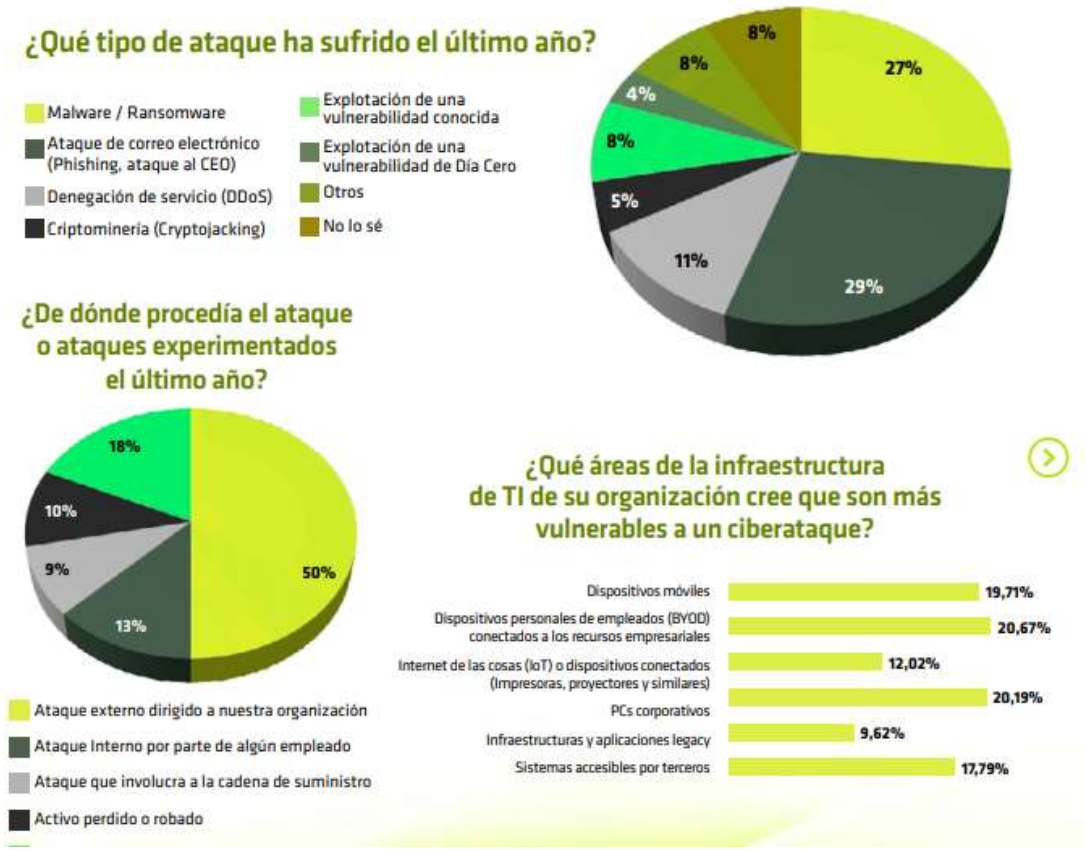

Figura 2Estadística comparativa del tipo de ataque y el área vulnerable
Fuente:(itResearch, 2020)

En cuanto a los números de ataques se estima que:

- El número de contraseñas existentes alcanzan alrededor de los 300 billones y este crecimiento se debió al Covid 19, porque el número de usuarios con conexión a Internet y plataformas obligó a mantener servicios como la educación con enlaces poco seguros. Cómo consecuencia inminente de esta pandemia, también aumentaron los ciberdelitos, es así que, 230000 nuevos malware se produjeron y continúan en ascenso, según Panda Security.
- El 43% de los ciberataques son dirigidos a los pequeños negocios (Natour, 2017).
- Más de 3 millones de golpes de Crypto Jacking, entre Enero y Mayo del 2018. (Giusto, 2018)

- El número de variantes de malware de crypto jacking creció de 8 en el 2017 a 25, en Mayo del 2018 (Giusto, 2018).
- Según Computer Weekly la inversión en Seguridad Informática llega a los 124 billones, mientras que el costo de un ciberataque exitoso es de más de 5 millones de dólares o $ 301,00 por empleado.

El mayor objetivo de un ataque es apropiarse de datos y la pérdida de estos, representa un 43% de los costos, como por ejemplo el ataque a Equifax que superó los 4 billones de dólares en pérdidas.

Esta es la causa por la cual, es necesario contar con un ordenamiento legal; una ley de Privacidad y Protección de Datos, su uso y aplicaciones, que avalen la autodeterminación que ejerce una persona o institución sobre la información que se encuentra almacenada en entidades públicas (instituciones estatales: IESS Ministerio de Economía, Registro Civil o empresas de servicios públicos, etc.) o privadas (bancos, empresas de salud, proveedores de productos y servicios), referente a su divulgación y uso. Esto significa que existe una susceptibilidad y vulnerabilidad, desde cualquier lugar, por cualquier persona, y en cualquier momento; quedando expuesto a sufrir ataques que pongan en riesgo la integridad personal, familiar, institucional, tanto física, psicológica y patrimonial.

En nuestro país, los bancos y algunas empresas grandes ya lo hacen, pero falta mucho por implementar y concientizar.

A nivel de empresas privadas e instituciones públicas, en las cuales se debe preservar los activos (todo aquello que tiene valor para la empresa), es vital impedir accesos de personas no confiables; siendo importante la autenticación y la autorización, para evitar la fuga y alteración de la información, amenazas a la infraestructura, mismas que pondrían en riesgo el funcionamiento normal de la empresa.

Para disminuir este problema, se necesitaría un nivel de inversión elevada, además de un equipo de expertos, para garantizar el funcionamiento normal de la empresa, debido a que existen herramientas y servicios que tienen un alto costo.

La prevención, es una medida para mitigar los riesgos, por ello, se propone analizar distintas metodologías, para determinar el comportamiento de

aplicaciones con malware, que permita escanear vulnerabilidades, analizar ataques, emitir alertas y procedimientos para minimizarlos.

Para controlar lo diagnosticado, es necesario establecer campañas de difusión acerca de amenazas, ataques y sus consecuencias. Actualmente se adquiere e implementa equipos tecnológicos como Firewalls, IDS (Sistemas de Detección de Intrusos), y Plataformas para la detección, alerta y sugerencias del procedimiento más adecuado.

1.1.2. Formulación del Problema

La apropiación de información privada, la susceptibilidad a un uso inadecuado como: borrar o cifrar datos (para extorsionar a cambio de devolverla), obtener ventajas sobre la pertinencia de la misma y lograr ventaja con relación a su competencia, realizar compras virtuales en las que no se necesita autenticación (estafa), son problemas comunes que a diario se presentan y que impiden el crecimiento de las instituciones, de las personas que pertenecen a ellas, así como también de colaboradores externos y que a pesar de la implementación de equipos tecnológicos y la aplicación de procedimientos existentes, resultan ser poco eficaces, o no completos por sí solos, esto se comprueba de acuerdo al último estudio realizado por (Bissell et al., 2020, pp. 1-46), sólo el 17 % de las organizaciones consideradas como grandes, realizan CIBERRESILENCIA, basadas en cuatro categorías:

1. Detener el mayor número de Ataques.
2. Detectar brechas de seguridad lo antes posible
3. Corregir las brechas.
4. Reducir el impacto,

concluyendo, que las empresas líderes que invirtieron en seguridad tienen cuatro veces más probabilidades de detectar una brecha y cuando sus sistemas fallaron, pudieron corregirlas en un 96%. Lo hicieron en 15 días en promedio, mientras que las demás lo hicieron en más tiempo (casi el doble, un mes o más). El considerar como un gasto a la inversión en Seguridad de la Información y no contar con

planes de prevención, políticas de seguridad (metodologías de análisis de malware, software actualizado) genera vulnerabilidad. En el reporte de (ESET, 2021), en el Ecuador se han registrado durante los últimos 12 meses: 5% de secuestro de información, 11% en DDoS (denegación de servicios), 11% en explotación de vulnerabilidades, 15% en ataques de Ingeniería Social, 21 % en acceso a base de datos o aplicaciones restringidas y 40% de malware.

Nuevamente se trata de proteger, no sólo los activos sino lo intangible: sus MARCAS, la REPUTACIÓN y LA CONFIANZA (Bissell et al., 2020, pp. 1-46).

1.2. Objetivo General

Valorar metodologías de detección de malware: Hooking (llamadas al Sistema Operativo), Redes Bayesianas e Ingeniería inversa, mediante una revisión sistemática para la comparación, nivel de eficiencia y establecer un Plan de Prevención.

1.2.1. Objetivos Específicos:

- Encontrar la cadena de búsqueda para las bases de datos que contienen artículos científicos de revistas publicadas, para la realización de la revisión sistemática.
- Analizar la metodología de Hooking, Redes Bayesianas e Ingeniería Inversa, y los artículos encontrados bajo criterios de quartiles Q1 y Q2, mediante la recopilación en bases de datos de publicaciones de revistas y artículos.
- Clasificar las publicaciones, de acuerdo con las metodologías mencionadas y la determinación de tendencias de investigación actuales para la comparación de los procesos
- Impedir amenazas y ataques de malware estableciendo un Plan de Prevención, enfocado en los Controles Críticos de Seguridad (CIS),

1.3. Justificación

La revisión sistemática de métodos propuestos en proyectos e investigaciones efectuadas, enmarcadas en tres metodologías: llamadas al sistema (System Call Procedures), Redes Bayesianas e Ingeniería Inversa, permitirá explorar: procedimientos, alcance y eficiencia, al momento de detectar malware, contribuyendo a encontrar la solución para evitar dichos ataques, adoptando modelos que se desprenden del análisis para citar un Plan de Prevención.

1.4. Marco Conceptual

Este proyecto se basa en realizar un análisis comparativo entre las metodologías para la detección de malware, Ingeniería inversa (análisis estático), Ambiente controlado (laboratorio aislado), Redes bayesianas (Heurísticas) y llamadas al Sistema Operativo, System Call Procedure (Método Hooking), con el fin de recomendar el método más eficiente para detectarlo.

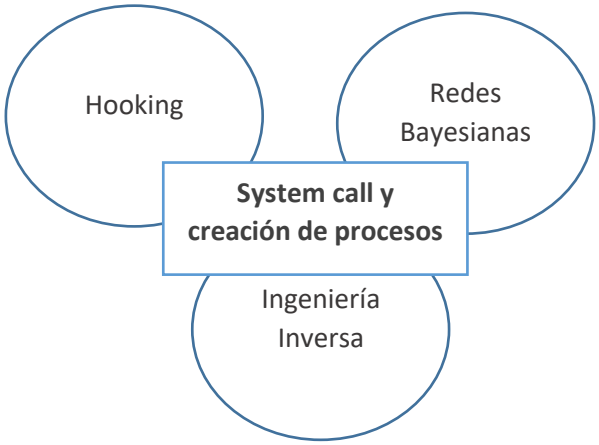

Figura 3Relación de las metodologías de detección de malware
Fuente: Elaboración propia

Algunos de los conceptos utilizados se exponen a continuación:

Vulnerabilidad: Es una debilidad de un sistema o procesos asociados que permitiría a una amenaza, actuar contra un activo. (b2b Consultores, 2019)

Ataque: manera de explotar una vulnerabilidad

Amenaza: intención declarada de infligir lesiones, daños u otro tipo de hostilidad a alguien (Soanes, 2000).

Ciberamenaza: cualquier acción que pueda resultar en un esfuerzo no autorizado para impactar negativamente la seguridad, la confianza y la disponibilidad de un sistema de información o de información almacenada en dicho sistema. Estado susceptible de que la información sufra una pérdida de confidencialidad, Integridad y disponibilidad.

Según (Velasco, 2014), existen los siguientes tipos de malware:

Virus: es un software que altera o destruye las funcionalidades de una aplicación, ejecutable, sustituyendo parcialmente el código fuente.

Propiedades de virus: algunas funciones de virus son las siguientes: («Características, Estructura y Clasificación de Virus Informáticos», 2013),

Cuando un Virus llega a memoria, puede afectar a múltiples archivos y a la red de ordenadores a la que pertenece, debido a que, muchos programas o archivos deben acceder a ella.

Otros, tienen la capacidad de modificar su código y generar variantes, lo que hace difícil su detección. Estos se llaman Polimórficos.

También se activan cada vez que abres un archivo infectado, por lo tanto, son No Residentes en Memoria.

Además, existen otros que tienen la capacidad de adjuntarse a archivos del ordenador y luego atacar al mismo en su totalidad. Estos se conocen como Furtivos (stealth).

Hay algunos que se ocultan muy bien, que no pueden ser detectados, como aquellos que generan infección al hardware, se graban en sector de arranque MBR o en el CMOS, capaces de dañar permanentemente la placa y aun formateándolo, no se pueden reparar.

Según (JÍMENEZ, JAVIER, 2020) malware con código ofuscado, es el mismo código original, pero que tiene otra forma que no puede ser identificado, con la finalidad de evitar ser detectado. Este código nuevo, en ese formato, es difícil de distinguir entre uno normal con uno maligno.

Estructura de un Virus

De acuerdo a (Chilán & Macías, 2014, pp. 1-22) la estructura de un virus tiene los siguientes módulos:

Reproducción, se encarga de ejecutar los scripts o rutinas de archivos ejecutables para tomar control del sistema e infectar a otros, mediante archivos o de la red a la que pertenecen.

Ataque. Estas rutinas al activarse afectarán al objetivo, causando un daño adicional hasta lograr a inutilizar la información del ordenador.

Defensa, esta parte del virus, protege de acciones de detección o eliminación del mismo, haciendo más difícil su remoción con la finalidad de ocasionar el mayor efecto(«Características, Estructura y Clasificación de Virus Informáticos», 2013).

Malware, en el estudio de (J Kolter & Maloof, 2004) se desarrolló una aplicación basada en métodos inductivos con árboles de decisión, máquinas vectoriales de soporte y refuerzo, obteniendo 1971 ejecutables maliciosos de un total de 3622, utilizando técnicas de aprendizaje automáticas y minería de datos.

Una de las amenazas más graves a/contra los sistemas informáticos son los programas o aplicaciones de tipo malware, (Tian et al., 2008, pp. 69-76) hicieron pruebas con algoritmos de aprendizaje autónomo, analizando la función de Longitud de prueba de frecuencia que describe a troyanos, logrando clasificarlos por su duración, constituyéndose en un método económico y escalable.

Otro concepto es**:** "Código y software malicioso, se refiere a un programa que se inserta en un sistema, generalmente de manera encubierta, con la intención de comprometer la confidencialidad, integridad o disponibilidad de los datos, las aplicaciones o el sistema operativo de la víctima o molestar o interrumpir a la víctima" (Anaid, 2019)

Métricas de seguridad: son un valor cuantificable para conocer los niveles de acción de las actividades que se hacen o dejan de hacer para reducir riesgos antes ciberataques (Ciberseguridad, s. f.). Entre los estándares y métricas aplicados en los Estados Unidos, tenemos las siguientes:

PCI DSS, estándar de seguridad para tarjetas de pago (crédito)

- Mantener un firewall activo para escanear tráfico de red y bloqueo de acceso no autorizados
- Cambios y actualización de datos proporcionados por el proveedor de servicios
- Protección de datos del titular. (Cifrado)
- Cifrado en la transmisión de datos, sobre todo en redes abiertas
- Protección y actualización de los sistemas contra malware
- Desarrollo y mantenimiento de sistemas y aplicaciones seguros
- Restricción acceso a los datos del titular
- Autenticar el acceso a los componentes del sistema
- Seguimiento y monitorización del acceso a los datos del titular
- Pruebas frecuentes a los sistemas y procesos de seguridad
- Difusión de Políticas de seguridad actualizadas a todo el personal

Métricas

HIPAA, ley vigente de privacidad de datos

GDPR, Reglamento general de Protección de Datos, USA, (Coip y ley de protección de datos, Ecuador)

CCPA, Ley de Privacidad y defensa del Consumidor

CPS 234, equivalente al esquema de seguridad de la información (EGSI)

LGPD, Ley General de Protección de Datos

PIPEDA, Ley de Protección de Información Personal y documentos electrónicos

FIPA, Ley de Protección de la Información, válida en USA

The SHIELD Act, Ley de Alto a los Hacks y mejora de datos electrónicos, USA (ley de comercio electrónico y de telecomunicaciones, Ecuador)

GLBA, Ley y Federal de instituciones financieras, USA. Ley reguladora de Instituciones Financieras.

FISMA. Ley federal de la Administración de Seguridad de la Información.

Tipos de Malware: La universalidad del uso del Internet para mejorar las comunicaciones, optimización del tiempo y el avance de la tecnología, ha permitido el desarrollo de las instituciones y emprendimientos, pero paralelamente, la competencia, la ambición y empleados descontentos con el trato y salarios bajos, provocaron que se creen

diferentes tipos de procesos (scripts), para aprovechar vulnerabilidades presentes, tanto en Software como en Hardware

Gusanos: los gusanos no son otros que los virus, pero con la capacidad de duplicarse y pueden infectar varios ordenadores de manera autónoma. Se alojan en la memoria y actúan en las redes de ordenadores.

Troyanos: este tipo de malware manipula ordenadores infectados de forma remota por un "Pirata Informático", accediendo a diferentes recursos como webcam, pantalla, teclado y archivos

Entre estos tenemos: Puertas Traseras o Back Doors (permanecen activas para ingresar al sistema mediante el acceso remoto),

Keyloggers (actúan sobre las teclas del keyboard y registran su actividad),

Password Stealer (roban contraseñas especialmente de correo electrónico),

Bancarios (la especialidad es el robo de credenciales y datos de cuentas, tarjetas de crédito o débito),

Botnets (crean una red zombie para ataques de denegación de servicios DDOS), Downloaders (tipo de troyanos para descargar malware e infectar equipos), Spyware (sirve para espiar, capturan historial de navegación, cookies, datos de usuario, contraseñas que sirven para suplantación de identidad, especialmente.

Adware (redirige a páginas publicitarias invasivas mediante Pop-ups o ventanas emergentes).

Ransomware (cifra la información del usuario y se convierte en un medio de extorsión económico para restablecer la información y recuperar los datos),

Exploits (explotan las vulnerabilidades de las aplicaciones y llegan a través del Internet y sirven para descargar otro malware más complejo, de manera transparente al usuario), además, menciona (Férnandez, 2019)**,** (son falsos antivirus que despliegan mensajes falsos sobre una infección del sistema y sugieren actualizaciones para instalar un malware más complejo,

Malware polimórfico, cualquier tipo de Malware citado con anterioridad con la capacidad de transformarse frecuentemente, alterando su apariencia para mantener el algoritmo, para hacer más difícil su detección,

Auditoría informática: según (Castañón, 2012) "es la revisión y evaluación de todos los aspectos de los sistemas automáticos de procesamiento de la información, incluyendo los procedimientos no automáticos relacionados con ellos y las interfaces correspondiente"

Sistemas de información informática: dice Chen, 2019. "es un conjunto de datos que interactúan entre sí con un fin común que ayudan a administrar, recolectar, recuperar, procesar, almacenar y distribuir información relevante para los procesos fundamentales y las particularidades de cada organización".

Empaquetadores: Un "empaquetador es un programa ejecutable de autoextracción y de tiempo de ejecución que, acumula distintos tipos de malware en un sólo paquete Los empaquetadores tienen la capacidad de hacer mutar sus firmas con el tiempo, lo cual dificulta mucho más la detección y eliminación del malware"(ESET, s. f.).

Fuzzing: "se considera el método más prometedor para descubrir vulnerabilidades de protocolo en seguridad de red de Internet de las cosas (IoT)"(Luo , et al., 2018).

EndPoints: La pregunta hecha por («Advance Networks», 2021), ¿Cuáles son las principales herramientas de protección a EndPoints? se orienta a las circunstancias actuales ocasionadas por la Pandemia del Covid 19 que dio lugar al incremento de los usuarios y servicios ofertados por Internet, dónde el trabajo remoto y compras en línea son la realidad económica y comercial actual.

La implementación de EndPoints ("dispositivo informático remoto que se comunica con una red a la que está conectado"(«ADMWARE Advance Data Manegement», s. f.)) siendo estos Ordenadores, Laptops, Tabletas, Servidores o estaciones de trabajo, constituyéndose en puntos finales susceptibles para que los ciberdelincuentes ejecuten código para explotar vulnerabilidades.

EPP, Endpoint Protection Plataform, es una herramienta de defensa que sirve para detectar y bloquear amenazas con las siguientes funciones: Antivirus, Antimalware, Prevención de Intrusiones IPS, Prevención de pérdida de datos DLP, Prevención de exploits, tecnología anti-ransomware («Tecnozero», s. f.).

EDR, Endpoint Detection and Response, es una herramienta para monitorizar eventos de usuario, archivos, procesos, registros, memoria y red, detectar y prevenir amenazas avanzadas APT en análisis continuo, en busca de amenazas desconocidas.

Un EDR utiliza técnicas de Machine learning y analítica, Sandboxing, categorización de los incidentes, rastreo del origen y evolución de malware, eliminación, cuarentena de ficheros y restauración al estado anterior a la infección.(«Tecnozero», s. f.)

Detección y respuesta extendidas (XDR) tiene funciones en "Endpoint, Servidores, Correo electrónico, firewalls, servicios de nube, mediante el uso de indicadores para identificar, señalar y neutralizar una amenaza, incluido aprendizaje profundo, anti-ransomware, protección de ataques sin archivos, listas blancas, monitoreo de integridad de archivos"(«https://www.sophos.com/en-us/products/endpoint-antivirus/edr.aspx», s. f.).

1.5. Estado del Arte

Valorar las diversas metodologías para detectar ataques de malware, es el objetivo de este proyecto; para ello, se ha estudiado varios trabajos realizados con relación a los sistemas detección, los cuales permiten conocer la estructura, funcionalidades y procedimientos de malware, para que, en una parte complementaria a este trabajo, se emita una rápida respuesta para mitigar, bloquear e impedir su función (ciberdefensa), que es la de vulnerar los sistemas informáticos.

Los sistemas comerciales tratan de explorar códigos maliciosos por medio de firmas y la agrupación de ellas, se denominan heurísticas, las mismas que se basan en el comportamiento (cambios inusuales en los archivos) y la misión de éstas, es la de detectar software malicioso.

A través de la plataforma Scopus, como una base de trabajos de investigación y publicaciones, se determina que existen artículos de revistas o conferencias con relación a la detección de malware y de esta manera, realizar una revisión sistemática que tengan relación con nuestro proyecto.

El uso de operadores booleanos universales es fundamental para depurar la información que se requiere y la forma de construir estás cadenas de búsqueda, son el éxito para optimizarla, ya que se puede tomar estos trabajos como base, para avanzar en la creación de conocimiento nuevo.

1.5.1. Metodología de Hooking (llamadas al Sistema)

Actualmente, la interfaz de programación de aplicaciones (API), es utilizada por un software para comunicarse con el Sistema Operativo (SO), tendiente a ejecutar un servicio o utilizar un recurso, tales como: abrir, copiar, grabar, eliminar, imprimir un archivo o administrar memoria, ejecutando para ello llamadas al sistema.

Todo pedido que se realice por alguna aplicación o software, por un servicio o recurso requiere una llamada al sistema como se confirma en el proyecto "Hooking the native API and controlling process creation on a system-wide basis," proponiendo inyección de código a la estructura de las librerías .dll necesarias para ejecutar dichas solicitudes (llamadas al sistema para creación de procesos, almacenamiento en memoria, reserva de espacio en disco y otras) para compararlos con aquellos procesos precalificados como seguros por el sistema operativo y definir si éstos son seguros o no, (Bassov, A, 2005).

Según (Murthy et al., 2019, p. 4991), analizar la secuencia de llamadas de API, "permite identificar actividades maliciosas (captura de pantalla, Hooking, Descargador, Enumerar el proceso, Anti depuración, Sincronización, Key Logger y Dropper)", mediante la Concordance de código (patrones similares que se repiten). En ésta, se destaca la condición de uso obligatorio del procedimiento "llamadas al Sistema" para detectar malware.

Como se determina en el estudio " Early detection of ransomware by indicator analysis and WinAPI, call sequence pattern, se utilizaron conjuntos de **indicadores de comportamiento y bibliotecas Microsoft Detours, para**

conectar las secuencias de llamadas de la API de Windows, comprender el comportamiento y el filtro en tiempo de ejecución, eliminar el ransomware del software benigno" (Sharma & Kant, 2018, pp. 201).

Un ataque de malware "invocará operaciones privilegiadas en un kernel de Windows o Linux en la búsqueda de procesos a nivel de usuario sin requerir que esos procesos requieran a llamadas al sistema reales correspondientes a las operaciones", (Srivastava, 2008, p. 421)

Las funciones que hacen llamadas al sistema operativo a ejecutarse son un ejemplo de reutilización de código y se la hace mediante la intercepción de las llamadas a través de una función llamada Hooking. El insertar una función Hook al programa analizado, implementa funcionalidades para registrar las llamadas a un archivo log. El malware al ejecutarse en modo usuario debe hacer las llamadas al sistema operativo, creando los registros conocidos como Trazas, las mismas que son convertidas en grafos para realizar comparaciones entre muestras de goodware (software confiable, útil) y malware para identificar variaciones polimórficas de amenazas conocidas.

Sin embargo, aquellos que utilizan el Kernel, tienen otro comportamiento y frecuentemente el malware, utiliza código auto-modificable y empacadores ("sintácticamente diferente pero semánticamente equivalente"), lo que hace muy difícil ser detectados, para lo cual, se utilizará la rutina de desempacado (volver a su estado original).

Un enfoque de detección dinámico de malware es Markhor, para casos de ofuscación de llamadas al sistema, empleando "dependencia de datos de llamadas del sistema y las secuencias de dependencia de control de llamadas del sistema para crear una lista ponderada de patrones maliciosos. Luego, la lista se utiliza para determinar los procesos maliciosos se extrae la similitud de las secuencias de llamadas del sistema de archivos con un patrón malicioso basándose en un algoritmo difuso y se determina la naturaleza del archivo" (Lajevardi, AM et al., 2021). La evaluación está alrededor de 0,982 de exactitud y 0,976 de precisión.

De acuerdo a (Thien-Phuc et al., 2020), la construcción de diagramas de flujo basado en llamadas API críticas, contribuyen a la deducción de los recursos manipulados por la aplicación maliciosa (Malware) que pueden ser ofuscadas o cifradas para evadir su detección, sin embargo, el análisis dinámico, utiliza herramientas de Hooking como xPosed o Frida para detectar su comportamiento que intentan tomar control de los recursos del dispositivo, mediante el modelo denominado HALWatcher (Hardware Abstract Layer) encargado de monitorear el uso de recursos para el sistema operativo.

El método propuesto por (Kim et al., 2018) de análisis dinámico utilizando "un árbol un árbol de sufijos que contienen API (Application Programming Interface) que generan valores probabilísticos de confianza, evitando la sobrecarga de uso de recursos, reduciendo tiempo , reescritura de aplicaciones para rastrear las invocaciones de API, mediante Hooking, que genera una clase con métodos de autocontrol para comparar con listas blancas y negras".

1.5.2. Redes Bayesianas

El análisis algorítmico se basa en funciones matemáticas puntuadas (Análisis Bayesiano) que, al ser comparadas con niveles preestablecidos, determinarán la presencia de un malware o código malicioso.

De acuerdo con (Castellanos, 2015) el análisis de malware se puede hacer de dos maneras: estático y dinámico. El dinámico se realiza mientras se están ejecutando el código, mientras que el estático se lo hace en laboratorio con ambiente controlado. Los programas y aplicaciones para interactuar con los equipos tienen dos maneras de hacerlo: Uno cuando las aplicaciones son ejecutadas en modo usuario, y el segundo, el sistema operativo a nivel de Kernel.

Según Eset, en su publicación sobre Análisis Heurístico, expone que el método utiliza una máquina virtual para resolver el sistema de ocultamiento con técnicas de "empaquetamiento, compresión o cifrado para desofuscar el código malicioso y luego debe ser analizado algorítmica o heurísticamente, mediante una exploración de un antivirus" (Harley, 2007). Existen dos tipos: Heurística pasiva, que "analiza una amenaza potencial mientras es explorada, realizando

un seguimiento a través de las instrucciones del programa antes de brindar el código al procesador para su ejecución" y una activa que crea equipo virtual dentro del motor de exploración, que permite al analizador observar lo que el programa podría hacer si se permite su ejecución en la computadora real.

El criterio versa sobre "enfoques de detección que utilizan análisis estático y aprendizaje automático / profundo con características básicas como: metainformación, códigos de operación y llamadas de Windows API, funciones híbridas y propiedades diseñadas como flujo de datos confidenciales, ofuscación, características invariantes y gráficos de relación API" (Feng et al., 2021).

El análisis de APIs , se orienta a la construcción de gráficos de llamadas al Sistema (en este caso Android), ya que invoca la utilización de recursos y servicios del SO, independientemente del tipo, pues, el objetivo es el aprendizaje de la estructura de malware y su posterior comparación con versiones posteriores de código malicioso, mediante los siguientes algoritmos propuestos "basados en permisos de riesgo en tiempo de ejecución con un enfoque mejorado de selección de características en clusters y el de Bayes (grafo), para proporcionar precisión y detección robusta de malware de IoT "(Kumar & Zhang, 2019, p. 277).

En la Segunda Conferencia Internacional, ML4CS 2019, Machine Learning for Cyber Security, (Sun et al., 2019, pp. 63-77) , expuso la generación de clasificadores de Malware e identificación de características de las llamadas de API para diferenciar malware, a partir de la construcción de gráficas de llamadas de aplicaciones (recuperando la semántica del gráfico de nivel para clasificar aplicaciones y aprovechando la técnica de minería de texto para capturar llamadas APIs), mediante 4 algoritmos, entre ellos el de Naive Bayes, que permite construir la función de vectores mediante CatraDroid.

En (Firdaus et al., 2018, pp. 712-736), se expone el análisis para la clasificación de malware mediante "búsqueda genética (GS), basado en algoritmo genético (GA), para seleccionar las características entre 106 cadenas

y utilizar cinco clasificadores de aprendizaje automático: Naive Bayes (NB), árboles funcionales (FT), bosque aleatorio (RF) y multicapa perceptrón (MLP). Entre estos clasificadores, FT dio la mayor precisión (95%) y la tasa de verdaderos positivos (TPR) (96,7%) con el uso de sólo seis funciones". Se optó por el análisis estático, porque se orienta en el código abarcando posibles actividades porque no ha sido ejecutada mediante ingeniería inversa y recuperar el código completo, mientras que el dinámico, investiga el comportamiento de una aplicación en un intervalo de tiempo definida.

Según (Marastoni et al., 2021), la mayoría de análisis de malware, no es posible llevarlo por la falta de suficientes datos, razón por la cual se generará datos ofuscados para luego entrenar a una red neuronal convolucional (CNN y memoria bidireccional a corto plazo (LSTM), logrando una precisión de alrededor del 93%.

1.5.3. Ingeniería inversa

(Liţă et al., 2017), la capacidad de reacción ante un ataque en el menor tiempo conlleva a estar preparado, esto ocurre porque cada vez se crean nuevas formas de malware para evitar ser detectados, (oligomórfismo y polimorfismo) y cumplir sus objetivos. El código malicioso generalmente tiene variaciones con respecto al original ("registro, intercambio, permutación de rutina, reordenamiento de código, equivalente a sustitución y transposición de código"(Liţă et al., 2017)), sin embargo, los atacantes han orientado su esfuerzo a transformar la forma de presentarse mediante la utilización de un empaquetador como alternativa para evadir la protección que ofrecen soluciones de seguridad, especialmente de ingeniería inversa. Sin embargo, la importancia de obtener código inicial de malware, el cual se ha logrado empleando Emuladores, radica en que éste es más fácil de comparar con los patrones existentes de código malicioso y comprender el funcionamiento interno.

Según (Preda et al., 2008), Identificar estructuras de firmas de comportamiento para detectar malware, durante el seguimiento dinámico de API y llamadas al sistema para encontrar coincidencias binarias, capturar comportamientos de tipo de similitud y reconstrucción de jerarquías. Sugiere que para que los detectores de malware sean robustos, deben manejar Ofuscación y se realizan por medio de llamadas al sistema, ya que es la única manera de interactuar con recursos del SO.

La Seguridad de Red, es parte de la Ciberseguridad y uno de los métodos utilizados es el de Fuzzing, sin embargo, una de las limitaciones para su eficiencia radica en la gran cantidad de archivos de prueba que se necesitan para garantizar una mejor eficacia. Esto llevó a (Luo , et al., 2018) a ensayar, mediante técnicas de ingeniería inversa a protocolos de comunicación, procedimientos para la identificación de formatos de mensajes en la capa de aplicación y así crear archivos de prueba, considerando a éstos, como una secuencia de bytes, de acuerdo a procesos estadísticos que indican los cambios que pudieron ocurrir en la sucesión de ellos.

Existen muchos métodos propuestos para detección de malware y que se orientan al manejo de aprendizaje con redes neuronales, entre ellos se mencionan el análisis de ofuscación persistente como ORDroid usando redes neuronales RNN (recurrentes) y NLP (Natural Language Processing) y LightDroid con una precisión del 99,47% para pruebas (Aghamohammadi & Faghih, 2019, pp. 125-139).

(Siddiqui et al., 2008, pp. 509-510) realizaron la autodetección de malware, utilizando técnicas de minería de datos, clasificadas en tres niveles jerárquicos de acuerdo a las características del archivo (programas binarios), tipo de análisis (estático o dinámico) y de detección (mal uso, de intrusos o de anomalías), también incluyeron el análisis para la clasificación de gusanos utilizando secuencias de instrucciones de longitud variable, por medio del análisis del flujo de control del programa, a través de minería de datos, partiendo de la información contenida en las secuencias de instrucción con una exactitud del 95,6 %.

CAPÍTULO II

MÉTODO

2.1. Marco Teórico

Según (Jarrín, 2019), el futuro de la humanidad se centra en que las comunicaciones sean fáciles, rápidas, fluidas y seguras. Actualmente, se utiliza métodos y dispositivos autónomos, como la inteligencia artificial, e-learning machine y *algoritmos genéticos*, que a cada momento generan respuestas y soluciones que, antes no se las podía visualizar.

El riesgo de sufrir un ciberataque es alto y puede deberse a varios motivos; entre los cuales se cita los siguientes: la inconformidad de empleados, el diseño defectuoso de sistemas informáticos, malos hábitos en el uso de equipos de interconexión y falta de políticas de seguridad (autenticación, autorización y derechos de uso).

Frente a este escenario, se debe realizar acciones que pretenden blindar la infraestructura, eliminar vulnerabilidades e implementar mecanismos de protección, mediante acciones rápidas, simples de accionar y ejecutar.

Entre las principales se tiene: (CONATEL, 2015)

- Actualizar los sistemas informáticos (Sistemas Operativos).
- Implementar Firewalls o proxy-servers.
- Normar el uso de dispositivos móviles a los empleados (Internet).
- Capacitación a los colaboradores para prevenir, enfrentar o remediar dichos ataques.
- Implementar métodos de autenticación más robustos, como los de doble factor o incluir sistemas de reconocimiento facial inclusive.
- Realizar backups o copias de respaldo frecuentes.
- Implementar políticas para la interconexión de redes de ordenadores y los sistemas de comunicación.
- Plan de continuidad de negocio (BCPs)

Este contexto conlleva a la implementación de normas de seguridad (políticas) y el estudio de los patrones de ataques cibernéticos (seguridad de datos, redes y sistemas),

como solución de estas debilidades o vulnerabilidades. La ciberseguridad se constituye en la base de este control y se la describe a continuación.

2.1.1. Ciberseguridad

Ciberseguridad es una parte de la Seguridad de la Información, que se encarga de garantizar la integralidad, confidencialidad y disponibilidad de la información y datos (activos de la información de la empresa).

En la Figura 4 se puede apreciar el ámbito de la Ciberseguridad con relación a la generalidad que es la Seguridad Informática, las áreas que comprende y son las siguientes:

- Seguridad de la aplicación
- Seguridad de red
- Seguridad de Internet

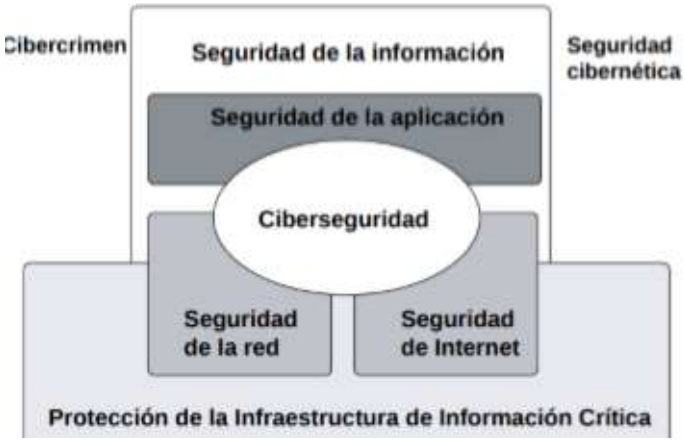

Figura 4Relación entre la ciberseguridad y otros dominios de Seguridad
Fuente: ISO/IEC 27032, 2012 (Giancarlo, 2017)

2.1.2. Ciberataque

Un Ataque Cibernético, es un conjunto de eventos intencionales o no, tendientes a causar daño, robar información o datos, y/o dañar sistemas; comprometiendo la seguridad de las empresa o personas, así como también su funcionamiento.

Según MADBOXPC(Rübke, 2018), la motivación para cometer un ciberataque es diversa, pero las más frecuentes son:

- Política y Social: Un evento de espionaje entre países (estrategia de ciberguerra), comprometiendo sistemas informáticos e instituciones públicas,
- Delincuencia organizada: grupos criminales (cibercrimen), ideológicos (activistas)
- Ganancias financieras: Personas que buscan beneficios económicos a cambio de entregar información relevante para sus intereses.

Un ataque es una serie de pasos ordenados, lógicos y secuenciales, con un objetivo definido para causar daño (apropiarse de información sensible para beneficio personal), rompiendo seguridades mal configurados (vulnerabilidades) para ingresar a sistemas, computadoras o redes de datos de una organización.

Estos ciberataques (Cyber Kill Chain) constan de siete pasos según (INCIBE, 2020):

1. **Reconocimiento:**

   En esta primera etapa, el atacante busca información sobre la organización como: tecnología, datos públicos, tecnología instalada (equipos e internet), aplicaciones y paquetes instalados. Interactúa a través de redes sociales, correos electrónicos o directamente con los empleados, como una actividad normal.

   La finalidad es analizar, cuál metodología será usada y la probabilidad de éxito.(INCIBE, 2020)

2. **Preparación**

   El logro de un atentado depende de esta fase; se planificará las acciones a seguir y construir el vector de ataque que se constituye en todos los pasos para lograr entrar en el sistema de la víctima. Por ejemplo: documento Office, correo electrónico, phishing, pen drive o el uso compartido del internet disponible.

3. **Distribución**

   En este punto se transmite el ataque de acuerdo a lo planificado en el punto anterior, sea documento infectado, correo electrónico u otro método, es decir, aquí ya se alcanza a la víctima.

4. **Explotación**

   Una vez dentro del objetivo, el siguiente paso es la denotación del ataque mediante la explotación de alguna brecha de seguridad o vulnerabilidad, y tomar el control remoto del equipo o la red. Cabe mencionar, que este proceso se realiza en un lapso mínimo para no ser detectado, bloqueado o eliminado.

5. **Instalación**

   En esta etapa se instala el software malicioso (malware) o se ejecuta scripts (programas autoejecutables) para capturar credenciales de acceso, nombres de usuario, DoS (denegación de servicios) y tomar control del equipo o software.

6. **Comando y Control**

   Realizado el paso anterior, se toma posesión del dispositivo y se controla el sistema de la víctima para realizar la acción prevista, sea acceder a servidores, copiar, encriptar, borrar información, hacer capturas de pantalla o acceder a documentos.

7. **Propagación:** Al final, el atacante tiene ya la información y puede extender su acción hacia otros equipos o servidores, convirtiéndose en un ataque cíclico, debido a que se ejecutará de manera automatizada a otros (gusano).

   Esta es la razón, por la cual se justifica la ejecución del presente proyecto, al analizar diversas metodologías de detección de Malware (Software Malicioso) para evitar las consecuencias al ser infectados por ellos.

Dentro de las acciones que pretende la infección de Malware, según (Monnappa - 2018 - Learning Malware Analysis.pdf, s. f.) se conoce las siguientes:

a) Interrumpir las operaciones de la computadora
b) Robar información confidencial, incluidos datos personales, comerciales y financieros
c) Acceso no autorizado al sistema de la víctima
d) Espiar a las víctimas
e) Envío de correos electrónicos no deseados
f) Participar en ataques distribuidos de denegación de servicio (DoS)
g) Bloquear los archivos en la computadora y retenerlos para obtener un rescate.

2.2. Llamadas al Sistema Operativo

El Sistema Operativo, tiene dos funciones:

- Hacer transparente los procesos necesarios ante los requerimientos de las aplicaciones
- Gestionar los recursos del ordenador

Esto significa que para ejecutar una aplicación se envían solicitudes al Sistema Operativo, el cual a través del Kernel (núcleo del Sistema Operativo), administra el

acceso al hardware del ordenador mediante servicios de llamadas al sistema.

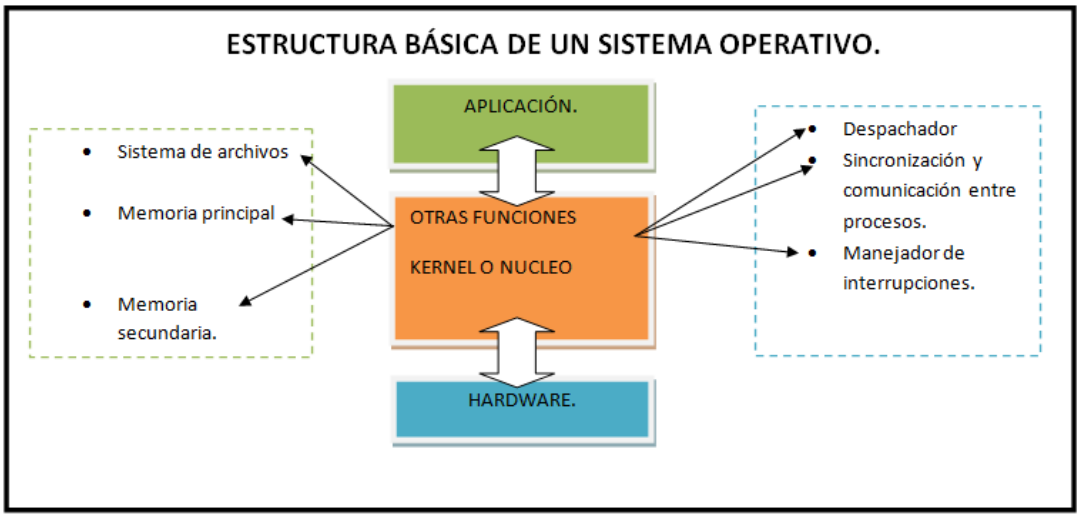

Figura 5Kernel, mediador entre Aplicaciones y Sistema Operativo.
Fuente: («Arquitectura de un sistema operativo», 2016)

Estos procesos se basan en un contador de programa, que varía después de ejecutar una instrucción. Los registros de control se asocian con los procesos y se almacenan en la Tabla de Procesos, mediante un arreglo de estructuras que se ejecutan en ese tiempo. Es imprescindible mencionar que un proceso genera un subproceso o proceso hijo y éste a su vez en otros, construyendo un árbol de procesos. Cada procedimiento ejecuta una llamada al solicitar un servicio mediante una petición, por ejemplo: leer, imprimir, escribir. Las llamadas al sistema entran al Kernel, mientras que las llamadas a procedimiento no lo entran. “El sistema operativo se encarga de ejecutar el servicio solicitado y se ejecutará una instrucción TRAP para transferir el control al Sistema Operativo” (Aguilar, 2020). Una instrucción TRAP ("interrupción iniciada por un proceso en ejecución” (Deitel,1993)) permite gestionar servicios como acceso a puertos de comunicación, sistemas de archivos o procesos.

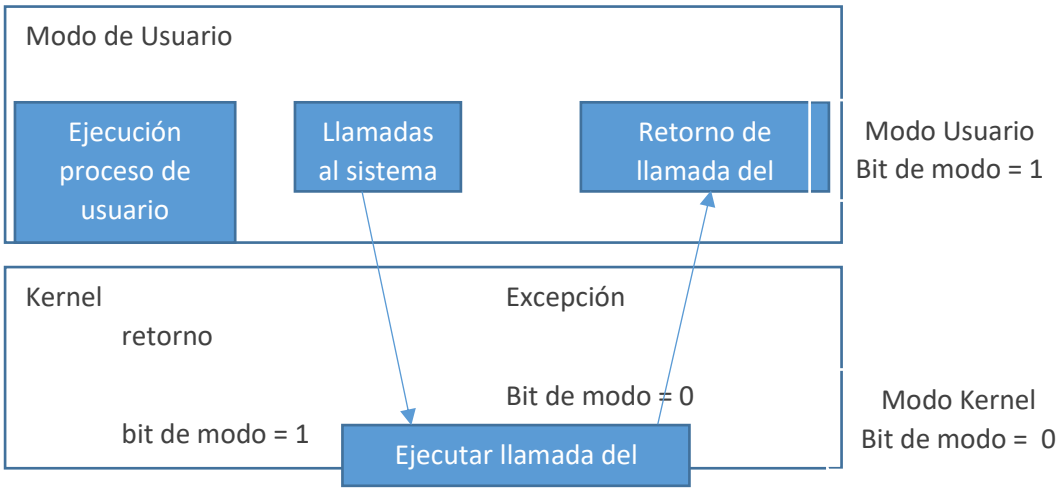

Figura 6Mecanismos de atención a eventos proporcionados por el procesador
Fuente:(Sylberschatz & Galvin, 2005)

El estudio, investigación y análisis de varias metodologías de detección de malware, permiten conocer la estructura, funcionamiento del código malicioso, para mediante una valoración recomendar cual es la más apropiada, con la finalidad de prevenir o mitigar ataques cibernéticos oportunamente, será el éxito de este proyecto, que se orienta al conocimiento de las mismas para la recomendación de cuál es la mejor, con la finalidad de prevenir infecciones de Malware, considerando una o la combinación de todas ellas.

Los resultados se obtienen de procesos autónomos que facilitan la toma de decisiones por el gerente de CiberSeguridad (CISO: Chief Information Security Officer) para contar con sistemas más seguros, confiables y eficientes.

Se ha descrito de manera general algunos conceptos, así como también varios ensayos que son necesarios considerar para desarrollar este proyecto

Conocer el campo de acción de la ciberseguridad, identificar la naturaleza de los ataques, generar conciencia de los peligros a los que se expone (vulnerabilidades), métodos de mitigación a través de diversos procesos para la detección de código malicioso.

2.3. Metodología por Redes Bayesianas

En la actualidad y durante la pandemia de Covid19, debido al uso del Internet y teletrabajo que han crecido significativamente, los ataques cibernéticos también lo han hecho, es así que según (ESET, 2021) los ataques masivos, desde el 2019 disminuyó y más bien se orientaron a ataques dirigidos a las grandes empresas, sin embargo, Ransomware "es un malware que cifra ciertos archivos o bien todo el disco duro de la víctima, bloqueándolo para impedir que el usuario acceda a sus ficheros y solicitando un rescate para recuperar el acceso al sistema y los ficheros"(INCIBE, 2017) y sigue siendo uno de los malware que se mantienen a pesar de haber disminuido en un 2%, como lo demuestra la Figura 7,

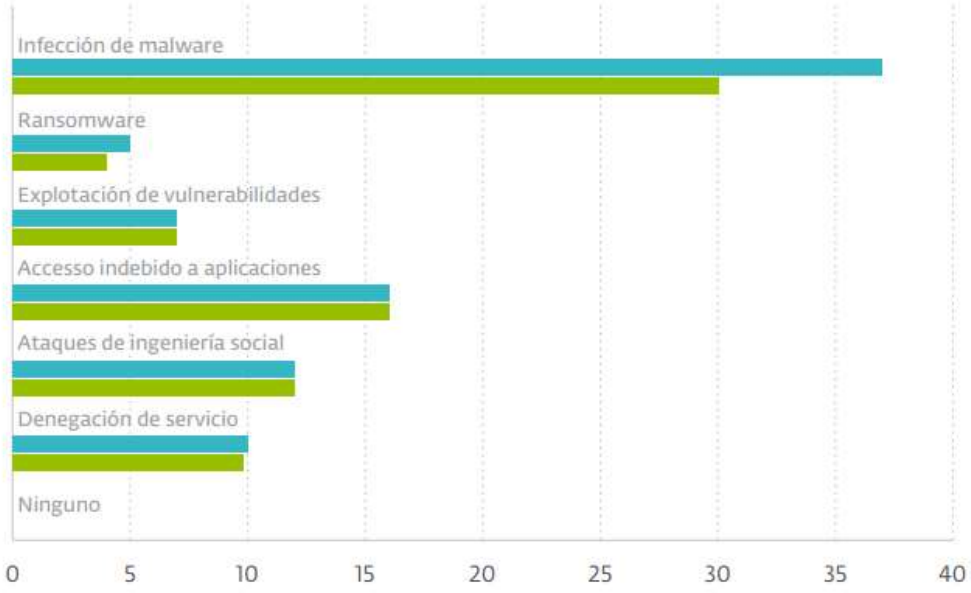

Figura 7Incidentes reportados por las empresas 2019, 2020
Fuente:(ESET, 2021)

Cómo se puede observar en la Figura 7 los ataques reportados por las empresas en mayor porcentaje son de Malware, por eso, es importante monitorear, detectar, identificar y realizar un análisis preliminar para evitar la propagación e infección a otros sistemas o archivos y directorios (Análisis de Malware Estático y Análisis de Malware Dinámico - 2021 - SOFTWARE, 2021).

En este contexto de análisis de seguridad, es importante el modelado gráfico para vincular vulnerabilidades y los posibles ataques en sus diversas etapas, esto se logra

mediante relaciones lógicas de causalidad con diversos órdenes y ajustes de configuración, las mismas que son Determinísticas. Los modelos capturan la incertidumbre casi en tiempo real, la intromisión, para conocer que está sucediendo, cuál es el alcance y el nivel de riesgo, sus posibles causas, así como también, las posibles contramedidas, el tipo y el uso de Redes Bayesianas por la capacidad que tienen, de no ser sensibles a la perturbación de parámetros.(Peng Xie∗ et al., s. f.)

2.3.1. Redes Bayesianas

Las redes bayesianas basan su concepto en la de Probabilidad, centrándose en que un evento ocurra cuando depende de otros factores (incertidumbre) y sobre éstos, muchos científicos entre ellos matemáticos, filósofos y psicólogos lo interpretan. Un típico ejemplo es la predicción del clima, si llueve o no, dependería de la estación climática por la que se esté atravesando, la vegetación del lugar (árboles), los vientos, nivel de altura y la ubicación en el planeta. El tratamiento bayesiano se basa en Probabilidades Condicionadas (PC), es decir que, si se cumple varias probabilidades de sucesos previos, él de estudio, puede también darse.

Los axiomas en la que se base la Probabilidad Condicionada son: a) la probabilidad de un suceso es mayor o igual a cero b) la probabilidad de un suceso seguro es igual a Uno y c) la probabilidad de dos sucesos incompatibles es igual a la suma de las probabilidades de cada uno de ellos. d) Si A y B son dos sucesos, entonces la Probabilidad del suceso A, supuesto que se cumpla B (Probabilidad de A condicionado a que ocurra B) quedaría así:

$$P\left(\frac{A}{B}\right) = \frac{P(A\ y\ B)}{P(B)}$$

Considerando que el teorema de Bayes es una fórmula que permite obtener esta Probabilidad Condicionada a partir de valores de otras probabilidades (Tubau & Cánovas, 2002).

El teorema de Bayes nos permite ver cómo las probabilidades previas pueden modificar el valor de la probabilidad de una determinada hipótesis.

"La fórmula de Bayes es un modelo normativo con el que podemos comparar los juicios de probabilidad emitidos por las personas, entonces cuando se tiende a emitir una estimación de probabilidad en una situación similar a la anterior", hablamos de **Inferencia Bayesiana** (Tubau & Cánovas, 2002).

Las Redes Bayesianas son una representación gráfica de dependencias para razonamiento probabilístico, conformada por nodos que representan variables aleatorias y unidos por arcos que representan relaciones de dependencia directa entre ellas.

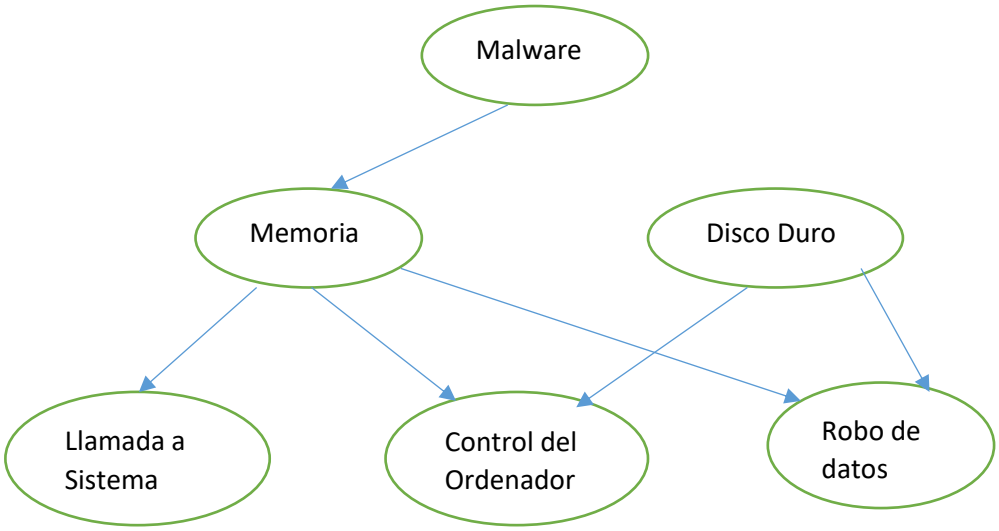

Figura 8Grafo de una red bayesiana
Fuente:(Sucar, s. f.)

En la Figura 8 se observa las dependencias probabilísticas y las interdependencias condicionales de una variable dada a otra. Los nodos Memoria (MEM) y Disco Duro (DD) representan dispositivos del ordenador. La variable a la que apunta un arco es dependiente de la que está al origen de este. Control de Ordenador (CO) depende de Memoria y Disco Duro.

Se completa la Red Bayesiana con los nodos: Llamada al sistema (LL), Robo de datos (RD), Malware (MA).

El nodo MEM separa al nodo LL del resto de variables.

Independencia en el grafo, LL independiente de DD dado MM.

Independencia en la distribución: P{LL|MA,MEM,DD,CO, RD}=P{LL |MEM}

Interdependencia en la distribución: P(MEM/LL,CO)=P(MEM/CO)

Interdependencia en el grafo: I<MA/LL/MEM>

Nodos de secuencia: MA→MEM→LL

Nodos divergentes: LL←MEM→CO,

CO←DD→RD

Nodos convergentes: X→Y←Z, no tenemos

Un nodo es independiente si en la red existen nodos no dependientes

DD es independiente de LL y MA,

MA es independiente de DD

El contorno se denomina a los Padres de cada variable y la estructura de la red bayesiana del ejemplo es la siguiente:

1. Pa (Malware) = ∅
2. Pa (Memoria) = Malware
3. Pa (Disco Duro) = ∅
4. Pa (Llamada al Sistema) = Memoria
5. Pa (Control del Ordenador) = Memoria y Disco Duro
6. Pa (Robo de datos) = Memoria y Disco Duro

Dados los Padres de una Red Bayesiana (variables), como las probabilidades condicionales de cada una de ellas:

- Nodos raíz: vector de probabilidades.
- Otros nodos: matriz de probabilidades condicionales

Para calcular la probabilidad conjunta, se parte de la comprobación de ellas, mediante la aplicación de la regla de la cadena e independencias condicionales.

La probabilidad conjunta se obtiene por el producto de las probabilidades de cada variable dados sus padres: (Sucar, s. f.)

$$P(X_1, X2, X_3 \dots X_n) = \prod_{i=1}^{n} P(\frac{X_i}{P_a(X_i)})$$

En el caso de que n hipótesis $H_1,\ H_2, .., H_n,$ constituyesen una partición, la fórmula de Bayes también se podría escribirse de la siguiente manera:

$$\mathrm{P}(^{H_1}/_{D}) = \frac{P(H_1)P(^{D}/_{H_1})}{P(H_1)P\left(^{D}/_{H_1}\right)+P(H_2)P\left(^{D}/_{H_2}\right)+\cdots+P(H_n)P(^{D}/_{H_n})}$$

Ejemplo:

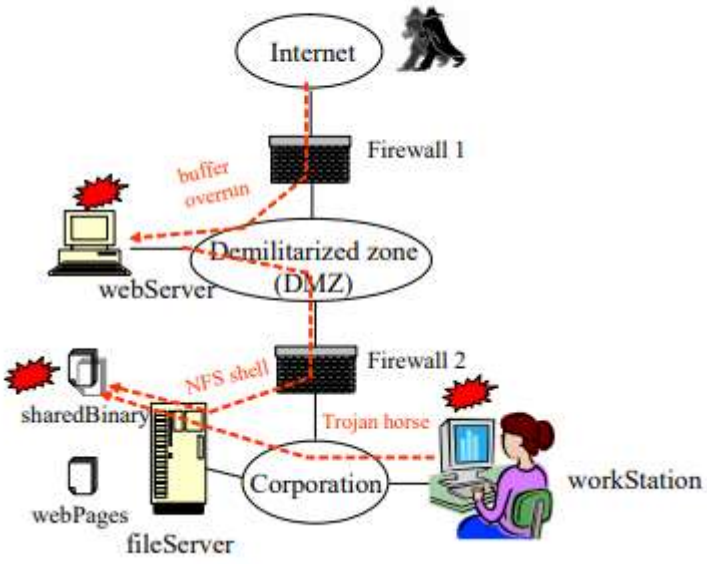

Figura 9Grafo de una red bayesiana
Fuente:(Peng Xie∗ et al., s. f.)

En la Figura 9, se aprecia una infraestructura corporativa, protegida en primer lugar con un Firewall para la salida a Internet, alimentado con un Servidor WEB, el cual se conecta a una DMZ (zona Desmilitarizada o segura) y se tiene un segundo Firewall, el cual protege a la corporación (Intranet) y al servidor de Archivos.

Se aprecia un Atacante y de acuerdo a la ruta especificada, línea punteada de color rojo, su primer objetivo es el Web Server de manera remota, aprovechando la vulnerabilidad Cve-2002-0392 (denegación de servicios de manera remota) y probablemente ejecute código arbitrario, mediante una petición html para obtener acceso al servidor. El Web Server accede a FileServer por medio NFS (Network File Server) el cual "habilita la escritura asincrónica en el servidor, mejorando el proceso de autorización de acceso a archivos"(Oracle, 2011). Una vez conseguido el acceso, se podrá instalar el malware en los binarios ejecutables en las estaciones de trabajo.

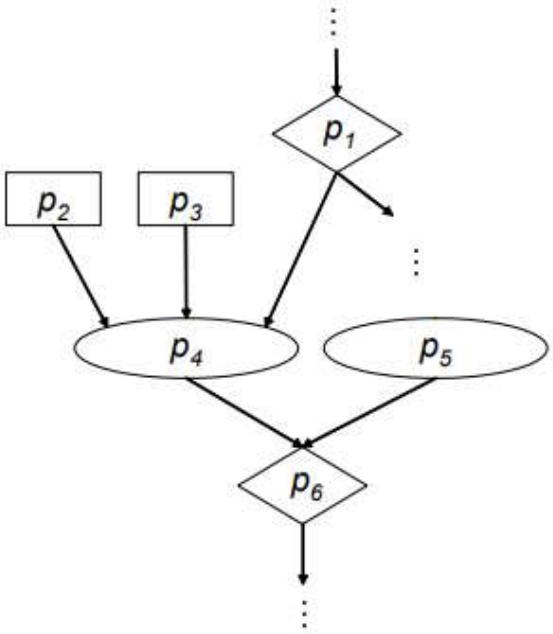

Figura 10Grafo del ataque de ejemplo
Fuente:(Peng Xie∗ et al., s. f.)

En la Figura 10, los nodos P_1, P_2, P_3 son padres de P_4, y este conjuntamente con P_5 son padres de P_6 y expresan la relación de causalidad en el ataque NFS Shell. Si un atacante compromete el servidor web (P_1), este podrá acceder al File Server mediante el protocolo NFS (P_2), y este enviará una partición al Web Server (P_3), y una vez alcanzado el intruso realizará el ataque NFS Shell para controlar los archivos del File Server (P_4). El cambio del archivo es evidente, así que, el Verificador de integridad comunicará las modificaciones, aún sin tener

la certeza, por lo tanto, se crea incertidumbre sobre las acciones del atacante. Cuando se lleva a cabo un ataque, el tiempo debe ser el menor posible, esto ocasionará falsos negativos o falsos positivos (incertidumbre) en el IDS (Intrusion Detection System) instalado. Este gráfico es importante para entender el ataque y preparar procesos de defensa. El modelar la incertidumbre con Redes Bayesianas facilitarán el análisis de seguridad incluyendo relaciones de Causa-Efecto. "La red bayesiana es un gráfico acíclico dirigido DAG, en el que: los nodos representan variables de interés, los enlaces representan la influencia causal entre variables. La influencia está representada por tablas de probabilidad condicional (CPT)" (Peng Xie∗ et al., s. f.).

Para el cálculo de la matriz de probabilidad condicional se utiliza el producto de probabilidades.

$$P(X_1, X2, X_3 \dots X_n) = \prod_{i=1}^{n} P(\frac{X_i}{P_a(X_i)})$$

p_1	p_2	p_3	p_4
MEM	MEM	MEM	0,8
	De lo contrario		0

La Red Bayesiana, es una herramienta poderosa para análisis de seguridad en tiempo real, a partir de construir el modelo de ataque.

El aprendizaje puede ser paramétrico (probabilidades asociadas) o estructural (topología de la red) que debe ser automatizado, porque no es fácil realizar el grafo del ataque. Se pueden utilizar algunas herramientas para ello como Nética, Bynet, Elvira, entre otras (Sucar, s. f.).

La utilidad, se da al generar una heurística para mediante sistemas de Inteligencia Artificial se produzca el aprendizaje y se pueda inferir que estamos frente a un ataque de malware en base a eventos o sucesos anteriores.

2.3.1.1. Algoritmo de Naive Bayes

(Román, 2019) Algoritmos Naive Bayes: Son algoritmos de clasificación de aprendizaje automático, sustentados por la teoría de clasificación estadística Teorema de Bayes, logrando construir modelos de comportamiento, calculando la probabilidad posterior (probabilidad condicionada) a partir de probabilidades de eventos anteriores.

$$P\left(\frac{A}{R}\right) = \frac{P\left(\frac{R}{A}\right)P(A)}{P(R)}$$

$P(A): Probabilidad\ de\ A$

$P\left(\frac{R}{A}\right): Probabilidad\ de\ que\ se\ de\ R\ dado\ A$

$P(R): Probabilidad\ de\ R$

$P\left(\frac{A}{R}\right): Probabilidad\ posterior\ de\ que\ se\ de\ A\ dado\ R$

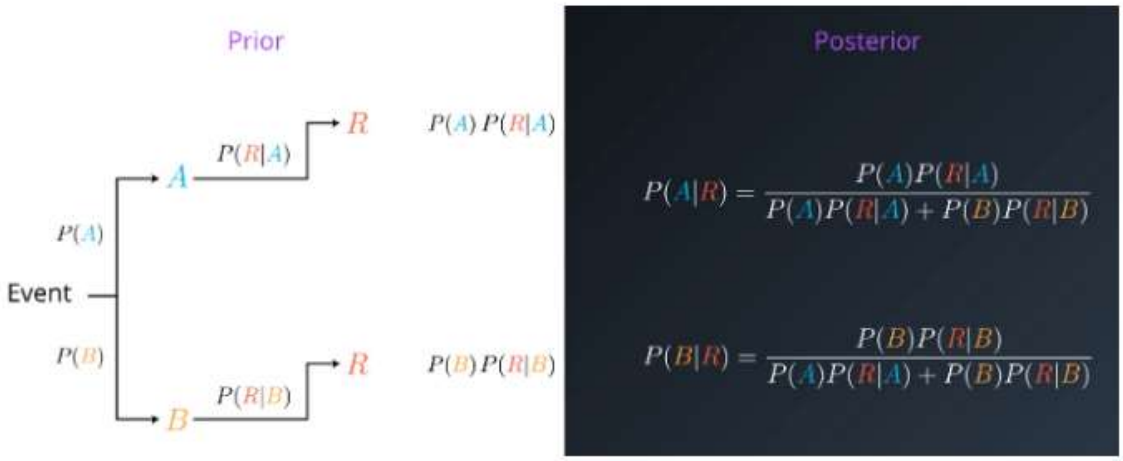

Figura 11Gráfico Previo y Posterior de Naive Bayes
Fuente:(Román, 2019)

Algoritmo Naive Bayes

Este algoritmo consta de 4 etapas que son las siguientes:

1. Convertir los datos en tabla de frecuencias
2. Una tabla de probabilidades de que ocurran los eventos

3. Calcular la probabilidad posterior de cada de clase, mediante Naive Bayes
4. La clase con la probabilidad más alta es la solución de la predicción.

2.3.2. Ingeniería inversa

La ingeniería inversa tiene su origen en los procesos de garantía de productos que fueron devueltos por uso de garantía contra fallas de fábrica. Este proceso se utiliza para conocer cuáles son las razones por las cuales su funcionamiento fue defectuoso. Actualmente, se ha extendido al análisis de malware, para evitar sus efectos dañinos con el objetivo de comprender la estructura y código. Como se podría pensar, este proceso podría tardar mucho tiempo para su análisis, sin embargo, se puede acelerar partiendo de varios análisis ya realizados y a partir de ellos continuar con la detección porque pueden existir variantes y contribuir a prevenir amenazas con soluciones oportunas.(Sanz, 2019)

Existen dos formas de realizarlo: análisis estático y dinámico

2.3.2.1. Análisis Estático

Existen tres lenguajes de programación para crear una aplicación: el lenguaje de alto nivel: donde los programadores realizan los sistemas, como por ejemplo Laravel, PHP, C++, Java y otros. (UNIOVI, s. f.).

Los ordenadores ejecutan código Binario, así que todo programa realizado en lenguaje de alto nivel se transforma en un archivo binario.

Y el lenguaje de Máquina, el cual se ejecuta en el ordenador, siendo complicado realizar una aplicación directamente.

Se menciona este proceso, porque es importante que dentro de un posible Malware se analizará este código binario.

El Análisis Estático consta de los siguientes pasos:

Desmontaje:

- Debido a que malware puede contener código ofuscado (código añadido para hacer inentendible el original, mediante símbolos, pero sin afectar sus funcionalidades) es revisado para identificarlo.
- **Cifrado de datos**, es una de las técnicas de seguridad para evitar la comprensión, por lo tanto, es necesario descifrar los mismos para entenderlo.
- El algoritmo criptográfico y funcionalidad del flujo de control y de datos sirve para identificar las relaciones de las variables, ósea el orden lógico en el que se ejecuta.

Las técnicas más frecuentas para realizarlo son: huellas dactilares de los archivos (hash)(*Análisis de Malware Estático y Análisis de Malware Dinámico - 2021 - SOFTWARE*, 2021)

Para el análisis estático, se empieza con el **desmontado** del binario de malware, pero se debe considerar que el código malicioso pueda estar ofuscado, empaquetado y comprimido, por lo tanto, es necesario eliminar estas capas, desempaquetar de manera manual, y mediante un depurador obtener el código auxiliar del binario en memoria, trasladándose la ejecución al punto de inicio mediante un salto de segmento y obtener un ejecutable descomprimido, eliminando las capas de ofuscación, el empaquetamiento y el desembalaje..

Considerando el peor de los escenarios, se debe considerar que aún exista partes cifradas, en cuyo caso, se debe realizar una referencia cruzada para ver la estructura de datos y el método para **descifrarlos** (algoritmos criptográficos), de esta manera recuperaremos cadenas, comandos C&C (Control y Comunicación) de la información empaquetada y por último se debe considerar las funcionalidades, etiquetándolas por cada llamada al sistema (Call System). El código libre, las referencias cruzadas de los datos y las etiquetas de

funcionalidad, sirven como referencia de la interacción con el sistema operativo (Rahimian, 2014).

Para destacar en este procedimiento es que se debe combinar el análisis estático y dinámico para facilitar la detección y la secuencia de ejecución del malware.

El código malicioso siempre hace llamadas al sistema operativo, para crear procesos y tomar control del ordenador para luego desencadenarse hacia otros computadores con la finalidad de infectarlos.

2.3.2.2. Análisis Dinámico

Este análisis se realiza, ejecutando el código binario, es decir, el Malware, para comprender su estructura, su funcionamiento y los objetivos que se pretenden.

- **Entorno aislado por seguridad**, para proteger el sistema real para establecer puntos de interrupción, controlar el desmontaje, volcado de memoria (reconstrucción de tablas, respaldo de imagen ejecutable
- **Llamadas al Sistema,** para monitorear el tráfico de red, descarga de archivos y protocolos de comunicación (Rahimian, 2014),
- **Basado en clones,** este procedimiento es necesario para detectar variantes de malware acelerando el análisis y reduciendo tiempo; para luego contrastar coincidencias de archivos binarios de malware anteriormente analizados.

El uso de herramientas como SANDBOXING, Debbuging Citadel, RE-Source, Olly Dump, Depurador o analizador de protocoles de red y otros, ayudarán a identificar depuración de la WEB, memoria forense, desempaquetado, descifrado, volcado de memoria, comunicación de red HTTP, comandos de tráfico de red; todos estos procedimientos citados, tienen una función definida para detectar ataques como Man In the

Middle, enlace a archivos DLL (wininet.dll o nspr4.dll), inyección web y otros.

2.4. Revisión Sistemática

La Revisión Sistemática (RS), es un proceso de investigación que genera una síntesis respondiendo a un tema de exploración y pueden ser de dos tipos:

"Cualitativa: aquellas que presentan la evidencia en forma descriptiva y sin análisis estadístico y Cuantitativa/metaanálisis, generan la evidencia de forma descriptiva, pero la gran diferencia está en el uso de técnicas estadísticas"(Aguilera, 2014).

La RS debe seguir las siguientes etapas:

- Formulación del problema,
- Definición de los criterios de inclusión y búsqueda de estudios
- Codificación de las características de los estudios que puedan moderar los resultados,
- Cálculo del tamaño del efecto,
- Técnicas de análisis estadístico e interpretación,

publicación del meta-análisis, (Sánchez, 2010, pp. 53-64)

2.5. Método

La metodología por utilizarse es Documental, porque se basa en los datos que entregan plataformas a través de herramientas como OllyDbg, IDA Pro, Owas, CERT-CC y otros.

"La investigación documental es un proceso basado en la búsqueda, recuperación, análisis, crítica e interpretación de datos secundarios, es decir, los obtenidos y registrados por otros investigadores en fuentes documentales" (Arias, 2012).

A los resultados obtenidos se determinará la Correlación entre ellos para determinar características comunes y determinar patrones (heurísticas).

Se utilizará 3 fases de acuerdo con la recomendación de (Hernández, Fernández, & Baptista, 2010), para la Investigación Documental, cuyas características se resumen a continuación:

- La recolección de los datos se hará de publicaciones realizadas, en este caso de fuentes electrónicas, plataformas y publicaciones como las publicadas por Code Project (Bassov, 2005), Systope (López, 2012), Information Security Technical (Harley, 2009) y otros.
- Se utilizará procesos de análisis, síntesis y deducción de la data.
- Estos procesos se realizarán en base de objetivos específicos con el fin de comparar las metodologías y recomendar la más apropiada para la detección de ciberataques.

Se utiliza dos métodos de investigación:

Comparativo: porque permite encontrar similitudes entre los diferentes datos con la finalidad de diseñar un modelo de comportamiento

Inductivo: este no permite generalizar para crear un escenario que permita evaluarlo y determinar si es uno de riesgo.

Los ataques cibernéticos también obedecen a ellos y a una planificación con objetivos determinados, y por esta razón, son motivo de este estudio.

URL	Filtro	Q1	Q2
https://h-demo100.herokuapp.com/scimagoannual2018Q1	Cyber Scurity	7680	6661

Para esta revisión sistemática se establece tres etapas:

1. Mediante la búsqueda ScimagoJR (se establece las revistas especializadas clasificadas en cuartiles Q1 y Q2, bajo la especialidad “Cyber Security”, Science Computer, para explorar los códigos ISSN (International Standard Serial Number) que “identifican todas las publicaciones periódicas y recursos continuos de toda clase y editada en cualquier soporte, ya sean impresos en papel o en formato digital”(ISSN, s. f.).

2. A partir de los ISSN encontrados en SCImago Q1 y Q2, utilizamos Scopus, encontrando artículos relacionados con el tema propuesto del proyecto, para establecer el criterio de búsqueda, cuya cadena dentro de Scopus, bajo el filtro de los ISSN explorados, para encontrar los artículos que se relacionen con las palabras claves, dentro de cada Abstract. Estas son: Malware (código malicioso), Malware Detection (detección de Malware), Methodologies (Metodologías), Methods (Métodos) o Techniques (Técnicas), Hooking, System Call (Llamadas al Sistema), Reverse Engineering (Ingeniería inversa), Bayesians Networks (Redes Bayesianas).

3. Descargar los artículos investigados a través de los enlaces encontrados en Google Académico envío de correo para solicitar autorización para descargar y utilizar aquellos calificados como OPEN.

El procedimiento consiste en encontrar revistas con relación al tema de estudio, en ScimagoJr, clasificadas por el cuartil Q1, obteniendo 7680 y al buscar por el criterio "Science Computer", de un total de 6661, se redujo a 372.

Al hacerlo, con la premisa de Cybersecurity en el nivel Q1, disminuyó a una, la Journal of CyberSecurity, siendo parte de las encontradas bajo Science Computer y en Q2, 0, como se aprecia en la Tabla 1.

Tabla 1 Filtros de búsqueda de revistas Q1 y Q2 en ScimagoJr

URL	Tópico	Filtro	# Revistas
https://h-demo100.herokuapp.com/scimagoannual2018Q1	Science Computer	Q1	372
https://h-demo100.herokuapp.com/scimagoannual2018Q1	Science Computer	Q2	294
https://h-demo100.herokuapp.com/scimagoannual2018Q1	Cyber Security	Q1	1 Encontrada en Science Computer
https://h-demo100.herokuapp.com/scimagoannual2018Q1	Cyber Security	Q2	0

Fuente: Elaboración propia

Mediante el algebra booleana se establece dicha secuencia con los operadores OR, AND y EXCLUDE. Este último sirve para eliminar aquellas publicaciones que no

pertenece a la naturaleza, como por ejemplo el tipo de documento (sólo Artículos de revista o conferencia, área de aplicación: médica, biológica, ambiental, física, año de publicación (desde el 2016 hasta la fecha)).

La cadena encontrada es:

(ISSN ("10848045" OR "1556181X " OR " 15561801 " OR " 09603174 " OR " 15731375 " OR " 09252312 " OR " 00200255 " OR " 09505849 " OR " 13845810 " OR " 1573756X " OR " 15740137 " OR " 18685145 " OR " 18685137 " OR " 15513203 " OR " 23274662 " OR " 21687161 " OR " 00189340 " OR " 15455971 " OR " 21921962 " OR " 21693536 " OR " 15462218 " OR " 15462226 " OR " 10949224 " OR " 15577406 " OR " 21586578 " OR " 2158656X " OR " 24712566 " OR " 24712574 " OR " 14698153 " OR " 03600300 " OR " 15577341 " OR " 01674048 " OR " 17422876 " OR " 09574174 " OR " 00457906 " OR " 20763417 " OR " 2268046 " OR " 00189162 " OR " 23765992 " OR " 15582256 " OR " 00189219 " OR " 10958592" OR "15740021 " OR "18767354 " OR "18767362 " OR "21916594 " OR "21916586 " OR "08848289 " OR "19474040 " OR "13283154 " OR "18695450 " OR "18695469 " OR "17262135 " OR "16848799 " OR "10001026 " OR "22143173 " OR "1674733X " OR "18691919 " OR "09696016 " OR "14753995 " OR "10762787 " OR "10990526 " OR "01976729 " OR "20423195 " OR "12962074 " OR "18690327 ")) AND (("Malware Detection") AND (comparative OR analysis) AND (methodologies OR methods OR techniques) AND (hooking OR "Bayesians Networks" OR "Reverse Engineering")) AND (EXCLUDE (SRCTYPE , "d") OR EXCLUDE (SRCTYPE , "p")) AND (EXCLUDE (DOCTYPE , "re")) AND (EXCLUDE (SUBJAREA , "MEDI") OR EXCLUDE (SUBJAREA , "CENG") OR EXCLUDE (SUBJAREA , "PHYS")) AND (EXCLUDE (PUBYEAR , 2015) OR EXCLUDE (PUBYEAR , 2014) OR EXCLUDE (PUBYEAR , 2013) OR EXCLUDE (PUBYEAR , 2011)).

De este procedimiento, se hallaron 115 documentos, los cuales se almacenaron con información relevante como son: Título del artículo, Autor o autores, número DOI, fecha de publicación, resumen, nombre de la revista, páginas, entre otros.

La lectura de éstos, son el último filtro para captar documentos relacionados con el objetivo del proyecto (en concordancia a las metodologías materia de investigación:

Hooking, Redes bayesianas e ingeniería inversa) y la realización de una tabla comparativa y cuantitativa, derivada de la revisión sistemática.

2.5.1. Selección de Instrumentos de Investigación

Para lograrlo, se utilizó dos motores de búsquedas de artículos científicos como SCImagoJr y Scopus, con el objetivo de realizar una revisión sistemática de publicaciones, estableciendo varios criterios siendo uno de ellos, el que se encuentren dentro de los Cuartiles Q1 y Q2 y determinando el lapso de las publicaciones (desde el 2016 hasta la fecha) y que se relacionen con el tema de este proyecto de investigación.

Clasificación y análisis de la data recopilada.

2.5.2. Procesamiento de Datos

Para depurar los resultados obtenidos, como se explicó anteriormente, se realizó en SCOPUS, empleando el número ISSN obtenido en SCImagoJr con la cadena construida, se realizan de la siguiente manera:

1. “Computer Science” se encontraron 372 de un total de 7680, esto representa un 4,84%.
2. Cybersecurity para Q1
3. Ingeniería Electrónica
4. Matemática aplicada
5. Ingeniería Artificial

Los datos obtenidos serán comparados para determinar los procesos comunes utilizados al detectarlos y la recomendación de su aplicación para prevenir amenazas e identificar el tipo de ***código malicioso*** para evitar ataques y bloquearlos.

CAPÍTULO III

ANÁLISIS COMPARATIVO DE LAS METODOLOGÍAS: HOOKING, REDES BAYESIANAS E INGENIERÍA INVERSA, PARA LA DETECCIÓN DE MALWARE

3.1.Introducción

El intercambio de información a través de dispositivos electrónicos, mediante la interconexión de redes a Internet, exige ***seguridad*** para preservar la integralidad, Confiabilidad y Disponibilidad de información, como principal acción a implementar en los sistemas de datos e infraestructura de comunicación. Todas las cosas en la naturaleza tienen y obedecen a un proceso; de esto, no puede estar alejado la tecnología, que es un conjunto de ellos, creados por seres humanos, con una lógica, secuencialidad y relacionamiento tendiente a alcanzar el objetivo para el cual fue diseñado (Algirde Pipikaite, & Marc Barrachin, Scott Crawford, 2021)

Con alguna experiencia sobre el avance tecnológico y más aún, en la actualidad con la emergencia sanitaria (COVID 19), los medios de comunicación cambiaron e incrementaron del uso de equipos electrónicos que facilitan el intercambio de información y datos, a través de ellos. Es vital analizar cómo se producen dichos ataques para defendernos o detectarlos antes de que ocurran.

La pandemia ha acelerado la Cuarta Revolución Industrial, donde la digitalización del ser humano y sus actividades como: comercio electrónico, educación en línea, teletrabajo se universalizó. De igual forma el equilibrio emocional, la capacidad adquisitiva y la brecha tecnológica, ahora más acentuada, por la pérdida de trabajo e ingresos. Los jóvenes, una generación que enfrenta desafíos en la calidad de educación, crecimiento económico, salud mental y la proyección más importante para los próximos 5 años: la automatización de 85 millones de trabajos (Kartz, 2021).

En el siguiente cuadro sinóptico se observará el desarrollo de este capítulo.

Cuadro Sinóptico descriptivo de los diversos métodos propuestos en los artículos estudiados y clasificados en las metodologías

Fuente: Elaboración Propia

Al realizar las síntesis de los artículos citados, se ha encontrado diversas propuestas que se identificaron en tres metodologías Hooking, clasificadores por Redes Bayesianas e Ingeniería Inversa.

3.2. Análisis de Malware (Software Malicioso)

Aunque la respuesta es intuitiva a la pregunta; ¿por qué es necesario hacer un análisis de Malware?, ésta va más allá y la importancia se enfoca en construir estructuras de defensa ante un ataque inminente, sustentada en el conocimiento que se obtiene sobre el funcionamiento, procesos, formato y efectos que causa.

Existen muchas razones para realizar este estudio, fundamentado, en la mayoría de los casos cuando se hace público que, instituciones (principalmente financieras) han sido atacadas, que los datos personales se encuentran hackeados y que está en riesgo el patrimonio por el que se ha trabajado, atentando la continuidad operativa de la institución, servicios de la infraestructura crítica del país y la garantía de que éstos estén seguros y protegidos.

Lamentablemente, no es tan simple, pues debemos confiar en la asertividad y ética de las empresas, tanto públicas y privadas, cuya responsabilidad es proteger los datos necesarios para el sostenimiento de procesos, como país y claro también de sus habitantes.

La estructura de malware según (Pandey, K; Mehtre, M, 2014), tiene procesos que se autoejecutan para evitar ser detectados, mediante técnicas, tales como: de ocultación, ofuscación, cifrado , polimórficas y metamórficas. Ampliando las definiciones, la estructura de malware consta de Propagación, Activación y Defensa. La primera acción es la de introducirse al equipo o dispositivo de la víctima (por Ingeniería Social, Correo electrónico, navegador), luego la etapa de Propagación (instalación por desconocimiento de los usuarios) y la etapa de Defensa (la más crítica, porque sirve para evitar ser detectado, cifrado, ofuscamiento, empaquetamiento y otros). Los efectos de malware dependen de la intencionalidad del atacante, sin embargo, entre los síntomas tenemos: ”ralentización del dispositivo (porque dependen de la capacidad de procesamiento), falta de espacio de almacenamiento, despliegue de ventanas

emergentes no deseadas, ransomware (bloqueo o negación de acceso al equipo y archivos, hasta que se pague un rescate)" (Belcic, 2021)

3.3. Metodologías de Detección de Malware

Desde el inicio de los sistemas de automatización de datos (década de los 60s) el sistema operativo se definió como un "Administrador de Hardware" (Deitel, 1993), sin embargo, en la actualidad realizan muchas operaciones como interfaz con el usuario, brindar acceso a datos, comunicación por redes de computadores, es así que un Sistema Operativo. proporciona una serie de llamadas al Sistema para cada requerimiento por el uso CPU, las cuales, ordenan ejecutar las tareas específicas requeridas por el programa o aplicaciones, por ejemplo, manipulación de archivos, almacenamiento en discos duros, directorios, reserva, extensión y liberación de espacios de memoria.

3.3.1. Llamadas al sistema (SystemCall)

Una llamada al Sistema es un procedimiento que, mediante pasos, sirve para solicitar un servicio del Sistema Operativo, ejemplo: usar Time: para la fecha y hora del sistema, Write: para escribir un dato en un dispositivo de salida (pantalla o un disco magnético), Read: lectura de un dispositivo de entrada, (teclado o un disco magnético), Open, obtener un descriptor de un fichero del sistema, para pasarse a Write. Estos métodos son instrucciones en lenguaje ensamblador y que se obtienen a partir de un lenguaje de alto nivel C++, Angular, Laravel, JAVA, etc. Mediante subrutinas o funciones se genera rutinas en tiempo de ejecución y dependiendo de la función requerida, existen 5 categorías: control de proceso, manejo de dispositivos, mantenimiento de información y comunicación.(Cruz, 2013).

"**Control de Proceso** se orienta a: cargar, ejecutar, abortar, terminar, crear procesos, establecer y obtener atributos de proceso, administrar el tiempo, asignar y liberar memoria.

Administrador de archivos tiene por funciones: Crear, borrar s, Abrir y cerrar archivo, establecer y obtener atributos de archivo, leer, escribir y reposicionar,

Administración de dispositivos: Solicitar y liberar dispositivos, Leer, escribir y reposicionar, establecer y obtener atributos de dispositivos, conectar y desconectar dispositivos lógicamente

Mantenimiento de información: Obtener y establecer hora o fecha, obtener y establecer datos del sistema, obtener y establecer atributos de proceso, archivos o dispositivo

Comunicaciones: "Crear, borrar conexión de comunicación, enviar, recibir mensajes, transferir información de estado, conectar o desconectar dispositivos remotos" (Cruz, 2013).

3.3.2. Revisión Sistemática

A partir de los artículos encontrados, se realiza la síntesis de los escogidos para determinar procedimientos internos de cada método propuesto por los autores que permiten identificar cuáles son comunes a todos.

3.3.2.1. Metodología de defensa por HOOKING

Hooking, es la técnica de modificación o alteración de funciones del sistema mediante la inyección de código, sobre todo a archivos DLLs, Se utiliza "para la depuración o extensión de las funcionalidades, interceptar mensajes de eventos de teclado o mouse antes de llegar a una aplicación o interceptar llamadas al sistema operativo con el fin de monitorizar el comportamiento" (Ferrer, 2011)

En Windows, las llamadas al sistema se llaman API (Application Programming Interface) y son funciones dinámicas llamadas dll (dinamic link library) que permiten el funcionamiento de una aplicación.

Existen varias categorías de API de acuerdo a sus funciones (López, 2012), tales como:

- Depuración y manejo de errores
- Dispositivos E/S
- Varias DLL, procesos e hilos
- Comunicación entre procesos
- Manejo de la memoria
- Monitoreo del desempeño
- Manejo de energía
- Almacenamiento
- Información del sistema
- GDI (interfaz para dispositivos gráficos)
- Interfaz de usuario

Se deduce que, cualquier aplicación siempre hará una llamada para la ejecución de sus procesos (Call System), el malware no es diferente a una aplicación y para alcanzar sus objetivos y tomar control del ordenador deberá hacerlo. Hay software malicioso que incluye en su estructura, lenguaje ensamblador para pasar inadvertido, pero a pesar de ello, tiene que invocar a una de ellas.

En Windows, no se utiliza código administrado (CLR, Common Language Runtime, código administrado en tiempo de ejecución) para una API, es decir, el "CLR toma el código administrado, lo compila en código de máquina y luego lo ejecuta (Microsoft, 2016).

La interoperabilidad de las API con el framework (.Net), se obtiene mediante el inicio del servicio que llama a las funciones no administradas implementadas en archivos DLL.

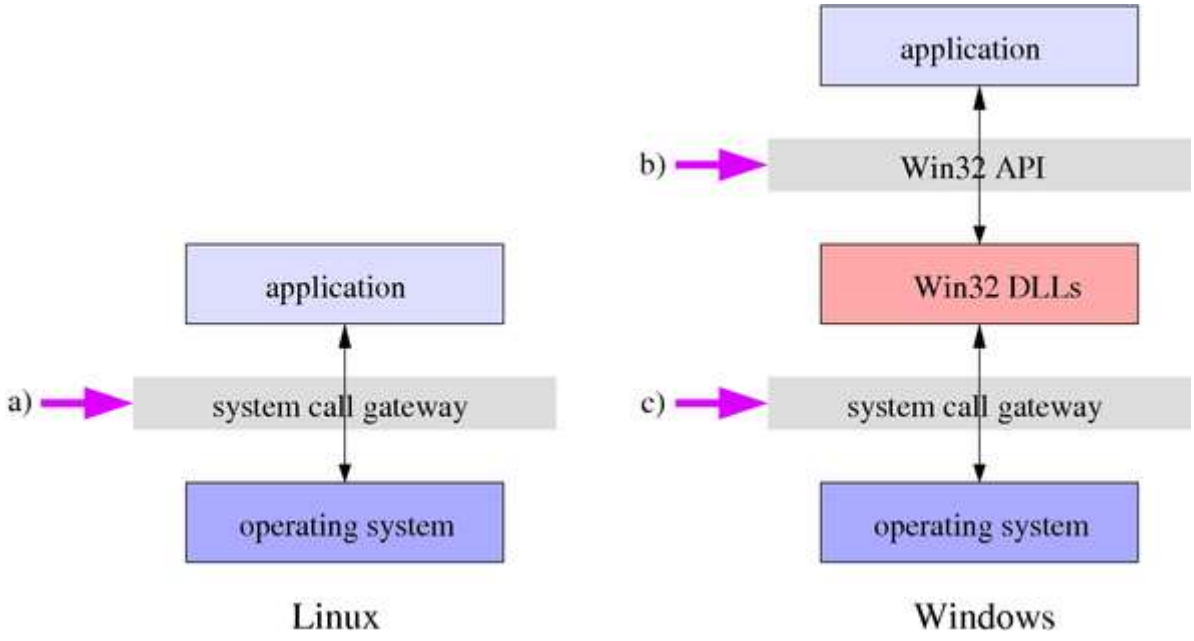

Figura 12Interoperabilidad de API de Windows
Fuente: http://systope.blogspot.com/(López, 2012)

En la Figura 12 se observa el proceso de la llamada al Sistema a partir del código no administrado y la ejecución de los archivos DLL, sin embargo, es recomendable utilizar código administrado en el framework para ejecutarlas.

Se analiza el archivo ejecutable iexplore.exe mediante la herramienta OllyDbg

OllyDbg es un "depurador a nivel de aplicación y su interfaz muestra código ensamblador, volcado hexadecimal, la pila y registros de la CPU, rastreo, puntos de interrupción condicionales, visión de cabecera PE, edición hexadecimal, plug-in de soporte, configuración de *breakpoints,* creación de *threads*, puntos de interrupciones de hardware, software memoria" (Hard2bit Dr., 2013).

Una API de Windows es analizada en el laboratorio de ESET, mediante técnicas de Hooking, de una muestra de malware que inyecta código malicioso a todo proceso que utiliza User32.dll, a través de la llave de registro AppInit.dlls. El código se encuentra empaquetado con

uPolyX y la descomprensión se efectúa con una llamada a menset y varios memcpy (www.welivesecurity.com, 2014).

El uso de herramientas para monitorear y detectar código ofuscado de este caso se observa en el anexo 1.

Este ejemplo, nos permite dimensionar que tan peligroso es el alcance que tiene un malware cuando se introduce código aleatorio en un navegador. A partir de esto se puede robar data, inclusive antes de que sea cifrada.

3.3.2.2. Conexión de API Nativa y Control de Creación de Procesos

La acción de todo malware se basa en la creación de procesos CreateProcess(), ya sea desde modo Kernel o modo Usuario, sin embargo, Kernel siempre deberá crear procesos en Modo Usuario, razón por lo cual la solución se enfoca en Modo Usuario, funciones de Shell, creación manual de procesos como secuencia de llamadas a API nativas (Bassov, 2005).

Una llamada para generar un proceso tiene los siguientes pasos:

1. El archivo ejecutable debe abrirse para acceder FILE_EXECUTE
2. La imagen ejecutable debe cargarse en la RAM.
3. Establecer Proceso de objetos Ejecutivo (estructuras EPROCESS, KPROCESS y PEB)
4. Se debe asignar un espacio de direcciones para el proceso recién creado.
5. Enhebrar Executive Object para el hilo principal del proceso (estructuras ETHREAD, KTHREAD y TEB) tiene que ser establecido.
6. Se debe asignar la pila para el subproceso principal.
7. Debe configurarse el contexto de ejecución para el hilo principal del proceso.
8. El subsistema Win32 debe estar informado sobre el nuevo proceso.

Ningún código que quiera generar un nuevo proceso podrá omitir la ejecución de los pasos citados anteriormente, por eso, para la detección de malware es necesario monitorearlos, controlarlos y bloquearlos.

Lo importante es la identificación de este proceso como nuevo, luego de comparar con los que ya se encuentran registrados y autorizados, para denegar el servicio a iniciarse.

Es decir, muchas soluciones de seguridad monitorean y bloquean las solicitudes de API conectando funciones de una tabla de funciones de servicio llamada SSDT para procesar las solicitudes de API existentes en el Kernel (S8).(Method for defending against dll injection without hooking, 2011).

La solución para detectar malware por inyección de código a las DLLs, es el Monitorear y Controlar la creación de procesos y subprocesos considerando los hilos de rutinas remotos y que las diversas metodologías también las generarán.

3.3.2.3. Método de Concordance:

(Murthy et al., 2019, pp. 4991-4997) en su artículo expone que, existen 3 formas de análisis de malware que son: Estático (sin ejecución de Malware), Dinámico (ambiente aislado y seguro, virtual o sandbox (ejecución de prueba de programas no verificados probablemente Malware) e Hibrido (estático y dinámico) y opta por el dinámico, mediante Concordance que, es una lista de palabras o patrones claves (KWIC, Key Word in Context) en un texto a ser buscados en aquellas que están en el entorno. Existen varios tipos de API, una de ellas es la WinAPI (con sufijo W, acepta cadenas ASCII y con sufijo Ex, función actualizada). Otro tipo son NTAPI y estás son las más usadas por los desarrolladores de malware porque posee funcionalidades más reservadas que WinAPI. Luego, para encontrar la mayor cantidad de APIs, se emplea Document Frecuency (DF).

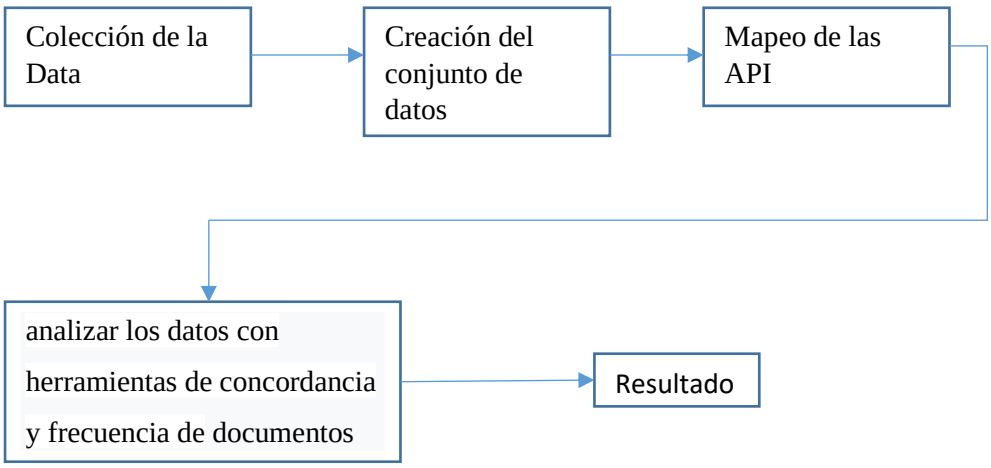

Figura 13Metodología del proceso
Fuente: (Murthy et al., 2019, pp. 4991-4997)

Destacamos los comportamientos maliciosos y la secuencia de llamadas a la API en el Dataset en la siguiente tabla.

Tabla 2 Actividad maliciosa vs Secuencia de llamadas API

Actividad Maliciosa	API
Screen Capture	(GetDC, GetWindowDC), CreateCompatibleDC, CreateCompatibleBitmap, SelectObject, BitBlt, WriteFile
Hooking	SetWindowsHookA, CallNextHookEx
Downloader	URLDownloadToFile, (WinExec, ShellExecute)
Enumerate all process	CreateToolhelp32Snapshot, Process32First, Process32Next
Anti debugging	(IsDebuggerPresent, CheckRemoteDebuggerPresent, OutputDebugStringA, OutputDebugStringW)
Synchronization	CreateMutexA, CreateSemaphoreW
Key Logger	(FindWindowA, ShowWindow, GetAsyncKeyState) (SetWindowsHookEx, RegisterHotKey, GetMessage, UnhookWindowsHookEx)
Dropper	FindResource, LoadResource, SizeOfResource

Fuente:(Murthy et al., 2019)

El resultado obtenido se aprecia en siguiente tabla:

Tabla 3 Comportamiento malicioso y secuencia de llamadas a la API

Comportamiento malicioso	Secuencia de llamadas a la API
Modificar el atributo de archivo	GetFileAttribute, SetFileAttribute
Modificar la hora del archivo	GetFileTime, SetFileTime
Registro de carga	RegCreateKey, RegSetValue, RegCloseKey
Enumere todo el proceso	CreateToolhelp32Snapshot, Process32First, Process32Next, WTSEnumerateProcesses
Escalada de privilegios	OpenProcessToken, LookupPrivilegeValueA, AdjustTokenPrivileges
Terminar proceso	TerminarProcess

,

Fuente:(Murthy et al., 2019, p. 4994)

Este método permite la comprensión del comportamiento, ya sea en los cambios de memoria, valores de variables y configuraciones, que exponen las características de malware y la secuencia de llamadas API con un alto nivel de detección de malware.

3.3.2.4. Análisis de indicadores y secuencia de llamadas de WinAPI

(Sharma & Kant, 2018, pp. 201-211) en el estudio “Early detection of ransomware by indicator analysis and WinAPI call sequence pattern” afirma que la proliferación de ataques sufridos por ransomware determina una alta incidencia proponiendo el método de análisis de indicadores que son los siguientes:

Tipo de archivo: para evitar su detección, Ransomware cifra los archivos del host, cambiando la extensión de en el tipo de archivo mediante AnalyzeIt (“tipo de archivo real, función de programa y/o extensión,

clasificación del archivo, Tipo MIME, caracteres específicos, ID del programa, información de ejecutables: ImageBase, EntryPoint, CheckSum, Tabla de importación y exportación, Directorios y secciones PE con información específica: nombre, dirección virtual relativa a RVA, tamaño virtual, compensación de archivo RAW, tamaño RAW, características, cuál fue el empaquetador / cifrador / compresor / compilador, que procesó el ejecutable y la forma hexadecimal", (*Shocking Soft*, s. f.)),

Similitud Hexadecimal: la encriptación implica una transformación de los datos a un conjunto indefinido de ASCII y por medio la herramienta WinMerge, compara las cadenas presentes con su hexadecimal y mediante la diferencia se miden los archivos.

Monitoreo de eliminación: una vez que se adquiere los archivos del host, Ransonware copia la forma cifrada y elimina el original, lo que implica una gran actividad en el sistema que, es fácilmente detectable por monitoreo,

Conversión de varios a uno. La probabilidad de Software de Benin es menor frente a una aplicación de ransonwere. Se utiliza IDA Pro para entender el código fuente al código de máquina ejecutable del ensamblador Se emplea las Bibliotecas de Microsoft Detours para la extracción de la secuencia de llamadas API de Windows conectadas con las API basadas en Windows, comprendiendo la actividad en segundo plano de Ransomware.

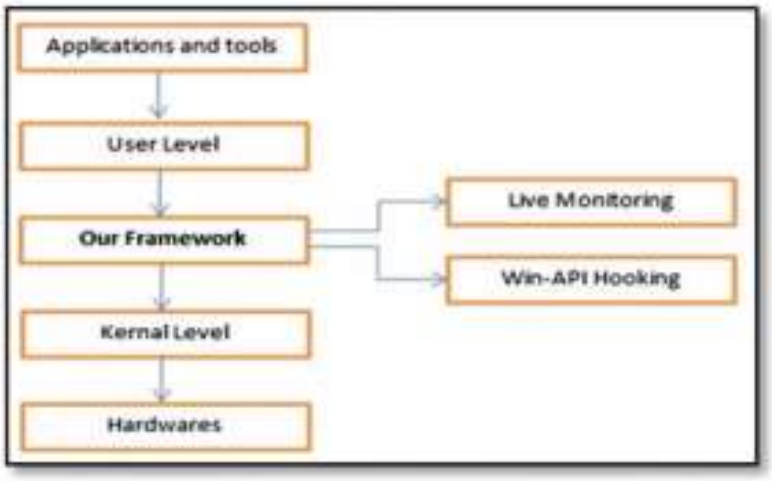

Figura 14Arquitectura propuesta
Fuente:(Sharma & Kant, 2018, pp. 201-211)

La detección de la palabra, texto clave, archivo o aplicación, contenían valores hexadecimales que actuaron como un diccionario y se etiquetaron con bandera roja. Luego se compara con las firmas de las secuencias de llamadas API con las extraídas mediante n-gramas para identificar las coincidencias como se observa en la siguiente figura.

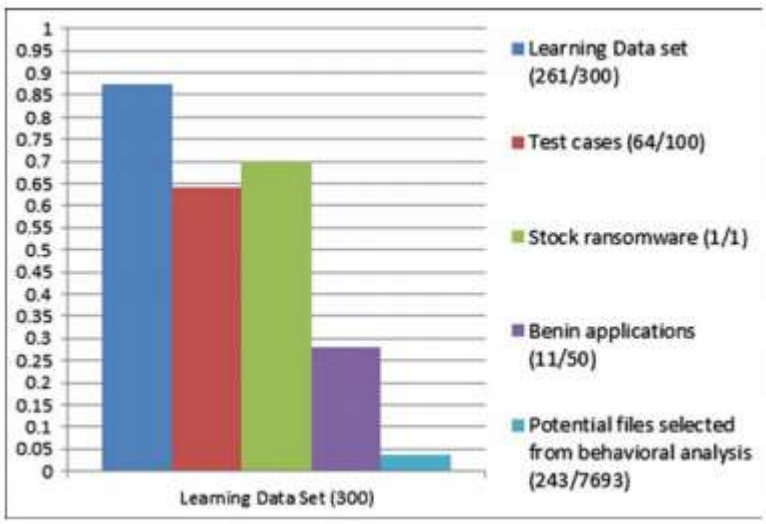

Figura 15Probabilidad de coincidencia de firmas de secuencias de llamadas Win API en todos los casos Fuente:(Sharma & Kant, 2018)

3.3.2.5. Ejecución de monitores de referencia en la interface

En su publicación (Srivastava, 2008), afirma que malware puede evadir ser detectados por herramientas antimalware, sin embargo, el procedimiento de cualquier aplicación para interactuar en un computador con el SO implica la creación de procesos, uso de espacios de memoria, o espacio de almacenamiento en disco, en busca de servicios o recursos, mediante las llamadas al SO, en modo Kernel o Modo Usuario. Estás llamadas al SO, cuando provienen de un malware redirigen interrupciones y ocultan procesos, alterando sintaxis de código, pero manteniendo la semántica (funciones) de los módulos. La **ejecución de monitores de referencia en la interface** de cada llamada al sistema revelaría el comportamiento inusual.

Markhor, propuesto por (Lajevardi, AM et al., 2021, pp. 123-133) que analiza secuencias de dependencias de control de llamadas del sistema para crear un conjunto de patrones maliciosos en base a un algoritmo difuso determinando la naturaleza mientras se ejecuta el malware.

"El comportamiento de malware se detecta en función de los recursos del sistema Windows que se requieren" (Lajevardi, AM et al., 2021, pp. 123-133) y se clasifican en cinco clases basados en:

- Procesos,
- Red,
- Funcionamiento de red,
- Alteración del sistema,
- API

El enfoque Markhor se evalúa en cuatro etapas:

Datos de prueba (malware y software benigno),

Extracción de características del comportamiento de malformaciones de software

Captura de patrones útiles de los datos de prueba mediante secuencias de dependencias de llamadas al sistema

La detección de malware se realiza comparando las reglas maliciosas extraídas (almacenadas en la base de datos) con los archivos sospechosos. El puntaje final, es la suma de la secuencia de dependencia de llamadas al sistema.

3.3.2.8. HAL, la solución para la detección de malware en Android,

según (Thien-Phuc et al., 2020, pp. 1-9) tiene dos etapas:

Análisis, utilizando herramientas que proporcionan un conjunto de características que servirán de data para la segunda parte. Ejemplos de estos comportamientos dañinos, son la manipulación de recursos del dispositivo (Android) como cámara, teléfono, mensajes para extraer información, enviar

SMS a distintos números, sin la conciencia del propietario. Otra posibilidad es tomar control del equipo, mediante escalación de privilegios, constituyéndose en una infiltración total, siendo necesario identificarlos mediante la construcción de diagrama de flujo en base a llamadas API, para deducir cuáles son los recursos o servicios a ser alterados. El análisis dinámico es complementario porque identifica código ofuscados o cifrado y se realiza por medio de técnicas de Hooking (enganche) a todas las aplicaciones API usadas, sin embargo, son identificadas porque intervienen directamente con la memoria usada por el proceso.

Detección, se la realizará por medio de Hardware Abstract Layer (HAL), que es una interfaz que permite implementar funcionalidades en Hardware sin modificar el nivel superior o inferior del sistema y se aplica en versiones de Android 8.0 o superiores.

La modificación de HAL es conveniente para explorar manipulación de recursos porque todas las peticiones al Hardware recaen en HAL ampliando el nivel de alcance. Además, Hal es independiente del controlador de Hardware y el módulo de monitoreo no se puede deshabilitar, porque no se ejecuta como un servicio.

Para mejorar la técnica de Hooking, se utiliza HALWatcher para monitorear la manipulación de recursos en Android Open Source Project, debido a que no necesita rootearse y su arquitectura permite desarrollar aplicaciones para rastrear comportamientos de malware y detectar ataques de manipulación de recursos. Al funcionar como parte del sistema Android, puede monitorear todos los paquetes instalados de Play Store. El modelo genera información de manipulación de recursos, cada vez que uno de ellos es solicitado por el hardware, ampliando la eficiencia de detección.

3.3.2.9. **Runtime Detection Framework for Android Malware**,

propuesto por (Kim et al., 2018, pp. 1-15) para la detección de Malware de Android mediante la reescritura de aplicaciones. El diseño del modelo es cliente -servidor basado en un árbol de sufijos y con este uno probabilístico para distinguir entre código benigno y malware. El modelo posee un algoritmo de puntuación para medir niveles dañinos de las API llamadas para mejorar la precisión de detección.

El framework tiene dos componentes: el lado del servidor (sistema de auditoría) y el lado del cliente (agente seguro).

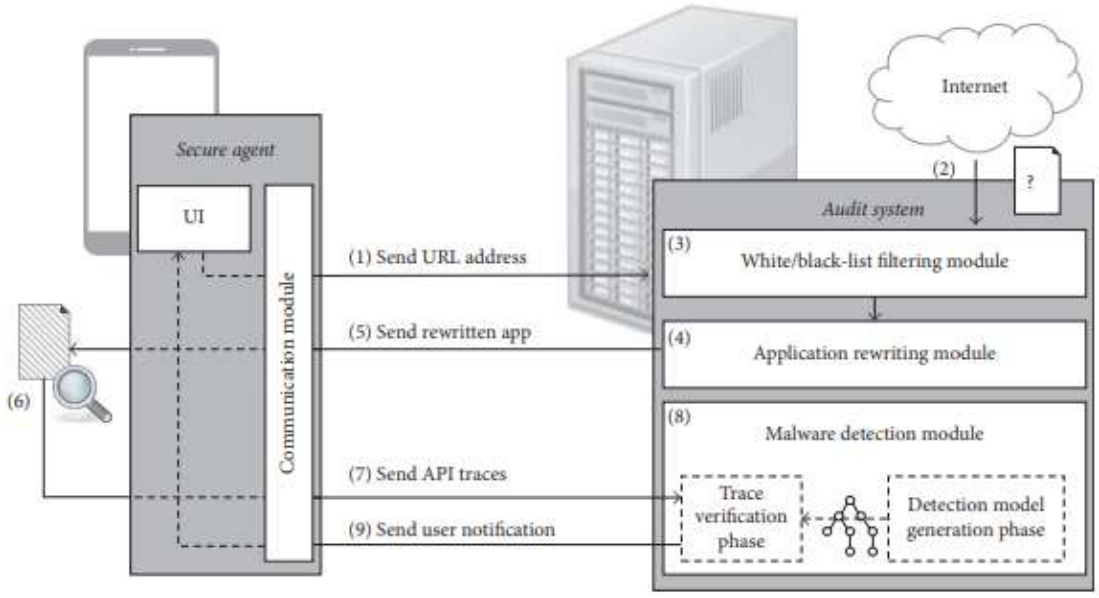

Figura 16Arquitectura general del framewok de detección de malware
Fuente: (Kim et al., 2018, pp. 1-15)

Como se observa en la Figura N°24, se observa los dos componentes citados: Agente seguro y el sistema de auditoría, cuando se requiere descargar una aplicación se envía el URL (Uniform Resource Locator) al sistema de auditoría, descarga la aplicación y verifica si es una aplicación conocida usando las listas blanca y negra. Si pertenece a la primera se transmite y el agente seguro instalará la aplicación, caso contrario se emite una alerta para el agente seguro, y la aplicación se reescribe mediante la inserción de código

de autocontrol para su análisis posterior. La aplicación reescrita se transmite al agente seguro instala la aplicación reescrita y el código de autocontrol, extrae y emite la información sobre la secuencia de llamadas de API al agente seguro y las trazas (algoritmo de árbol de sufijos) al agente de auditoría que, analizará con el modelo de detección para comparar con algún rastro de malware. Si se encuentra alguna coincidencia se notifica al agente seguro y este genera alarmas en el dispositivo. Las fases utilizadas son:

- Filtrado de lista blanca/negra
- Reescritura de aplicaciones
- API peligrosas
- Código de autocontrol:
  - entrega de seguimiento de invocación al agente seguro
  - Identificación de los tipos de API
  - Invocación de API originales
- Módulo de detección de Malware
  - Fase de generación del modelo de detección
  - Examen de trazas
- Ejecuciones automáticas de aplicaciones
- Precisión de detección de Malware
  - Selección de Parámetros
  - Rendimiento de detección de malware
- Gastos generales en tiempo de ejecución y sobrecarga de extremo a extremo
  - Tiempos de:
  - Principio a fin
  - Reescritura de la aplicación
  - Autocontrol
- Aumento de tamaño de las aplicaciones reescritas
- Comparación con otros métodos.

3.3.3. Metodología de detección por Redes Bayesianas

3.3.3.1. Android Malware Detection Via Graph Representation Learning,

ensayo realizado por (Feng et al., 2021, pp. 1-14) y a diferencia de los artículos anteriormente citados, el modelo propuesto se fundamenta en el aprendizaje de representación gráfica a través de la red neural gráfica (GNN), que analiza el código de las aplicaciones Android para captar información semántica de alto nivel. Se construye gráficas de las llamadas a partir de las relaciones de las invocaciones de funciones dentro de la aplicación con niveles de seguridad y extraer más atributos de nodo de las estructuras gráficas mediante la red neural gráfica, generando una representación vectorial vía Word2Vec por redes neuronales gráficas para posteriormente realizar la detección de malware en este entorno.

Las aplicaciones de Android están codificadas en JAVA en una máquina virtual Dalvik (DVM) y tiempo de ejecución de Android (ART).

El proceso consta de:

Exploración de la dirección del desarrollo de malware basada en el aprendizaje de representación gráfica y el nuevo sistema de detección CGDroid mediante la red neural gráfica y la técnica Word2Vec

CGDroid combina análisis estático con representación gráfica para aprender a realizar detección de malware para Android de precisión y eficiencia y capturar información semántica de los gráficos, demostrando ser superior a otros métodos de detección existentes.

El código fuente de una aplicación es compilado en código de bytes de Dalvik, y estas instrucciones contienen semántica crítica en el

código llamado Smali como: Cadena constante, instrucción const-string, URL maliciosas y comandos de sistemas maliciosas.

3.3.3.2. Research on Data Mining of Permission-Induced Risk for Android IoT Devices,

Investigado por (Kumar & Zhang, 2019, pp. 1-22) para la detección de malware en dispositivos Android mediante la clasificación, selección de características con similitudes y reglas de asociación para la minería de permisos, mejorando la precisión por el algoritmo de bosque aleatorio para dispositivos de IoT (Internet de las cosas).

3.3.3.3. CatraDroid: A Call Trace Driven Detection of Malicious Behaiviors in Android Applications

Sugerido por (Sun et al., 2019, pp. 63-77), cuyo método aplicado es la detección de malware sustentado en el aprendizaje automático, por medio de Catradroid, que aprovecha la técnica de minería de texto para capturar una lista sensible de API a partir del conocimiento previo de exploits, bases de datos, configuración de bases y ejemplos de código, construyendo una gráfica Android que identifica las huellas de entrada y el seguimiento de funciones para discriminar o clasificar malware. Existen varias formas de extraer características de comportamiento malicioso de aplicaciones Android como son permisos de usuario, la semántica de llamadas de API, frecuencia de éstas, código bloque y relación de paquetes, similitud de gráficos de dependencia entre API, y abstractas.

La clasificación radica en tres etapas:

1. Generar una lista de API sensibles desde la rama que explota la base de datos, códigos de muestra y mediante la técnica de minería de texto la configuración de bases de código.

2. Identificar las huellas de llamadas de entrada, construyendo una gráfica de llamada completa.
3. Codificar las huellas de las llamadas a características de vectores, aplicando el aprendizaje supervisado con diferentes modelos de clasificadores para diferenciar el malware de aplicaciones benignas.

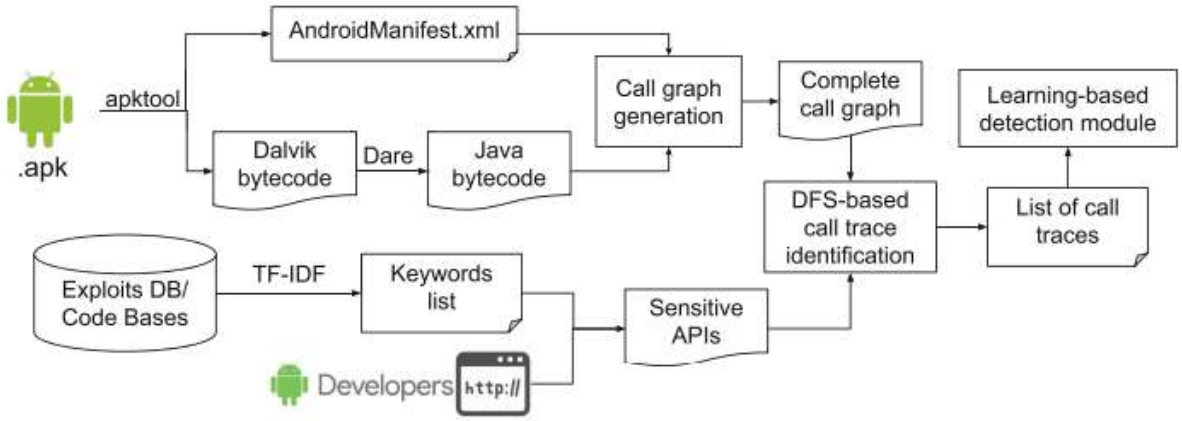

Figura 17Implementación de CatraDroid
Fuente:(Sun et al., 2019, pp. 63-77)

Como se aprecia en la figura N°25, Apktool y Dare, obtienen el código de bytes para mostrar el archivo AndroidManifest.xml, mediante IBM WALA, el cual genera la gráfica de llamada y la huella de identificación. En la generación del gráfico de la llamada de la aplicación, búsqueda del origen múltiple en amplitud, devoluciones de llamada e intenciones implícitas, la sustitución de la información de tipo componente con la de categoría fundamental para controlar la diversidad de características derivadas por la profundidad del recorrido en la profundidad del gráfico de llamadas. El aprendizaje basado en la detección del módulo se fundamenta en las ya construidas (algoritmo scikit-learn).

Es así, que CatraDroid supervisa el enfoque del aprendizaje de clasificación para detectar malware usando la semántica de la llamada de la gráfica de Android.

3.3.3.4. Discovering optimal features using static analysis and a genetic search based method for Android Malware detection

La metodología propuesta por (Firdaus et al., 2018, pp. 712-736) consta de 4 etapas:

Recopilación de datos: se la realiza, tanto para datos benignos como como malignos. Para los primeros serán las aplicaciones de Play Store (alrededor de 1209) y cuya validación se la realiza en línea y con más de 40 aplicaciones de virus, incluida VirusTotal, mientras que, del lado de datos malignos, se utilizó Drebin con una total de 5560 en 179 familias de malware aplicando ingeniería inversa. De este proceso se obtendrá el código de aplicación, un archivo AndroidManifest.xml del proceso inverso y archivos .java y una carpeta en común "res".

Identificación de cadenas: a partir del código de aplicación es necesario la identificación de características de las cadenas de permiso, identificadores, comandos del sistema operativo, funciones de código presentes en 307 muestras de malware, como createSubprocess, la cual no se halla en las benignas, código de funciones de permiso que se encuentra en el archivo .xml necesario en cada "directorio de la aplicación Android de cada paquete de permiso, actividades, servicios, receptores de radiodifusión y proveedores de contenido"(Firdaus et al., 2018, pp. 112-136), así como también las características de la ruta del directorio y funciones de los comandos del sistema

Búsqueda y características genéticas. La búsqueda genética (GS) soportada por algoritmos genéticos (GA) que proporciona una técnica para mejorar la generación, excluyendo aquella incomparable. Este proceso es continuo y repetitivo para mejorar siempre las características que se clasifican en tres categorías: permiso Android (leer, recibir y escribir servicio de mensajes cortos, SMS), Basado en código (checkPermission y com.Android browser. application.id) y

Ruta de Directorio (directorio (system / bin / secbin) quedando claro la existencia mayoritaria de características malware que las benignas.

Máquina Clasificador de aprendizaje (GA): la construcción del modelo de aprendizaje automático predictivo, a partir de los clasificadores por el entrenamiento del conjunto de datos, permite predecir si una aplicación es maligna o benigna. La herramienta WEKA permite cambiar el formato por uno con delimitadores (,) generando el archivo .csv, para luego convertirlo en un archivo atributo – relación .arff. Este aprendizaje debe ser medido, mediante el método de validación cruzada.

La evaluación se calcula tomando en cuenta las métricas de confusión aplicadas a la máquina de evaluación del clasificador de aprendizaje.

El clasificador de Naive Bayes (NB) fue el mejor con un porcentaje del 96,8 %

El optar por GA, por el número de veces de repetición de106 funciones y la reducción del número de características mejoran la predicción de aprendizaje.

3.3.3.5. Anti-emulation trends in modern packers: a survey on the evolution of anti-emulation techniques in UPA packers,

Como habíamos citado anteriormente, la lucha existente entre desarrolladores de malware y programadores de sistemas antimalware, es continua y creativa; los primeros para evitar ser detectados con facilidad y los segundos para lograrlo.

Existen varios procedimientos de protección de código maligno, que cambian la forma en que este se presenta, tales como: cifrar el cuerpo del código (manteniendo las funcionalidades), empaquetar el mismo, con la finalidad de pasar inadvertido. Según (Liţă et al., 2017,

pp. 123-143), tanto los Oligomórficos como los Polimórficos utilizan descifradores (Oligomorfismo: varios) y los segundos (un mismo descifrador), alternando con algunas técnicas como el intercambio de registros, permutación de rutinas, reordenamiento, sustitución y trasposición de código. Hay que destacar que, debido a que el virus no cambia o es muy similar, obtener el código original de carga útil es importante.

La imposibilidad de generar todos los descifradores, obliga a iniciar emuladores para la detección y como contraparte nacen los anti - emulación, anti - máquinas virtuales, anti – sandbox, anti – depuración, anti – desmontaje.

Los anti – emulación requieren actualizaciones (parches) que difícilmente pueden ser cambiados dentro del mismo, pero una vez realizado los empacadores no podrán ser alterados para detener a los emuladores.

Los empaquetadores frecuentes son:

- **Upatre** (UPA1), con procesos de: instrucciones de ensamblaje extrañas (“monitoreo de mecanismos para cifrar la carga útil en combinaciones de base 64, comprensión RTL y un cifrado de XOR”(Liţă et al., 2017, pp. 123-143)), **Funciones de API raras** (la llamada a una API desconocida, provocará la finalización de la aplicación), **Estructura PEB** (Bloqueo de entorno de proceso, obtener la imagen de las bibliotecas cargadas al comprobar el valor de NtGlobalFlag), **Devolución de llamadas TLS** (calcula las direcciones de las bibliotecas del Kernel: net32, ntdll y RtlDecompressBuffer, si éstas no se inicializan correctamente se bloquearán), **Resultados de API Windows** (comprueba el valor de retorno de la función API como técnica de emulación), ***Devoluciones de llamada*** **FastPebLockRoutine** (Rutina de aseguramiento de PEB, descifra la carga útil cifrada y la inyecta en un proceso nuevo creado), **Devolución de llamada de SecureMemoryCache** (Memoria Cache Segura, registra una

llamada de función de devolución invocando para activar la devolución de la llamada registrada RtlFlushSecureMemoryCache), **Devolución de llamada de TopLevelExceptionFilter** ***(****Filtro de excepción de nivel superior****),*** **Devoluciones de llamada de CreateWindows ExA** ***(****creación de una Ventana superpuesta extendida* ***), Bucles grandes*** (millones de iteraciones)

- **Gamarue** (UPA2): **Verificación de la Pila** (se ejecuta la emulación comprobando la dirección de la pila), **Comprobación de la bandera** (la anti – emulación comprueba el inicio de ejecución con el estado de la bandera ZF activo), **Estado del registro después de las llamadas a la API** (comprueba los valores de los registros particulares, para conocer si éstos se modifican luego de una llamada específica de API de Windows), **Devolver valores de llamadas a API**, ("Verificar el valor de una API específica, si es menor que 0 o iguala 1, salta a una dirección incorrecta y se bloquea, caso contrario continúa normalmente" (Liţă et al., 2017, pp. 107-126)) **y Presencia de DLL** y valores de encabezado (Verifica la presencia de archivos .dll y algunos campos de las cabeceras).
- **Hedsen (UPA3), Valores de retorno de las llamadas a la API** (Comprueba valores de RegisterClassA son distintos de cero y si acmStreamConvert devuelve un valor específico), **Mensajes de ventana** (Como se puede observar se referencia a las llamadas en el paso y recepción de parámetros en la devolución de llamadas).

Todos éstos son usados por malware para evitar los anti - emuladores.

3.3.3.6. Deep learning for image-based mobile malware detection

La inyección de código malicioso (*payloads)* a aplicaciones legítimas obtenidas desde Google Play con un 86 %, se llama Reempaquetado; el cual modifica las mismas (carga útil), para luego volver a cargarlos en los sitios oficiales. Con este ataque, la aplicación se instala en el dispositivo, mientras malware espera el momento o las condiciones para propagarse. Esto es posible porque se utiliza interfaces GUI proporcionados por el framework, haciéndolo muy difícil detectarlos. Como en todos los métodos para detección de malware, es necesario, contar con los datos de las muestras de imágenes de archivos ejecutables, recopilando un gran número características para construir clasificadores de malware (binario), la familia (clases múltiples) y sus variantes (clases múltiples), como se observa en la Figura 26. De acuerdo a (Mercaldo & Santone, 2020, pp. 123-148), la ***Generación de funciones,*** consiste en convertir un binario a imagen; siendo la secuencia de bytes la escala de grises de una imagen .png tomando en cuenta un ancho predefinido, longitud variable y el tamaño del binario (cada byte se convierte en número que definirá un color de pixel, que existirá en escala de grises).

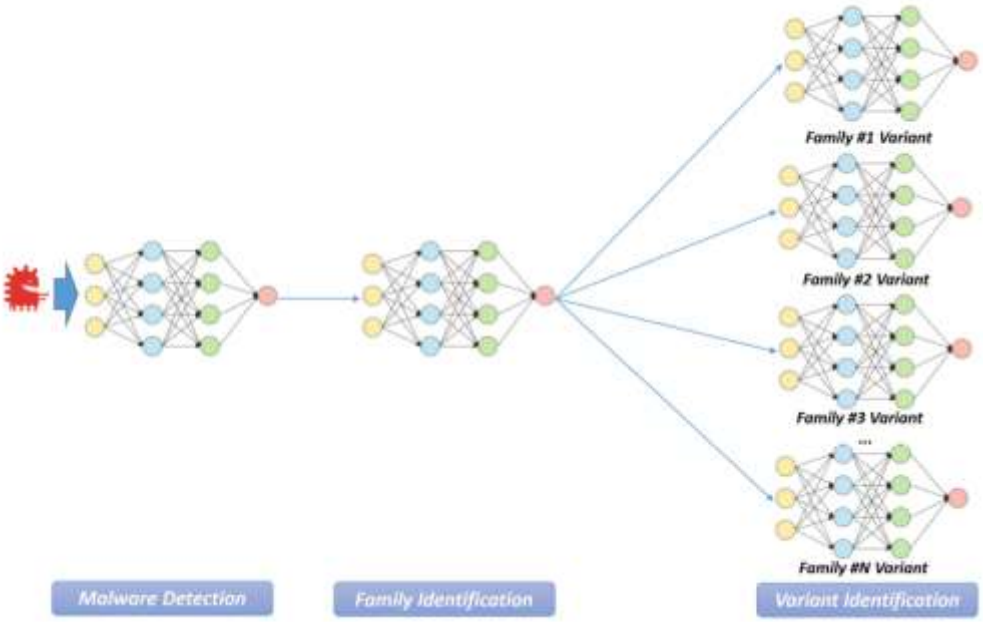

Figura 18Arquitectura principal
Fuente:(Mercaldo & Santone, 2020, pp. 123-148)

Los tres modelos se construyen con una red de aprendizaje profundo neuronal de k capas ocultas, como se aprecia en la Figura 26

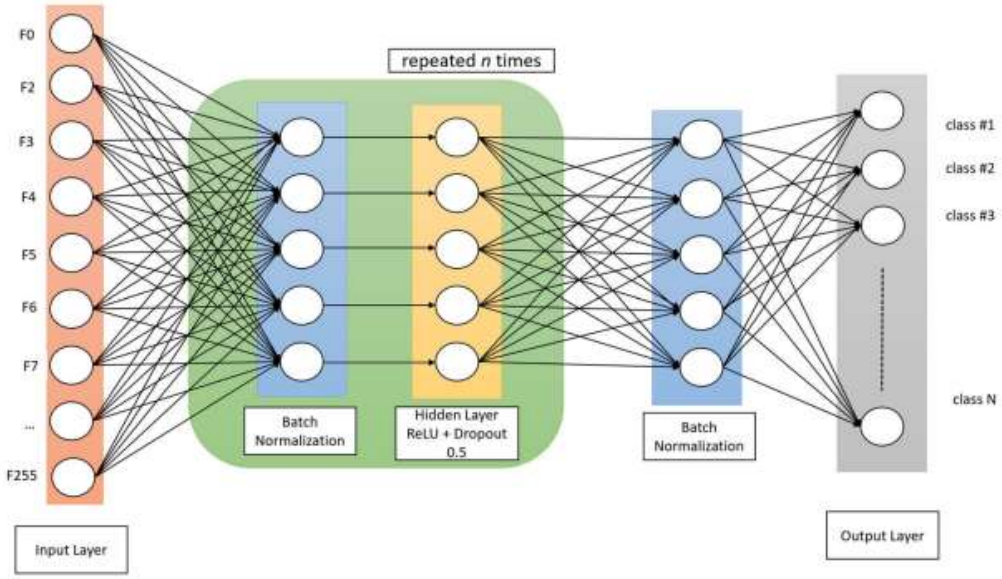

Figura 19Diseño de una red neuronal de aprendizaje profundo
(Mercaldo & Santone, 2020, pp. 123-148)

Entre las capas constan:

Capa de entrada: es la entrada a la red. Un nodo para cada característica ***Capas ocultas:*** consta de neuronas artificiales (perceptrones y la salida es la suma ponderada de las entradas a través de una función de activación (función sigmoidea, ReLu o soft plus).

Capa de abandono: el objetivo es reducir la complejidad del modelo para evitar demasiados ajustes, que desactiven aleatoriamente algunas neuronas en una capa con cierta probabilidad p, de una distribución de Bernoulli. La red reducida se entrena sobre los datos en cada etapa y los nodos desactivados se reinsertan en la red con sus pesos originales (Mercaldo & Santone, 2020, pp. 123-148).

Normalización por lotes: se usa para mejorar el entrenamiento mediante el método de aprendizaje (feed–forward profundo).

Redes neuronales: aumenta la velocidad, tasas más altas de aprendizaje, flexibilidad en la inicialización de parámetros, saturaciones no lineales.

Capa de salida: mediante una operación lineal, cada entrada se conecta con una salida.

El MLP fue capacitado al usar entropía cruzada como una función de pérdida y estocástica del descenso de gradiente (SGD) para optimizar la función de pérdida.

La clasificación de malware se basa en el ***entrenamiento profundo*** de la red neural con un número creciente de capas y ***evaluación de rendimiento de la clasificación,*** mediante la realización de pruebas.

En primer lugar, se infiere una función a partir de los datos (histograma de escala de grises) que proporciona el aprendizaje profundo para que la red pueda discriminar las características que pertenecen a las clases (detección de: malware, familias y variantes) definiendo los límites entre los vectores numéricos.

Las funciones inferidas deben ser libres de sesgos, ya que generará modelos para predecir una clase incorrecta.

Para la evaluación, es fundamental considerar instancias no incluidas en los datos de entrenamiento mediante validación cruzada dividida en: datos de entrenamiento y el conjunto de datos de prueba.

El modelo se complementa mediante aprendizajes automáticos basados en: Árboles de decisión, Random Forest (RF), RandomTree (RT), Bayesian Network (BN) y AdaBoost.

3.3.4. Metodología de Ingeniería Inversa

3.3.4.1. A semantics-based approach to malware detection

Las etapas de un virus son tres, Propagación, Ataque y Defensa y es en ésta última, en la que los constructores de malware le ponen mayor esfuerzo, debido a que, si son detectados, el objetivo del malware quedaría anulado. Según (Preda et al., 2008, p. 25.1-25.54) se utilizan varias técnicas como empaquetadores, emuladores, cifrado y ofuscación. La ofuscación tiene el deber de "cambiar la sintaxis, pero no su intención". Dentro de este análisis, existen varios trabajos que manejan ciertas ofuscaciones de código de malware y es así que, el método propuesto consiste en un framework basado en la semántica para razonar y evaluar la fortaleza de detectores a pesar de las transformaciones de ofuscación.

Se menciona algunas definiciones utilizadas a este aporte:
O: Transformación confusa
P: Potente transformador de programa (ofuscador)
D: Detector
M: Malware

"Cuando un programa P, está infectado con un malware M, escribimos M$\boxed{\supset}$ → P. Intuitivamente, un detector de malware es sólido, si nunca afirma erróneamente que un programa está infectado, es decir, no hay falsos positivos, y es completa si siempre Detecta

Programas que están Infectados, es factible de, no hay falsos negativos"(Preda et al., 2008, p. 25.1-25.54). El estudio se basa en varias definiciones que se enumeran:

"Definición 1. Un detector de malware D está completo para una ofuscación O ∈ O si ∀ M ∈ P, O (M) ⟲ → P ⇒ D (P, M) = 1. A de software malicioso detector D es de sonido para una ofuscación O ∈ O si ∀ M ∈ P, D (P, M) = 1 ⇒ O (M) ⟲ → P"(Preda et al., 2008, p. 25.1-25.54).

"Definición 2. Un oracle es un algoritmo sobre programas. Por ejemplo, un CFGOracle es un algoritmo que toma un programa como entrada y produce control del diagrama de flujo" (Preda et al., 2008, p. 25.1-25.54).

"Definición 3. Un detector de malware D(or), es Oracle-completo con respecto a una ofuscación OR(cfg), si D(or), está completo para esa ofuscación O cuando todos los oracles en el conjunto O es perfecto. Solidez del oracle de un detector D(or) se puede definir de una manera similar" (Preda et al., 2008, p. 25.1-25.54).

El método propone que una clasificación de técnicas de ofuscación sustentadas en las transformaciones ocasionadas en la semántica del programa, analizando la solidez e integridad de Detectores, puede ser diseñada.

3.3.4.2. IoT Application-Layer Protocol Vulnerability Detection Using Reverse Engineering

En la carrera de encontrar soluciones a las vulnerabilidades de los protocolos de seguridad de red del Internet de las cosas (IoT), se cuenta con unos de los métodos más eficaces como Fuzzing para detectarlas, sin embargo, una de las limitaciones, no sólo para este, sino de manera general es la cantidad de archivos de prueba para ampliar su alcance.

En su publicación, Application-Layer Protocol Vulnerability Detection Using Reverse Engineering, (Luo , et al., 2018, pp. 1-13), utiliza técnicas de ingeniería reversa de protocolos para analizar formatos de mensajes en la capa de aplicación de IoT y mediante la generación de mensajes con campos errados, crear archivos de prueba. El mensaje de protocolo identifica puntos de cambio en cada secuencia de bytes, acorde a las propiedades estadísticas para dividirlos en segmentos, los cuales, serán procesados a través de ocurrencias basados en el análisis de probabilidades para identificar la posición de campos de: palabras clave, datos, desconocidos y la generación de procedimientos con operaciones mutadas sobre los campos del mensaje para así construir archivos de prueba, reduciéndolos en número para el método de fuzzing.

Para la detección de los puntos de cambio existen dos formas: Formulación Bayesiana y Minmax. La primera asume un el punto de cambio τ tiene una distribución previa conocida como anterior y debe extenderse a múltiples puntos de cambio, mientras que la segunda supone un punto de cambio y la distribución estadística, son conocidas.

El algoritmo de segmentación de mensajes, lo divide en un conjunto de campos asociados a un protocolo específico que se concatenan para formar un nuevo mensaje según al tiempo de aparición y un sufijo de profundidad.

Los dos conjuntos de puntos de cambio se fusionan y el mensaje estaría segmentado en los puntos de cambio y mediante un preprocesamiento llamado ocurrencia se aplica para filtrar los campos de datos cuya probabilidad de ocurrencia de los segmentos son muy pequeños, por lo tanto, el campo de datos se puede encontrar con aquellos segmentos estadísticamente con valor cero.

Por último, se evalúa mediante mensaje de segmentación y fuzzing. Los protocolos normalmente utilizados son HTTP, FTP, SMTP, POP, DNS, y QQ que se utilizan en la capa de aplicación.

3.3.4.3. Lightweight versus obfuscation-resilient malware detection in android applications

La metodología propuesta tiene dos partes: la técnica de detección de malware ofuscado y un método Lightweight (rápido, ligero y fácil de implementar). De acuerdo con (Aghamohammadi & Faghih, 2019, pp. 125-139), el archivo APK en Android, tiene la misma función que un binario ejecutable de Windows, cuyo archivo principal es "classes.dex" que después se relaciona con un Dalvik.exe (DEX) y puede ser ejecutado en cualquier dispositivo a través de Dalvik Virtual, siendo legible para un ser humano y se hacen referencia mediante números hexadecimales.

Técnica resistente de Ofuscamiento, la evasión a los detectores de malware se realiza mediante ofuscación de las aplicaciones con cifrado de datos, direccionamiento indirecto de llamadas, reordenamiento de código e inserción de código. Entre los métodos más destacado se menciona RevealDroid que consta de cuatro etapas: nivel de paquete, uso de API (PAPI), uso de API de nivel de método (MAPI), características de reflexión y nativas. El método propuesto utiliza el orden de la secuencia de bytes para detección de malware, usando redes neuronales convolucionales y filtros de aprendizaje, para extraer las características n-gram más importantes, mediante redes neuronales recurrentes (RNN).

LightDroid, selecciona un mínimo de características del archivo AndroidManifest y con las imágenes del archivo ejecutable de Dalvik, la precisión es alta. "Primero, se extrae estas características de todas las muestras y luego se crea un único vector booleano para el aprendizaje, si este se vuelve muy grande debemos reducir el tamaño, considerando sólo

las funciones que ocurren más en malware en comparación de las con muestras benignas"(Aghamohammadi & Faghih, 2019, pp. 125-139)

Luego de la experimentación, se comprueba que LightDroid es eficiente por su precisión y por lo ligero que es, debido al uso de los recursos que necesita.

3.3.4.4. A survey of data mining techniques for malware detection using file features

La importancia de este artículo reside en la recopilación de diversas técnicas basadas en el análisis dinámico para la detección y evasión de malware.

Se cita definiciones y diferentes tipos como: Gusanos, virus, caballo de Troya, Software espía, Bot, Rootkit, entre otros.

(Egele et al., 2012, p. 6.1-6.42) analiza varios casos de vectores de infección dentro del software para infectar un sistema como: explotación de servicios vulnerables a través de la red, descargas Drive-by, análisis estático y dinámico, y dentro de éstas tenemos las siguientes:

Monitoreo de llamadas de función,

- Interfaz de programación de aplicaciones (API),
- Llamadas al sistema,
- Sistema de interfaz de llamada,
- API nativas de Windows,
- Resultado de Hooking de funciones,

Implementación de funciones Hooking,

- Reescritura binaria

Seguimiento de llamadas de función post procesamiento.

Análisis de parámetros de función

Seguimiento del flujo de información

- Fuentes de contaminación y sumideros

- Dependencias de datos directos
- Abordar las dependencias
- Controle las dependencias del flujo.
- Flujo de información implícito
- Implementación de sistemas de seguimiento del flujo de información

Seguimiento de instrucciones

Puntos de extensibilidad de inicio automático

Los desarrolladores de malware construyen técnicas de evasión a los programas detectores, mismos que utilizan entornos, herramientas y procesos para realizarlo.

(Egele et al., 2012, p. 6.1-6.42) hace referencia a:

Análisis en el espacio de usuario / kernel,

Análisis en un emulador,

Emulación de memoria y CPU

Emulación de sistema completo.

Análisis en una máquina virtual,

Restablecimiento del entorno de análisis,

Solución de software.

Instantáneas de máquinas virtuales.

Soporte de hardware para instantáneas

Simulación de red,

Sin Internet, pero red simulada,

Acceso a Internet filtrado

Análisis de malware

Código y empaquetadores auto modificables

Mitigar el problema de los empacadores

Ingeniería inversa del algoritmo de desembalaje

Detección genérica y reconstrucción de código empaquetado

Detección de entornos de análisis.

- Hardware.
- Entorno de ejecución.

- Aplicaciones externas.
- Comportamiento.

Contrarrestar las técnicas anti-análisis.

Bombas lógicas

Rendimiento del análisis

Herramientas de análisis de Malware

- Anubis
- Exploración de rutas múltiples
- CWSandbox
- Norman SandboxJoebox
- Ether: Malware Analysis via Hardware Virtualization Extensions
- WILDCAT
- Hookfinder
- Dealing with Packed Binaries

3.3.4.5. Data augmentation and transfer learning to classify malware images in a deep learning context

Para detectar malware mediante una red de aprendizaje profunda (Marastoni et al., 2021, pp. 1-19), cita que es necesario contar con un conjunto grande de datos para generar modelos que generalicen eficientemente, siendo estos, los de imágenes basadas en técnicas de aumento y la aplicación de una secuencia de cambios sintácticos, pero preservando la semántica de código (ofuscación) , a un conjunto pequeño de datos de programa para crear un conjunto más grande. Luego se diseña dos modelos de aprendizaje: red neuronal convolucional (CNN, LSTM) y otra de memoria bidireccional a largo y corto plazo, que se entrenan con imágenes extraídas de binarios compilados del nuevo conjunto generado. En este punto se toma las características de aprendidos de los binarios ofuscados, por medio de aprendizaje por transferencia.

Entre los problemas que pueden encontrarse en los modelos, son la poca data con la que se cuenta y esto se soluciona con la reutilización de una parte de ellos que ya se encuentren entrenados. Además, “se comprueba que es posible utilizar funciones aprendidas a partir de la clasificación de un conjunto de datos binarios personalizados para clasificar un conjunto de datos del mundo real” (Marastoni et al., 2021, pp. 1-19). Entre las características del método tenemos las siguientes:

- Diseño de modelos para clasificar binarios ofuscados de sus imágenes
- Comparación exhaustiva de los enfoques
- Validación de los modelos en dos malwares de última generación.
- Experimentos de transferencia de aprendizaje exitosos entre modelos entrenando con diferentes conjuntos de datos.

N°	Metodología	HooKing	Redes Bayesianas	Ingeniería Inversa		Método
1	Título	"Exploring the API Calls for Malware Behavior Detection using Concordance and Document Frequency"			x	Llamadas API
	Revista o Conferencia	Blue Eyes Intelligence Engineering and Sciences Publication			x	Creación de Procesos Almacenamiento de memoria de
	cita	(Murthy et al., 2019, pp. 4991-4997)				Redes neuronales
	Método propuesto	Concordance para buscar la secuencia de llamada API				
2	Título	"Early detection of ransomware by indicator analysis and WinAPI call sequence pattern"			x	Llamadas API
	Revista o Conferencia	3rd International Conference on Information and Communication Technology for Intelligent Systems			x	Creación de Procesos Almacenamiento de memoria
	cita	(Sharma & Kant, 2018, pp. 201-211)				Redes neuronales
	Método	Análisis de indicadores y secuencia de llamadas de WinAPI				
3	Título	"System call API obfuscation"			x	Llamadas API

	Revista o Conferencia	11th International Symposium, RAID 2008			x	Creación de Procesos Almacenamiento de memoria
	cita	(Srivastava, 2008, pp. 421-422)				Redes neuronales
	Método	Ejecución de monitores de referencia en la interface				
4	Título	"Markhor: detección de malware utilizando similitudes difusas de secuencias de dependencia de llamadas del sistema."			x	Llamadas API
	Revista o Conferencia	J Comput Virol Hack Tech (2021)			x	Creación de Procesos Almacenamiento de memoria
	cita	(Lajevardi, AM et al., 2021)				Redes neuronales
	Método	Markhor, secuencia de dependencia de llamadas al sistema				
5	Título	"HAL-Based Resource Manipulation Monitoring on AOSP"			x	Llamadas API
	Revista o Conferencia	IEEE Transactions on Information Forensics and Security			x	Creación de Procesos Almacenamiento de memoria
	cita	(Thien-Phuc et al., 2020, p. 9)				Redes neuronales
	Método	Hardware Abstract Layer (HAL) Monitoreo de manipulación de recursos por aplicaciones maliciosas				
6	Título	"Runtime Detection Framework for Android Malware"			x	Llamadas API

	Revista o Conferencia	Mobile Information Systems			x	Creación de Procesos Almacenamiento de memoria
	cita	(Kim et al., 2018)				Redes neuronales
7	Título		“Heurística en productos antivirus”		x	Llamadas API
	Revista o Conferencia		ESET		x	Creación de Procesos Almacenamiento de memoria
	cita		(«ESET», 2019)		x	Redes neuronales
8	Título		“Android Malware Detection Via Graph Representation Learning”		x	Llamadas API
	Revista o Conferencia		Mobile Information Systems		x	Creación de Procesos Almacenamiento de memoria
	cita		(Feng et al., 2021)		x	Redes neuronales
	Método	CGDroid				
9	Título		“Research on Data Mining of Permission-Induced Risk for Android IoT Devices”		x	Llamadas API
	Revista o Conferencia		MDPI		x	Creación de Procesos Almacenamiento de memoria
	cita		(Kumar & Zhang, 2019, p. 277)		x	Redes neuronales
10	Título		“CatraDroid: A Call Trace Driven		x	Llamadas API

			Detection of Malicious Behaiviors in Android Applications ”			
	Revista o Conferencia		Machine Learning for Cyber Security		x	Creación de Procesos Almacenamiento de memoria
	cita		(Sun et al., 2019, pp. 63-77)		x	Redes neuronales
	Método	CatraDroid	aprendizaje de clasificación para detectar malware usando la semántica de la gráfica de llamada Android.			
11	Título		“Discovering optimal features using static analysis and a genetic search based method for Android malware detection”		x	Llamadas API
	Revista o Conferencia		Frontiers of Information Technology and Electronic Engineering		x	Creación de Procesos Almacenamiento de memoria
	cita		(Firdaus et al., 2018, pp. 712-736)		x	Redes neuronales
	Método	Detección de malware mediante Búsqueda Genética (GS) por algoritmos Genéticos (GA)				
12	Título		“Anti-emulation trends in modern packers: a survey on the evolution of anti-emulation techniques		x	Llamadas API

			in UPA packers”			
	Revista o Conferencia		Journal of Computer Virology and Hacking Techniques		x	Creación de Procesos Almacenamiento de memoria
	cita		(Liţă et al., 2017, pp. 107-126)		x	Redes neuronales
	Método	Técnicas de anti-emulación en empaquetadores UPA				
13	Título		“Deep learning for image-based mobile malware detection”		x	Llamadas API
	Revista o Conferencia		Journal of Computer Virology and Hacking Techniques		x	Creación de Procesos Almacenamiento de memoria
	cita		(Mercaldo & Santone, 2020, pp. 157-171)		x	Redes neuronales
	método	Reempaquetado de aplicaciones legitimas				
14	Título			“A semantics-based approach to malware detection”	x	Llamadas API
	Revista o Conferencia			ACM Transactions on Programming Languages and Systems	x	Creación de Procesos Almacenamiento de memoria
	cita			(Preda et al., 2008, pp. 1-54)	x	Redes neuronales
	Método	Técnicas de ofuscación sustentadas en las transformaciones ocasionadas en la semántica del programa, analizando la solidez e integridad de Detectores				

15	Título			“IoT Application-Layer Protocol Vulnerability Detection using Reverse Engineering”	x	Llamadas API
	Revista o Conferencia			School of Electronic and Information	x	Creación de Procesos Almacenamiento de memoria
	cita			(Luo, et al., 2018, pp. 1-54)	x	Redes neuronales
	Método	Segmentación y método Fuzzing				
16	Título			“Lightweight versus obfuscation-resilient malware detection in android applications”	x	Llamadas API
	Revista o Conferencia			Journal of Computer Virology and Hacking Techniques	x	Creación de Procesos Almacenamiento de memoria
	cita			(Aghamohammadi & Faghih, 2019, pp. 125-139)		Redes neuronales
	Método	Técnicas resistentes de detección de malware ofuscado y LightDroid				
17	Título			“A survey of data mining techniques for malware detection using file features”	x	Llamadas API

	Revista o Conferencia			Proceedings of the 46th Annual Southeast Regional Conference on XX	x	Creación de Procesos Almacenamiento de memoria
	cita			(Siddiqui et al., 2008, pp. 509-510)		Redes neuronales
	Método	Recopilación de diversas técnicas, herramientas y métodos, basadas en el análisis dinámico para la detección y evasión de malware				
18	Título			"Data augmentation and transfer learning to classify malware images in a deep learning context"	x	Llamadas API
	Revista o Conferencia			Journal of Computer Virology and Hacking Techniques	x	Creación de Procesos Almacenamiento de memoria
	cita			(Marastoni et al., 2021)	x	Redes neuronales
	Método	Red de aprendizaje profunda				

Tabla 4:Comparación de metodologías para la detección de Malware
Fuente: Elaboración propia

La tabla comparativa de doble entrada extrae procesos internos necesarios para demostrar los métodos propuestos y está estructurada de la siguiente manera:

Consta de 5 filas: Metodología, Título del Artículo, Nombre de la Revista o conferencia, cita y Método propuesto, y 5 columnas: Hooking,

Redes Bayesianas, Ingeniería Inversa, columna verificadora y la última con los procesos principales de cada metodología.

Para tabular los resultados se tomó en cuenta la Metodología, y los procesos que cumplen, de esa manera podemos cuantificar la relevancia de éstas.

N°	HooKing	Redes Bayesianas	Ingeniería Inversa	Total
Llamadas API	6	7	5	18
Creación de Procesos Almacenamiento de memoria	6	7	5	18
Redes neuronales		7	3	10
Total	12	21	13	46

Tabla 5: Cuantificación de resultados
Fuente: Elaboración propia

Ratificando la premisa inicial de esta revisión sistemática y explorando la tabla, todas las metodologías deben crear procesos mediante llamadas al sistema API, ya sea para gestionar recursos o servicios (creación de procesos, reservar espacios de memoria y almacenamiento, guardar en disco duro o dispositivos externos, utilizaciones de funciones, clases, objetos, comparación con la lista blanca incluidos en el SO, para que se incluyen como normales y benignos o para toma de decisiones (hooking) por el usuario para manejar ofuscaciones, cifrados, a partir de la secuencia de código binario y demás clasificadores de Malware, que consiste en solicitar autorización para el uso de recursos, ya sea en Modo Kernel o Modo Usuario, de igual manera como fueron concebidos los Sistemas Operativos en su origen. La ejecución de procesos se la ideó, en un orden determinado y esta estructura al desarrollar malware (software)no se aparta de ella y debe seguir de la misma manera, de cómo se ingresa a los sistemas u ordenadores, la forma de propagarse y explotar mediante la ejecución de otros para cumplir con su fin;

esto conlleva a que todas las metodologías se canalicen en la forma de como ingresan.

La comparación se basa en función de los procedimientos que utilizan para lograr detectar malware, ya sea en su estructura, análisis de código semántico y sintáctico, empaquetamientos, secuencia de bytes, código binario, llamadas de API de gráficos de comportamiento, como en los objetivos; unos por cumplir para hacer daño y otros por impedirlo.

Lamentablemente ningún procedimiento o implementación de sistemas o equipos de prevención serán suficientes, cuando la parte sensible de ejecución de los sistemas, ordenadores e infraestructura de comunicación, son los SERES HUMANOS como la parte más vulnerable y fácil de atacar. Minimizar resultados y alcance, es la decisión más crítica por parte de los usuarios.

3.4. Plan de Prevención, Concientización y Mitigación

Ante los riesgos a los que se expone el país, una organización, a nivel público, privado o personal, dispositivos o equipos electrónicos; es importante contar con medidas o planes de prevención para eliminar o atenuar efectos dañinos, ya sean globales o individuales.

Para garantizar un nivel de seguridad se debe tomar en cuenta los controles de seguridad CIS que consideran información de ataques realizados, acciones de mitigación, prevención y defensa.

(*CIS Controls*, 2021), se fundamenta en los siguientes principios:

1. Utilizar información de los ataques reales que han sucedido y que alimentan una base de conocimiento para construir defensas.
2. Inversión prioritaria en controles para garantizar entornos seguros.
3. Establecer parámetros globales que conozca la institución, fáciles de identificar, implementar y medir.

4. Monitorear constantemente para diagnosticar y mitigar.
5. Automatizar procesos de defensa que sean confiables, escalables y continuas.

Las bondades de las plataformas corporativas de seguridad EndPoints ejecutan evaluaciones diarias, valoración de resultados y mitigación de aquellos sistemas desactivados y actualización de definiciones de virus (*CIS Controls*, 2021).

De manera general, para evitar ser atacados por un malware se recomienda los siguientes procesos:

1. Establecer Políticas de Seguridad que deben ser aprobadas por los directivos de la organización, con relación a la protección de activos: Hardware y Software, normas de uso e integración de equipos de cómputo autorizados o no, periféricos, instalación y actualización de software y uso de dispositivos de almacenamiento. Controles CIS 1 y 2: Inventario de Dispositivos y Software Autorizados o no Autorizados. CIS N°11: Configuración segura de los equipos de red, tales como cortafuegos, enrutadores y conmutadores.
2. Dentro de las tareas administrativas debe constar el establecimiento de normas de contratación de estricto cumplimiento:
    - Contrato laboral debidamente firmado, acuerdo de confidencialidad, Tareas de formación con aprobación en temas de prevención de riesgos laborales debidamente certificados.
    - Alta de usuario en el dominio corporativo
    - Asignación de un ordenador personal
    - Instalación de software requerido por el personal de TI
    - Permisos de acceso o carpetas del servidor y acceso a aplicaciones
3. Instalar un software antivirus/malware, EDR, XDR (preferible con licencia, versiones originales y completas) CIS N°8: Defensa contra malware.

4. Actualizar el software instalado (SO y aplicaciones, las actualizaciones cubren brechas de seguridad) CIS N°2: Inventario de Software autorizados y no autorizados.
5. Automatizar el escaneo de aplicaciones, software instalado o dispositivos de almacenamiento interno, en el equipo en períodos programados o necesarios CIS N°3: Gestión continua de vulnerabilidades.
6. Monitorear el tráfico de red (evitar conexiones WIFI abiertas, puertos abiertos que no se necesitan ni se usan) CIS N°5: Configuración Segura de Hardware y Software, CIS N°9: Limitación y control de puertos de red, protocolos y servicios, CIS N°15: Control de acceso inalámbrico.
7. Precaución con redes sociales, medidas con el navegador cuando no sean sitios seguros, no abrir archivos que se desconozca su origen y procedencia, ni caer en promociones o premios falsos. Antes de abrir enlaces, verificar con los actores directos, comunicándose con ellos y por último evitar publicaciones personales como fotografías personales de familia, viajes o lugares del desarrollo de actividades cotidianas. CIS N°7: Protección de correo electrónico y navegador web
8. Realizar copias de seguridad, almacenamiento continuo y asegurar la información personal (autenticación de doble factor, contraseñas robustas y cifradas y diferentes) de preferencia en la nube, CIS N°10: Capacidad de recuperación de datos, CIS N°13: Protección de datos, CIS N°16: Procedimientos y herramientas.
9. No abrir archivos .exe, ya estos podrían inyectar código malicioso. CIS N°16: Procedimientos y herramientas.
10. Observar el marco jurídico establecido en el país.
11. Capacitación, difusión y concientización, sobre políticas de seguridad (el ser humano es el elemento más sensible de la organización), CIS N°17: Implementar un programa de concienciación y capacitación en seguridad.

    Educación a Usuarios.

    - Aprender a reconocer un ataque de ingeniería social para evitar no proporcionar información corporativa o personal a entes no autorizados.

- Proteger adecuadamente su puesto de trabajo, en relación con el antivirus, actualizaciones, correo electrónico, etc.
- Cómo aplicar controles de acceso físico: dependencias autorizadas, acompañamiento en todo momento a clientes, proveedores y visitantes, etc.
- Tratamiento y manejo adecuados de los soportes y dispositivos móviles, como portátiles, smartphones, etc.
- Comprender el acceso a páginas web externas, aplicaciones de terceros, descargas o actualizaciones no validadas por el departamento de informática («INCIBE», 5/02, pp. 1-20).

Luego de la Capacitación es necesario continuar con el Seguimiento, Evaluación, Retroalimentación y establecimiento de Controles, estos procedimientos constituyen el **Plan de Concientización**.

En cuanto al establecimiento de puntos de Control, es importante involucrar a todo el personal y ensayar de pruebas de ingeniería social y Phishing y evaluar el cumplimiento de las normas, políticas y procedimientos, establecidos en la capacitación (Pruebas simuladas de ataques cibernéticos).

En caso de haber sido atacados, es necesario contar con un **Plan de Continuidad de Negocio**, el cual garantizará las condiciones mínimas para el funcionamiento de la organización. Se debe observar los siguientes pasos:

- Para todos los planes, se debe contar con una organización administrativa que sea responsable de la ejecución de estos y la debida autorización de los directivos para su ejecución.
- El plan de continuidad debe conocer cuáles son las actividades críticas y contar con niveles de redundancia (líneas de alimentación, backups, varios servidores e imagen de datos y aplicaciones, para evaluar su estado de funcionamiento y dar prioridad a las actividades de recuperación.
- Considerar los tiempos mínimos de recuperación planificados para los recursos o servicios vulnerados.

- Contar con los recursos necesarios para reiniciar las tareas prioritarias.
- En caso de pérdida de datos o cifrado de los mismos, antes de recuperarlos a partir de los backups, es necesario implementar seguridades para evitar nuevos daños y empezar el proceso de mitigación.
- Los empleados de la organización deben conocer los procesos de recuperación para que se cumplan las acciones previstas.
- Todos los planes deben ser revisados, actualizados y probados.

Se debe considerar que:

La Seguridad de la organización es responsabilidad de todos.

La experiencia frente a agresiones genera la mayor conciencia frente a ataques cibernéticos y además se evalúa al personal sobre el nivel de respuesta y manejo de incidentes.

CAPÍTULO IV

4. CONCLUSIONES Y RECOMENDACIONES

Al final del estudio realizado a las diversas metodologías y su valoración, se concluye que:

La revisión sistemática permitió la recopilación de artículos publicados en revistas indexadas con relación a métodos propuestos para la detección de malware, mediante el uso de bases de datos y los filtros aplicados por cuartiles (para las revistas), de esta manera se obtuvo los códigos ISSN en SCImago. Con los ISSN y las palabras claves como: Malware, Detection, Methodologies, Methods, Bayesian Network, Reverse Engeneering, Hooking, de Scopus, se encontró los artículos analizados que, sin embargo, se redujeron en cantidad, por la imposibilidad de descargarlos por restricciones y costos. A pesar de estas limitaciones, se logró obtener varias publicaciones, porque algunos autores accedieron al envío de sus publicaciones, luego de las solicitudes enviadas por correo.

El procedimiento de C, que inyecta código a archivos .DLL, mediante el almacenamiento, intercambio y actualización de direcciones en memoria y creación de procesos, a pesar de que malware contenga código empaquetado, cifrado, ofuscado en varias capas, mediante llamadas al sistema o secuencia de llamadas API, debido a que todas las aplicaciones, requieren crear procesos para ejecutarlos. Sin embargo, es necesario contar con personal capacitado al momento de tomar la decisión para autorizar la ejecución de un evento, ya que dependerá de éste, salvaguardar la información, la seguridad del equipo y de la infraestructura de comunicaciones.

La aplicación de Redes Bayesianas, basan su principio en el análisis del comportamiento de Malware para clasificarlos, mediante llamadas API, variables (nodos) y las relaciones que existen entre ellas, la interconexión, dependencia, contorno, y la incertidumbre que se crea, define la estructura, el contorno y la probabilidad de la ocurrencia de una evento, sin embargo, la implementación de sistemas de inteligencia artificial, deducirán si se está frente a un ataque de malware con mucha certeza (entre el 93 y 97%) y se crean modelos para evitar ataques posteriores de los mismos, de sus variantes o familias.

La Ingeniería inversa requiere de un laboratorio aislado, con el fin de blindar los activos de la organización, y realizar el proceso de desmontaje, cifrado y des ofuscamiento del código de malware, con la finalidad de identificar la estructura, funcionalidad, procedimientos y posibles efectos, así como también, descubrir brechas de seguridad que tiene la organización.

Ninguna metodología puede considerarse completa y global, porque el crecimiento de la creación de código maligno, sus formas, sus objetivos y variantes, se actualizan y continúan en ascenso, más los métodos y procedimientos para detectarlas, por sí solas son insuficientes, debido a que no existe ningún sistema cien por ciento seguro.

Hay que destacar que la orientación para obtener una solución de detección de malware más eficaz es la inteligencia artificial (EDR / XDR) con redes neurales de aprendizaje profundo para evaluarlas como tal, de la familia a la que pertenecen o de sus variantes.

El establecer Planes de Prevención y Concientización, es la mejor forma de evitar ataques cibernéticos, especialmente malware (basándose en el análisis comparativo de las diversas metodologías de detección).

Limitantes en telefonía móvil, es la capacidad de memoria, almacenamiento y tiempos de respuesta.

La mejor de forma de evitar ataques, está en el ser humano, siendo disciplinados en cumplir con normas de Prevención, mediante el establecimiento de una cultura de procesos, disciplina, ordenamiento y dispuesto a seguir instrucciones (Políticas de Seguridad).

5. BIBLIOGRAFÍA

ADMWARE Advance Data Manegement. (s. f.). *¿QUÉ ES UN ENDPOINT?* https://admware.es/que-es-un-endpoint/

Advance Networks. (2021, septiembre 30). *Cuáles con las principales herramientas de protección a EndPoints?* https://advance-nt.com/2021/09/30/cuales-con-las-principales-herramientas-de-proteccion-a-endpoints/

Aghamohammadi, A., & Faghih, F. (2019). *Lightweight versus obfuscation-resilient malware detection in android applications*. *16*, 125-139. https://doi.org/10.1007/s11416-019-00341-y

Aguilera, R. (2014). *¿Revisión sistemática, revisión narrativa o metaanálisis? 21*. https://dx.doi.org/10.4321/S1134-80462014000600010

Algirde Pipikaite, & Marc Barrachin, Scott Crawford,. (2021, febrero). *Los riesgos de Ciberseguridad crecen en The Global Risks Report 2021*. https://www.datasec-soft.com/blog/losriesgosdeciberseguridadcrecenglobalrisksreport2021

Análisis de Malware Estático y Análisis de Malware Dinámico—2021—SOFTWARE. (2021, marzo 26). https://es.weblogographic.com/difference-between-static-malware-analysis-and-dynamic-malware-analysis-7238

Arquitectura de un sistema operativo. (2016, septiembre 9). *arquitecturapablop.blogspot.com*. http://arquitecturapablob.blogspot.com/2016/09/arquictectura-de-los-sistemas.html

Bassov, A. (2005, octubre 18). *Hooking the native API and controlling process creation on a system-wide basis*. Code Project. https://www.codeproject.com/Articles/11985/Hooking-the-native-API-and-controlling-process-cre

Belcic, I. (2021, mayo 19). ¿Qué es el malware? *Avast Academy*. https://www.avast.com/es-es/c-malware

Bissell, K., Lasalle, R., & Dal, P. (2020). *INNOVATE FOR CYBER RESILIENCE*. *4*, 1-46. https://www.accenture.com/_acnmedia/PDF-116/Accenture-Cybersecurity-Report-2020.pdf#zoom=40

Características, Estructura y Clasificación de Virus Informáticos. (2013, mayo 23). *grupo4introduccionunesr*. http://grupo4introduccionunesr.blogspot.com/2013/05/v-behaviorurldefaultvmlo.html

Castañón, B. (2012). *Auditoría de sistemas de información*. https://www.gestiopolis.com/auditoria-de-sistemas-de-informacion/

Castellanos, J. (2015). *Q-Floid—Análisis de malware en Android por flujo de datos* [Universidad Autónoma de los Andes]. https://repositorio.uniandes.edu.co/handle/1992/13109

Chilán, B., & Macías, E. (2014, diciembre). Estudio de metodologías para defensa contra virus informático que pueden dañar el equipo de cómputo Metodología para la defensa contra virus. *4/10/2014*, *5*(2), 1-22.

Ciberseguridad. (s. f.). LAS MÉTRICAS DE SEGURIDAD MÁS IMPORTANTES PARA CUMPLIR LA NORMATIVA. *Ciberseguridad*. https://ciberseguridad.com/servicios/metricas-seguridad/

CIS, Center for Internet Security. (2021). https://www.cisecurity.org/controls/

Cruz, E. (2013). *Llamadas al Sistema*. https://silo.tips/download/llamadas-al-sistema-sistemas-operativos

Egele, M., Scholte, T., Kirda, E., & Kruegel, C. (2012). A survey on automated dynamic malware-analysis techniques and tools. *ACM Computing Surveys*, *44*(2), 6.1-6-42. Scopus. https://doi.org/10.1145/2089125.2089126

El Tiempo. (2019, noviembre 20). *El cibercrimen no descansa, estas son las proyecciones para el 2020*. https://www.eltiempo.com/tecnosfera/dispositivos/cifras-de-ciberataques-de-2019-y-tendencias-para-el-2020-435508

ESET. (s. f.). *Introducción al glosario de ESET*. https://help.eset.com/glossary/es-CL/packer.html

ESET. (2019, octubre 15). *¿Qué es la heurística?* https://support.eset.com/es/kb127-que-es-la-heuristica

ESET. (2021). *SECURITY REPORT LATINOAMÉRICA 2020*. https://security-report.eset-la.com/

Feng, P., Jianfeng, M., & Xindi, M. (2021). *Androide Malware Detection Via Graph Representation Learning*. *2021*. https://doi.org/10.1155/2021/5538841

Ferrer, S. (2011). *DETECCIÓN DE MALWARE A PARTIR DEL COMPORTAMIENTO DEL NAVEGADOR*.

Firdaus, A., Razak, M. F. A., Karim, A., & Anuar, N. (2018). *Discovering optimal features using static analysis and a genetic search based method for Android malware detection*. *6*, 712-736. https://doi.org/10.1631/FITEE.1601491

Giusto, D. (2018, mayo 21). *Cryptojacking y explotación: Una combinación que marca tendencia*. https://www.welivesecurity.com/la-es/2018/05/21/cryptojacking-explotacion-vulnerabilidades/

Hard2bit Dr. (2013, septiembre 10). *Análisis de Malware: Enfoque y caso práctico /Seguridad Informática*. https://hard2bit.com/blog/analisis-de-malware-enfoque-y-caso-practico/

Harley, D. (2007, septiembre 15). *Heurística en productos antivirus*. https://www.welivesecurity.com/la-es/2007/09/15/heuristica-productos-antivirus/

Harley, D. (2009). Making sense of anti-malware comparative testing. *Information Security Technical Report*, *14*(1). Scopus. https://doi.org/10.1016/j.istr.2009.03.002

INCIBE. (2017). *Ransomware: Una guía de aproximación para el empresario*. https://www.incibe.es/sites/default/files/contenidos/guias/doc/guia_ransomware_metad.pdf

INCIBE. (2020). *Las 7 fases de un ciberataque. ¿Las conoces?* https://www.incibe.es/protege-tu-empresa/blog/las-7-fases-ciberataque-las-conoces

INCIBE. (5/02). *DESARROLLAR CULTURA EN SEGURIDAD*. https://www.incibe.es/sites/default/files/contenidos/dosieres/metad_desarrollar-cultura-en-seguridad.pdf

ISSN. (s. f.). *¿Qué es el número ISSN?* [Org]. ISSN International Standard Serial Number International Centre. https://www.issn.org/es/comprender-el-issn/que-es-el-numero-issn/

itResearch. (2020, mayo 21). *Ciberseguridad en 2020, de obligado cumplimiento*. https://www.ittrends.es/whitepapers/content-download/5332c96a-7bb0-4498-85b1-e8b46faa0af6/doc-ejecutivo-it-trends-ciberseguridadl-2q2020.pdf

J Kolter, & Maloof, M. (2004). *Learning to Detect and Classify Malicious Executables in the Wild∗. 7*, 470-478. https://doi.org/10.1145/1014052.1014105

JÍMENEZ, JAVIER. (2020). *Qué es y cómo detectar malware ofuscado en un servidor*. https://www.redeszone.net/tutoriales/seguridad/que-es-malware-ofuscado/

Kartz, G. (2021). *2021 Global Threat Report*. CrowdStrike. https://go.crowdstrike.com/rs/281-OBQ-266/images/Report2021GTR.pdf

Kim, T., Kang, B., & Im, E. G. (2018). *Runtime Detection Framework for Android Malware*. *2018*. https://doi.org/10.1155/2018/8094314

Kumar, R., & Zhang, X. (2019). *Research on Data Mining of Permission-Induced Risk for Android IoT Devices*. *9*, 277. https://doi.org/10.3390/app9020277

Lajevardi, AM, Parsa, S, & Amiri, MJ. (2021). Markhor: Detección de malware utilizando similitudes difusas de secuencias de dependencia de llamadas del sistema. *J Comput Virol Hack Tech (2021)*. https://doi.org/10.1007/s11416-021-00383-1

Liţă, C. V., Gavrilu, D., & Cosovan, D. (2017). *Anti-emulation trends in modern packers: A survey on the evolution of anti-emulation techniques in UPA packers*. *14*, 107-126. https://doi.org/10.1007/s11416-017-0291-9

López, D. (2012, junio 6). Llamadas al sistema en Windows. *SYSTOPE*. http://systope.blogspot.com/2012/06/llamadas-al-sistema-en-windows.html

Luo , J.-Z., Shan, C., Cai, J., & Liu, Y. (2018). IoT Application-Layer Protocol Vulnerability Detection using Reverse Engineering. *School of Electronic and Information*, *10*.

Marastoni, N., Giacobazzi, R., & Preda, M. D. (2021). *Data augmentation and transfer learning to classify malware images in a deep learning context*. https://doi.org/10.1007/s11416-021-00381-3

Mercaldo, F., & Santone, A. (2020). *Deep learning for image-based mobile malware detection*. *16*, 157-171. https://doi.org/10.1007/s11416-019-00346-7

Method for defending against dll injection without hooking. (2011). https://patents.google.com/patent/KR101097590B1/en

Microsoft. (2016). *Qué es el «Código administrado»*. https://docs.microsoft.com/es-es/dotnet/standard/managed-code

Monnappa—2018—Learning Malware Analysis.pdf. (s. f.).

Murthy, G., Chowdary, M., Sangameswar, M., & Vital, T. (2019). *Exploring the API Calls for Malware Behavior Detection using Concordance and Document Frequency*. *8*, 4991-4997. https://doi.org/10.35940/ijeat.F9144.088619

Natour, L. (2017, agosto 17). El 43% de los ciberataques se concentran en las pymes. *ABC software*. https://www.abc.es/tecnologia/informatica/software/abci-43-por-ciento-ciberataques-concentran-pymes-201708081259_noticia.html

Oracle. (2011). *Guía de administración del sistema: Servicios de red*. https://docs.oracle.com/cd/E24842_01/html/E22524/rfsintro-101.html

Pandey, K; Mehtre, M. (2014). *Performance of malware detection tools:A comparison*. 1811-1817. https://doi.org/DOI: 10.1109/ICACCCT.2014.7019422

Peng Xie∗, , Jason H Li∗, , Xinming Ou†, , Peng Liu‡, & , Renato Levy∗. (s. f.). *Using Bayesian Networks for Cyber Security Analysis*. https://www.researchgate.net/publication/220957917_Using_Bayesian_Networks_for_Cyber_Security_Analysis

Preda, M. D., Mihai, C., & Somesh, J. (2008). *A semantics-based approach to malware detection*. *30*, 1-54. https://doi.org/10.1145/1387673.1387674

Rahimian. (2014). *Sobre la Ingeniería inversa de la bonet CITADEL*.

Román, V. (2019, abril 25). Algoritmos Naive Bayes: Fundamentos e Implementación [Www.medium.com]. *Ciencia y Datos*. https://medium.com/datos-y-ciencia/algoritmos-naive-bayes-fudamentos-e-implementaci%C3%B3n-4bcb24b307f

Rübke, M. (2018, octubre 10). *¿Por qué los hackers cometen ciberataques?* https://www.madboxpc.com/por-que-los-hackers-cometen-ciberataques/

Sánchez, J. (2010). Cómo realizar una revisión sistemática y un meta-análisis. *2010*, *38*, 53-64.

Sanz, M. (2019, marzo 30). *¿Qué es y en qué consiste la Ingeniería Inversa?* https://computerhoy.com/reportajes/tecnologia/consiste-ingenieria-inversa-396691

Sharma, H., & Kant, S. (2018). *Early detection of ransomware by indicator analysis and WinAPI call sequence pattern*. *107*, 201-211-2019. https://doi.org/10.1007/978-981-13-1747-7_20

Shocking Soft. (s. f.). http://www.shockingsoft.com/AnalyzeIt.html

Siddiqui, M., Wang, M. C., & Lee, J. (2008). *A survey of data mining techniques for malware detection using file features*. 509-510. https://doi.org/10.1145/1593105.1593239

SOPHOS. (s. f.). *Detección y respuesta de puntos finales (EDR)*. https://www.sophos.com/en-us/products/endpoint-antivirus/edr.aspx

Srivastava, A. (2008). *System call API obfuscation*. *5230 LNCS*, 421-422. https://doi.org/10.1007/978-3-540-87403-4_36

Struct EPROCESS. (s. f.). *Nirsoft*. https://www.nirsoft.net/kernel_struct/vista/EPROCESS.html

Sucar, L. (s. f.). *Redes Bayesianas*.

Sun, C., Chen, J., & Feng, P. (2019). *CatraDroid: A Call Trace Driven Detection of Malicious Behaiviors in Android Applications*. 63-77. https://doi.org/10.1007/978-3-030-30619-9_6

Tecnozero. (s. f.). *¿Qué es un EDR? ¿Por qué es diferente de un antivirus?* https://www.tecnozero.com/antivirus-y-anti-ransomware/que-es-un-edr/#caracteristicas-clave-del-edr

TENDENCIAS 2016 (IN) SECURITY EVERYWHERE. (ESET (Enjoy Safer Technology)). https://www.welivesecurity.com/wp-content/uploads/2016/01/tendencias-2016-insecurity-everywhere-eset.pdf

Thien-Phuc, D., Jungsoo, P., & Souhwan, J. (2020). *HAL-Based Resource Manipulation Monitoring on AOSP*. *2020*, 9. https://doi.org/10.1155/2020/8863385

Tian, R., Batten, L., & Versteeg, S. (2008). *Function Length as a Tool for Malware Classification*. 69-76. https://dro.deakin.edu.au/eserv/DU:30018116/batten-functionlengthasatool-2008.pdf

Tubau, E., & Cánovas, D. (2002). *Inferencias bayesianas: Una revisión teórica*. https://www.researchgate.net/publication/277272935_Inferencias_bayesianas_una_revision_teorica

UNIOVI, es. (s. f.). *Lenguajes de programación*. http://di002.edv.uniovi.es/~dani/asignaturas/apuntes-leccion2.PDF

www.welivesecurity.com. (2014). *Hooks en la API de Windows y cómo se roba información*. https://www.welivesecurity.com/la-es/2014/12/03/hooks-api-windows-robo-informacion/

ANEXOS

ANEXO 1: Monitoreo de procesamiento de la Técnica de Hooking

Llamada al Kernel

Como se aprecia en la Figura 13, las llamadas se llevan a cabo de forma dinámica con GetProcAddress[1] y se observa la copia del contenido de una parte de memoria a otra y está cifrado como se explicó en el párrafo anterior ($memcpy$).[2]Al mirar el código notamos un ciclo que descifra dicho contenido, logrando un ejecutable desde la dirección de memoria asignada $0x333130$[3], para luego trasladar el control de este código al malware, de donde se logra uno, cifrado[4] . Se continúa con una llamada a la subrutina 0x251084, la misma que recibe el nombre del proceso inyectado con el código. Verificará que si un navegador y empezará a robar datos de navegación.

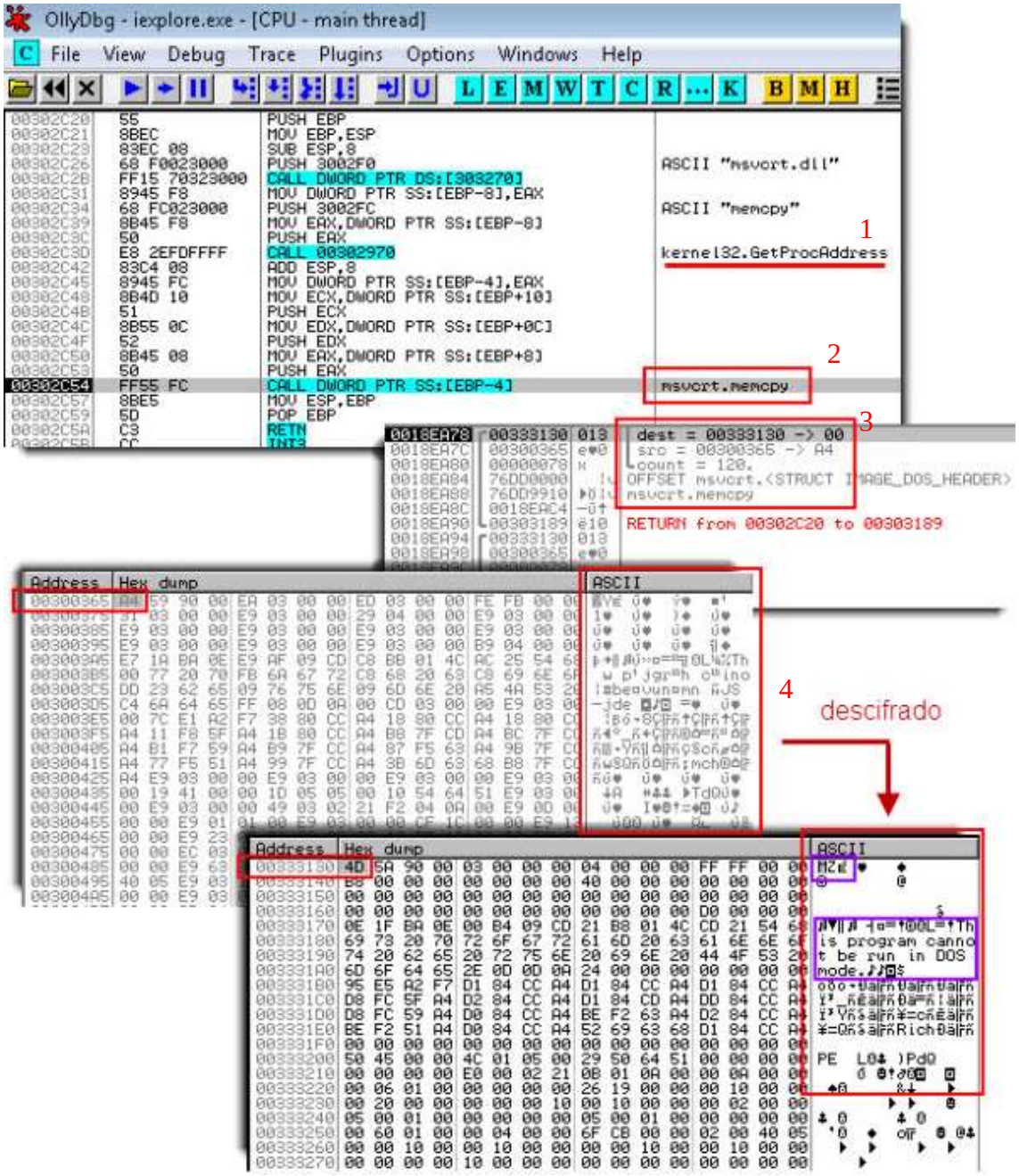

Figura 20Llamada al Kernel
Fuente: (Gallop, 2014)

Como se explicó anteriormente, cada proceso realiza siempre una llamada al Kernel o llamada al usuario, primero se observa una llamada a VirtualAlloc[5] para reservar memoria, después GetProcAdderss[6] para la dirección de 4 rutinas de ws2_32.dll (*getaddrinfo*[6,1], gethostbyname,[6.2] send[6.3] y WSASend[6.4]), las cuales serán *hookeadas*[7] cada vez que iexplore.exe las requiera y ésta sea

intervenida antes de que se realice la rutina y se ejecute el código malicioso, como se observa en la siguiente Figura 14

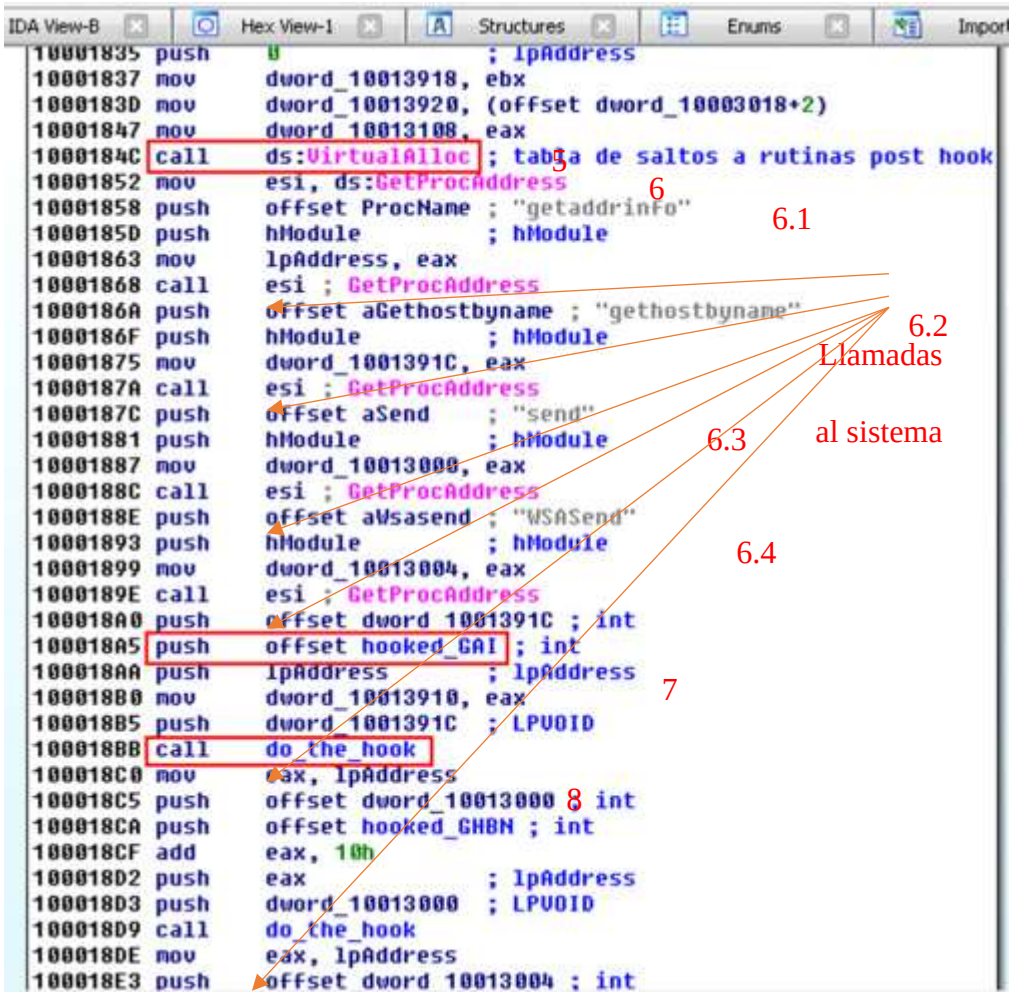

Figura 21Obtención de código "do_the_hook"
Fuente: (Gallop, 2014)

Posteriormente se llama a la subrutina do_the_hook, cuya funcionalidad se observa en la Figura 14

IDA Pro

"Al igual que OllyDbg, IDA Pro es un depurador / desensamblador a nivel de aplicación que ayudará enormemente en seguir la pista de la ejecución del programa"(Hard2bit Dr., 2013).

Figura 22Modificación de Rutina con introducción de JMP y uso de VirtualProtect
Fuente: (Gallop, 2014)

El código de do_the_hook, altera la dirección de cada rutina obtenida con GetProcAdress para construir un "salto incondicional hacia otro ubicación y se introduce un JMP[1] en la primera instrucción de la rutina a ser hookeada" (www.welivesecurity.com, 2014). Podemos ver también a VirtualProtect[2] para conceder y retirar permisos antes y después de la escritura en la dirección de memoria (www.welivesecurity.com, 2014)[1].

[1] Ejemplo de inyección de código a la librería.dll

Anexo 2 Creación y Control de Procesos

Para la creación de un proceso existen varias formas de hacerlo y es a través de: NtCreateProcess (), CreateProcess () que configura estructuras en modo Kernel sin recurrir al primero, NtCreateFile (), NtOpenFile () o NtCreateSection ().

No se puede analizar independientemente la creación de un proceso o uno de I/O (entrada / salida), por lo que se requiere hacerlo en su globalidad para deducir si pertenece al hilo de la creación o no (FILE_ALL_ACCESS). La alternativa es realizar un método HOOKING (enganche) a NtCreateSection (), para interceptar la llamada y solicitar mapear el archivo ejecutable como una imagen (atributo SEC_IMAGE) conjuntamente con la solicitud de seguridad de protección de página para su ejecución e identificar si el proceso listo a crearse es seguro.

Al no ser un proceso administrado, es necesario cargar la librería ntdll.dll, se cargará NtCreateSection() con el índice del servicio EAX, mientras EDX señala los parámetros de la función y transferir la ejecución KiDispatchService() como una rutina en modo Kernel y la posterior validación de los parámetros de la misma para implementar el servicio y cuya dirección se encuentra en la **Tabla de descriptores de servicio** (ntoskrnl.exe exporta el puntero como variable KeServiceDescriptorTable) con la siguiente estructura:

```
struct SYS_SERVICE_TABLE {
    void ** ServiceTable;
    CounterTable larga sin firmar;
    ServiceLimit largo sin firmar;
    void ** ArgumentsTable;
};
```

Figura 23Descriptor de Medios
Fuente:(Bassov, 2005) .Licencia de código CPOL (The Code Project Open License)

El campo ServiceTable, referencia a la matriz que contiene las direcciones de todas las funciones que implementan los servicios del Sistema. Para conectar cualquier función de API nativa en todo el sistema debemos escribir la dirección en la i-ésima (i=índice de servicio) entrada de la matriz, apuntada por el campo ServiceTable de KeServiceDescriptorTable. De este análisis se deduce que, para monitorear y controlar

la creación de procesos, debemos conectar aquellas funciones de API, las cuales son obligatorias a ser ejecutadas por el código del nuevo proceso a crearse.(Bassov, 2005).

Control de Creación de Procesos

Cómo se citó anteriormente la estructura del virus tiene varias fases, y la primera es la de propagación, siendo en esta etapa donde se ejecutan los scripts (procesos) o rutinas de archivos ejecutables para insertarse dentro del ordenador.

Es así que la solución para detectar Malware se enfoca en el Monitoreo y Control de la Creación de Procesos que se ejecutarán para su difusión.

Para estas funciones, se crea un controlador en modo Kernel y una aplicación en modo Usuario respectivamente. La aplicación pasa el índice de servicio correspondiente a NtCreateSection(), más la dirección del búfer de intercambio al controlador, mediante el siguiente código (Bassov, 2005):

```
// dispositivo abierto
dispositivo = CreateFile ("\\\\. \\ PROTECTOR", GENERIC_READ | GENERIC_WRITE,
       0,0, OPEN_EXISTING, FILE_ATTRIBUTE_SYSTEM, 0);

// obtener el índice de NtCreateSection y pasarlo al controlador, junto con el
// dirección del búfer de salida
DWORD * addr = (DWORD *)
   (1+ (DWORD) GetProcAddress (GetModuleHandle ("ntdll.dll"),
                                    "NtCreateSection"));
ZeroMemory (búfer de salida, 256);
controlbuff [0] = dirección [0];
controlbuff [1] = (DWORD) & outputbuff [0];
DeviceIoControl (dispositivo, 1000, controlbuff, 256, controlbuff, 256, & dw, 0);
```

Figura 24Código de controlador
Fuente:(Bassov, 2005) .Licencia de código CPOL (The Code Project Open License)

Los códigos auxiliares ntdll.dll comienzan con MOV EAX, ServiceIndex es compatible a cualquier versión de Windows NT (base de los sistemas operativos actuales) y le corresponde al primer byte de los 5 bytes totales de longitud y los 4 restantes para el índice de servicio. Cuando se obtiene este índice, se tiene la certeza de que corresponde a una función API nativa leer estos 4 bytes desde la dirección después del byte del principio del stub.

Después se asigna el búfer de intercambio al espacio de direcciones del Kernel con MmMapIoSpace() y se escribe la dirección de la función Proxy en la Tabla de Servicio (variable global RealCallee) y sobrescribir la entrada asignando la dirección de destino MmMapIOSpace() pues es posible que ésta pueda residir en memoria, sólo de lectura y es necesario comprobar si se tiene acceso a escritura, para dejar de lado la protección de la página y tener la certeza de acceso a escritura de la página destino.

```
NTSTATUS DrvDispatch (IN PDEVICE_OBJECT dispositivo, IN PIRP Irp)
{
    UCHAR * potenciador = 0; ULONG a, base;

    PIO_STACK_LOCATION loc = IoGetCurrentIrpStackLocation (Irp);

    if (loc-> Parameters.DeviceIoControl.IoControlCode == 1000)
    {
        buff = (UCHAR *) Irp-> AssociatedIrp.SystemBuffer;

        // tabla de despacho del servicio de enlace
        memmove (& Index, buff, 4);
        a = 4 * Índice + (ULONG) KeServiceDescriptorTable-> ServiceTable;
        base = (ULONG) MmMapIoSpace (MmGetPhysicalAddress ((vacío *) a), 4,0);
        a = (ULONG) & Proxy;

        _asm
        {
            mov eax, base
            mov ebx, dword ptr [eax]
            mov RealCallee, ebx
            mov ebx, a
            mov dword ptr [eax], ebx
        }

        MmUnmapIoSpace (base, 4);

        memmove (& a, & buff [4], 4);
        salida = (char *) MmMapIoSpace (MmGetPhysicalAddress ((void *) a), 256,0);
    }

    Irp-> IoStatus.Status = 0;
    IoCompleteRequest (Irp, IO_NO_INCREMENT);
    return 0;
}
```

Figura 25Controlador IOCTL recibido de la aplicación
Fuente:(Bassov, 2005) Licencia de código CPOL (The Code Project Open License)

La función Proxy() guarda registros y estado de banderas, empuja un puntero a los parámetros del servicio en la pila y llama a la función Check(); que dependerá si es True para continuar. Entonces Proxy() restaura registros y banderas y transfiere el control a la implementación del servicio, caso contrario, escribirá a EAX, STATUS

ACCESS DENIED, restauraciones EPX desde la instancia que llama a NtCreateSection() con el estado de la llamada a fracasado con el estado de error.

```
// esta función decide si debemos
// permitir que la llamada a NtCreateSection () sea exitoso
Comprobación de ULONG __stdcall (PULONG arg)
{

    MANIJA mano = 0; archivo PFILE_OBJECT = 0;
    Información de POBJECT_HANDLE_INFORMATION; ULONG a; char * buff;
    ANSI_STRING str; LARGE_INTEGER li; li.QuadPart = -10000;

    // verifica las banderas. Si no se solicita el acceso PAGE_EXECUTE a la sección,
    // no tiene sentido preocuparse por eso
    if ((arg [4] & 0xf0) == 0) return 1;
    if ((arg [5] & 0x01000000) == 0) return 1;


    // obtener el nombre del archivo a través del identificador de archivo
    mano = (ASA) arg [6];
    ObReferenceObjectByHandle (mano, 0,0, KernelMode, & archivo, & info);
    if (! file) return 1;
    RtlUnicodeStringToAnsiString (& str, & file-> FileName, 1);

    a = str.Length; buff = str.Buffer;
    mientras (1)
    {
        if (buff [a] == '.') {a ++; rotura;}
        a--;
    }
    ObDereferenceObject (archivo);

    // si no es ejecutable, no tiene sentido preocuparse por ello
    // devuelve 1
    if (_stricmp (& buff [a], "exe")) {RtlFreeAnsiString (& str); return 1;}

    // ahora vamos a pedir la opinión del usuario.
    // Escribe el nombre del archivo en el búfer y espera hasta
    // el usuario indica la respuesta
    // (1 como primer DWORD significa que podemos continuar)

    // sincronizar el acceso al búfer
    KeWaitForSingleObject (& evento, Ejecutivo, KernelMode, 0,0);
```

```
    // establece los primeros 2 DWORD de un búfer en cero,
    // copia la cadena en el búfer y repite
    // hasta que el usuario establezca el primer DWORD en 1.
    // El valor del segundo DWORD indica el usuario
    // respuesta
    strcpy (& salida [8], buff);
    RtlFreeAnsiString (& str);

    a = 1;
    memmove (& salida [0], & a, 4);
    mientras (1)
    {
        KeDelayExecutionThread (KernelMode, 0, & li);
        memmove (& a, & salida [0], 4);
        si (! a) se rompe;
    }
    memmove (& a, & salida [4], 4);
    KeSetEvent (& evento, 0,0);

    return a;
}

// solo guarda el contenido de ejecución y llama a check ()
_declspec (desnudo) Proxy ()
{
    _asm {
        // guarda el contecto de ejecución y llama a check ()
        // - el resto depende del valor que devuelve check ()
        // si es 1, proceda al destinatario real.
        // De lo contrario, devuelve STATUS_ACCESS_DENIED
        pushfd
        pushad
        mov ebx, esp
        agregar ebx, 40
        empujar ebx
        cheque de llamada
        cmp eax, 1
        bloque de jne

        // proceder al destinatario real
        popad
        popfd
        jmp RealCallee

        // devuelve STATUS_ACCESS_DENIED
        bloque: popad
        mov ebx, dword ptr [esp + 8]
        mov dword ptr [ebx], 0
        mov eax, 0xC0000022L
        popfd
        ret 32
    }
}
```

Figura 26Función Proxy
Fuente:(Bassov, 2005) Licencia de código CPOL (The Code Project Open License)

La función check() tiene como argumento el valor del puntero que apunta a los parámetros del servicio y verifica el estado de las banderas y atributos. Esta función tiene la información para asignar una sección como una imagen ejecutable, o para negar la ejecución por protección de página, de esta manera, se asegura de que NtCreateSection() no esté en la creación del proceso, en este caso check() retorna Verdadero, caso contrario, se comprueba la extensión del archivo subyacente; para mapear algún archivo DLL mediante el atributo SEC_IMAGE, si este archivo no es ejecutable (.exe), entonces check() devuelve Verdadero, caso contrario, brinda la opción al modo de usuario, de tomar su decisión al escribir el nombre del archivo y la ruta al búfer de intercambio hasta obtener una respuesta (Bassov, 2005).

El siguiente código evalúa el buffer de intercambio para verificar si hay alguna petición del controlador, si hay una, comprueba el nombre del archivo y su ruta con la lista de aquellos que puede ejecutar el ordenador, de haber alguna coincidencia retorna Verdadero y se agregará el programa a la lista y se pasa el control al controlador, mediante la escritura en el buffer, para la creación de un nuevo proceso y no se puede iniciar ningún proceso sin la autorización del usuario.

```
hilo vacío ()
{
    DWORD a, x; char msgbuff [512];

    mientras (1)
    {
        memmove (& a, & outputbuff [0], 4);

        // si no hay nada, Sleep () 10 ms y vuelva a comprobar
        si (! a) {Dormir (10); continuar;}

        // parece que se pide nuestro permiso. Si el archivo
        // en cuestión ya está en la lista blanca,
        // dar una respuesta positiva
        char * nombre = (char *) & outputbuff [8];
        para (x = 0; x <stringcount; x ++)
        {
            if (! stricmp (nombre, cadenas [x])) {a = 1; ir a saltar;}
        }

        // pedir permiso al usuario para ejecutar el programa
        strcpy (msgbuff, "¿Quieres ejecutar");
        strcat (msgbuff, & outputbuff [8]);

        // si la respuesta del usuario es positiva, agregue el programa a la lista blanca
        if (IDYES == MessageBox (0, msgbuff, "ADVERTENCIA",
           MB_YESNO | MB_ICONQUESTION | 0x00200000L))
        {a = 1; strings [stringcount] = _ strdup (nombre); stringcount ++;}
        si no a = 0;

        // escribe la respuesta en el búfer y el controlador la importante
        saltar: memmove (& outputbuff [4], & a, 4);

        // dile al conductor que siga adelante
        a = 0;
        memmove (& outputbuff [0], & a, 4);
    }
}
```

Figura 27Hilo de ejecución
Fuente:(Bassov, 2005) Licencia de código CPOL (The Code Project Open License)

De la misma manera debemos extender la aplicación para los dispositivos de Hardware, Entrada y Salida de archivos, tráfico de red para controlar hasta los puertos, sin embargo, la automatización asegurará su funcionamiento.

Sintetizando, malware es un código ejecutable con fines maliciosos con la capacidad de replicarse e infectarse, y se refiere a inyectar la biblioteca de vínculos dinámicos (DLL), mediante la creación de un hilo remoto (es importante analizar el entorno, el hilo al que pertenece, como lo mencionamos anteriormente para definir si es válido y confiable), como método para infiltrarse en un proceso normal, sin embargo siempre se deberá crear proceso o subprocesos registrando la ruta de la DLL que se inyectará en memoria virtual del proceso destino de la siguiente manera:

“Primero, la inyección de DLL mediante la creación de un subproceso remoto se realiza adquiriendo un identificador de proceso de destino para ser penetrado (S1)

y cargue la dirección de función LoadLibrary (xx) de Kernel32.dll en el espacio de memoria virtual obtenido del proceso de destino para ser penetrado. Para aplicaciones en la misma plataforma, Kernel32.dll usa la misma dirección de memoria virtual. La cadena de ruta completa de la DLL que se inyectará se registra en el espacio de memoria virtual del proceso de destino. Solicitudes para crear un hilo remoto usando la API CreateRemoteThread o NtCreateThreadEx según el sistema operativo, pasando la dirección de inicio del hilo a la dirección de la función LoadLibrary (xx) registrada anteriormente, y usando el parámetro como la ruta completa de la DLL inyectar registrado en lo anterior.