Universitat Oberta de Catalunya

Botnet educativa P2P para el robo de criptomonedas: una perspectiva del atacante

Autor: Rubén Espinosa de los Monteros León
Máster Universitario en Seguridad y Privacidad

Tutor: Ángela Maria Garcia Valdés
Empresa: INCIBE

01 / 2024

Resumen

Este proyecto presenta el desarrollo de una botnet educativa P2P centrada en el robo de criptomonedas en entornos Windows, con el propósito de comprender detalladamente la perspectiva del atacante en la creación de una botnet y concienciar sobre esta amenaza.

La investigación se enfoca en diseñar e implementar una botnet controlada remotamente mediante grupos públicos del protocolo Tox. Esta botnet puede realizar ataques de clipboard hijacking, robo de carteras criptográficas almacenadas en los equipos y permite el envío de comandos a los sistemas comprometidos.

Durante las pruebas en un entorno controlado, se evidenció su capacidad para llevar a cabo estos ataques y el envío de comandos de manera efectiva, sin ser detectada

Abstract

This project presents the development of an educational P2P botnet focused on cryptocurrency theft in Windows environments. Its aim is to provide a detailed understanding from the attacker's perspective in creating a botnet and raising awareness about this threat.

The research focuses on designing and implementing a remotely controlled botnet using public groups of the Tox protocol. This botnet is capable of executing clipboard hijacking attacks, stealing cryptographic wallets stored on devices, and allowing the sending of commands to compromised systems.

During testing in a controlled environment, its ability to carry out these attacks and send commands effectively was demonstrated, all without being detected.

Índice

1. Introducción

1.1. Planteamiento del problema

En la actual era digital, la ciberdelincuencia se ha convertido en una de las principales amenazas para gobiernos, empresas y usuarios por igual. Dentro de este contexto, el auge de las botnets representa una tendencia alarmante por su versatilidad como herramienta para diversas actividades cibercriminales.

Las botnets están detrás de algunos de los ciberataques y campañas de malware más notorios de los últimos años. Su capacidad para coordinar miles o incluso millones de dispositivos comprometidos las convierte en el vector ideal para ataques distribuidos de denegación de servicio, envío masivo de SPAM y phishing, cifrado de sistemas mediante ransomware, entre otras acciones delictivas.

Si bien las botnets existen desde hace décadas, se han sofisticado enormemente incorporando técnicas tales como infraestructuras peer-to-peer, fluxing y módulos de inteligencia artificial[1] con el objetivo de dificultar su desmantelamiento. Incluso se han expandido más allá de computadoras para reclutar dispositivos móviles y aparatos IoT. Esta continua evolución las convierte en un desafío cambiante para la seguridad informática.

Los daños potenciales de las botnets son enormes. Pueden ser utilizadas para el robo de credenciales bancarias, para influenciar políticamente[2], ataques contra infraestructura crítica y un sinfín de actividades que amenazan tanto la seguridad nacional como los derechos individuales de los ciudadanos. Incluso se han documentado casos de botnets estado-nación[3], controladas por agencias gubernamentales con fines geopolíticos.

Mientras tanto, en la sociedad existe una gran falta de concienciación sobre la realidad de esta amenaza, que a menudo se percibe como algo ajeno o excepcional. Sin embargo, cualquier usuario, empresa o institución es un objetivo potencial.

Además, entre profesionales de la ciberseguridad predomina un enfoque reactivo, centrado en medidas defensivas como el análisis de malware o la mitigación de ataques. Pero para una protección efectiva es indispensable entender en profundidad cómo operan los propios cibercriminales a la hora de construir y gestionar botnets.

Ante esta realidad, la investigación proactiva de las botnets y el desarrollo de estrategias innovadoras para su detección y desactivación se vuelven esenciales. Este trabajo pretende llevar a cabo el desarrollo de una botnet con fines educativos, con el propósito de obtener una comprensión detallada de su funcionamiento desde la perspectiva del atacante.

Concretamente el desarrollo de la botnet se enfoca en el robo de criptomonedas, un sector en auge con e ataques mediáticos conocidos que involucren botnets. Este robo es crítico porque los activos no están directamente vinculados a la identidad del usuario, lo que dificulta la identificación. Además cibercriminales pueden ocultar el rastro de la transacción utilizando mixers. Por otro lado, persisten lagunas en la legislación internacional, lo que complica la persecución de estos delitos cibernéticos.

1.2. Objetivos del trabajo

El presente apartado tiene como finalidad enumerar los principales objetivos que se pretenden alcanzar con la realización de este proyecto, tanto a nivel general como específico. A pesar de que los objetivos están ordenados por orden de ejecución, es importante destacar que los de especial relevancia son aquellos relacionados con la creación de la botnet, ya que representan el valor diferencial de este proyecto de investigación en el campo de las botnets.

Objetivos generales:

- Realizar un estudio completo sobre las botnets, profundizando en su funcionamiento, evolución histórica, tipologías, propagación, detección y mitigación.
- Desarrollar una botnet educativa que aplique los conocimientos expuestos en la fase teórica, con el objetivo de concienciar al lector sobre la gravedad de esta amenaza y motivar la creación de soluciones innovadoras para su prevención y mitigación.

Concretamente este trabajo tiene como objetivo la creación de una botnet con comunicación P2P orientada al robo de criptomonedas. La plataforma objetivo será dispositivos Windows de 64 bits.

Objetivos específicos:

- Definir qué es una botnet y explicar sus componentes básicos.
- Analizar la historia y evolución de las botnets hasta la actualidad.
- Estudiar en detalle la arquitectura y el funcionamiento de las botnets.
- Conocer los distintos tipos de botnets.
- Investigar los mecanismos utilizados para el reclutamiento de dispositivos y la propagación de botnets.
- Examinar cómo los cibercriminales operan con las botnets para obtener réditos económicos.
- Revisar las técnicas y estrategias para la detección, el análisis y la mitigación de botnets.
- Explorar las técnicas comunes utilizadas por todos los tipos de malware.
- Diseñar e implementar una botnet educativa, documentando detalladamente su desarrollo.

1.3. Metodología

Para la consecución de los objetivos planteados se seguirá la siguiente metodología:

- **Estudio previo:** Se llevó a cabo un estudio previo mediante una búsqueda preliminar de referencias relacionadas con la temática, con el propósito de determinar un punto de partida para el proyecto.
- **Definición del plan de trabajo:** Se elaborará la hoja de ruta que guiará la ejecución del proyecto. Aquí se desde los objetivos y las metodologías hasta los recursos necesarios y los plazos de ejecución.
- **Búsqueda de información:** se realizará una extensa revisión de libros, artículos científicos, informes y otras fuentes para recopilar información actualizada sobre todos los temas objeto de estudio. Esta fase incluye tanto la realización del estudio de mercado como la recopilación de información para llevar a cabo el estudio teórico de las botnets y el estudio teórico sobre malware
- **Estudio teórico sobre botnets:** Se realizará una investigación específica de conceptos, características, arquitectura, propagación, uso criminal, detección, etc. de las botnets.
- **Estudio teórico sobre malware:** Se investigarán las técnicas comunes utilizadas por todos los malware, como la persistencia y la inyección de procesos, con el objetivo de entender su funcionamiento.
- **Desarrollo de botnet educativa:** se diseñará e implementará una botnet a pequeña escala con fines formativos, documentando exhaustivamente su arquitectura y funcionamiento. El desarrollo se llevará a cabo por módulos, para poder probar cada parte de forma independiente.
- **Experimentación:** Se realizarán pruebas de la botnet en un entorno controlado para observar su comportamiento. La fase de experimentación constará de dos fases. Por un lado se realizarán pruebas unitarias a medida que se desarrollen los módulos de la aplicación, para finalmente realizar las pruebas de regresión de toda la aplicación.
- **Evaluación de resultados:** Los datos obtenidos en las fases anteriores se compararán y contrastarán con los resultados obtenidos de las pruebas de la botnet, permitiendo una evaluación integral de su desempeño.
- **Formulación de conclusiones:** se sintetizarán los hallazgos más relevantes.

1.4. Herramientas y tecnologías usadas

Equipo usado

Para la realización de este proyecto cobra especial importancia la elección del equipo a utilizar, que tiene que ser capaz de correr simultáneamente varias máquinas virtuales corriendo la botnet, el entorno de desarrollo en caso de que sea necesario realizar debugging en remoto, el editor de texto y el navegador Web. La máquina que se usará en el proyecto será un MSI-GE62 6QD con las siguientes especificaciones:

- Procesador I7-6700HQ.
- 16GB de RAM (8GB integrados + 8GB que se han adquirido para garantizar la disponibilidad de recursos.
- Disco duro físico de 1TB.
- Disco duro SSD de 128GB.
- Tarjeta gráfica NVIDIA 960M.
- Windows 11 Pro de 64 bits.

Herramientas y tecnologías

Para llevar a cabo el proyecto, se utilizarán las siguientes herramientas

- Chromium como navegador en el que realizar las consultas
- La suite de Libreoffice, para redactar la memoria y preparar la presentación
- GanttProject, como herramienta para hacer el diagrama de Gantt.

En cuanto a las herramientas y tecnologías propias del desarrollo se han seleccionado las siguientes:

- VMWare Workstation Pro 17 para la creación y gestión de máquinas virtuales aisladas donde se ejecutará de forma controlada la botnet desarrollada
- Sistemas operativos Windows 11 Pro instalados en las máquinas virtuales
- Visual Studio 2022 Profesional como entorno de desarrollo para la implementación de la botnet.
- Git como sistema de control de versiones,.
- Wireshark como herramienta de análisis y captura de tráfico de red para controlar las comunicaciones de la botnet creada,
- Process monitor para controlar el funcionamiento del malware desarrollado,
- Bitcoin Core, como cartera de criptomonedas,.

1.5. Análisis de riesgos del proyecto

Para este proyecto, se han identificado una serie de posibles desafíos. A continuación, se describen estos riesgos y las medidas que se han tomado para mitigarlos:

- Retrasos en la entrega del proyecto debido a una planificación temporal demasiado ajustada o imprevistos en alguna fase. Se mitigará estimando el tiempo estimado en los hitos temporales al alza y haciendo un seguimiento periódico del proceso. Adicionalmente se reservarán días de vacaciones por si fuera necesario emplearlos en alguna fase..
- Perdida de datos a causa de un daño en el disco duro. Se mitigará realizando copias de seguridad periódicas que se subirán a la nube. Además se subirá periódicamente el código a la nube usando un sistema de control de versiones.
- Demoras imprevistas en la implementación de la botnet educativa debido a imprevistos o falta de conocimiento. Se mitigará planificando un diseño por módulos, con sus respectivas pruebas unitarias.
- Cambios en el enfoque de desarrollo: Existe el riesgo de que surjan circunstancias imprevistas que provoquen que desarrollo planteado no sea factible. Se mitigará planteando un enfoque alternativo con el tutor en caso que suceda.
- Limitaciones al no conseguir simular fielmente el comportamiento real de la botnet por escasez de recursos en el sistema. Se mitigará asegurando que el equipo utilizado disponga de los recursos necesarios.
- Cambios en el entorno de pruebas tales como actualizaciones de software que alteren el comportamiento correcto de la botnet. Se mitigarán realizando una snapshot de las máquinas que se usarán para realizar las pruebas.
- Detección por parte de los sistemas de seguridad como antivirus u otras tecnologías o bien el envío muestras que puedan llevar a su detección en el futuro. Se desactivará envío de muestras tanto en el ordenador anfitrión como en las máquinas de pruebas.
- Mal uso de la botnet por parte de terceras personas. Se mitigará restringiendo el acceso al código a investigadores o profesionales del sector.

1.6. Tareas a realizar

En este apartado se enumeran las principales tareas a llevar a cabo para completar el presente trabajo de acuerdo con la metodología y objetivos establecidos.

Su secuenciación busca estructurar el trabajo en un proceso ordenado que aborde de forma integral los aspectos teóricos y prácticos del estudio.

A continuación se presenta el listado detallado de estas tareas organizado por fases:

Búsqueda preliminar: Antes del inicio del proyecto se llevó a cabo una búsqueda previa para familiarizarse con el campo a tratar y elegir el enfoque que tendrá la investigación.

Elaboración del plan de trabajo: Se creará la hoja de ruta que seguirá el proyecto, abordando todos los temas necesarios para llevar a cabo el trabajo de forma exitosa.

Estudio de mercado: Se hará un estudio de mercado con el objetivo de ver dónde encaja el proyecto en el contexto actual.

Recopilación de información: se llevará a cabo una extensa búsqueda de fuentes de información relevantes, incluyendo, artículos científicos, libros, páginas web, y otras publicaciones actualizadas sobre botnets, malware y temas relacionados.

Lectura, resumen y organización del Material: Se realizará una lectura completa del material bibliográfico seleccionado. Se elaborarán resúmenes de los textos, destacando los conceptos y datos más relevantes. Además, se llevará a cabo una organización estructurada de la información, incluyendo comentarios analíticos y referencias bibliográficas sobre botnets, malware y temas relacionados para su posterior análisis y referencia."

Estudio teórico sobre botnets: A partir de la información recopilada se organizarán y plasmarán los conceptos clave relacionados con las botnets, incluyendo sus diferentes tipologías, arquitecturas típicas, métodos de propagación, infraestructura de red, métodos de detección y posibles técnicas de mitigación. El resultado de esta tarea será una versión preliminar de la memoria sobre el estudio de las botnets.

Estudio teórico sobre malware: Más allá del campo específico de botnets, se plasmarán los conceptos teóricos sobre los malware genéricos.

Diseño de la botnet educativa: se planificarán sus objetivos, arquitectura, funcionalidades y otros aspectos técnicos.

Implementación de la botnet: se desarrollará y documentará el código fuente de todos los componentes de la botnet educativa.

Pruebas unitarias: Se probarán los distintos módulos de la botnet a medida que estos sean desarrollados.

Pruebas de regresión: Se probará el funcionamiento de la botnet en su conjunto y se simulará el funcionamiento real de la botnet usando varias máquinas corriéndola simultáneamente.

Documentación del desarrollo: Se documentará la implementación de la botnet.

Documentación de las pruebas: Se documentarán las pruebas realizadas.

Redacción de la memoria final: se redactarán los contenidos del trabajo siguiendo la estructura y apartados correspondientes. Se redactarán también los apartados evaluación de resultados, conclusiones y trabajo futuro.

Revisión y edición: se revisará el proyecto para realizar correcciones y mejoras finales en su redacción y presentación.

Preparación de la presentación: se elaborarán los recursos necesarios para la adecuada exposición y defensa del trabajo.

Preparación de la defensa: Se practicará la presentación propuesta. Se revisará el trabajo con el objetivo de aportar respuestas adecuadas ante las preguntas del jurado.

1.7. Impacto ético, social y ambiental

Si bien este proyecto busca generar conocimiento académico sobre las botnets, es importante considerar sus posibles impactos negativos para tomar las debidas precauciones.

En términos éticos, la investigación sobre técnicas de malware conlleva dilemas, ya que dicho conocimiento podría ser utilizado con fines delictivos. Por ello, los resultados se compartirán exclusivamente en medios académicos y de investigación responsables.

La botnet educativa se implementará en un entorno controlado y aislado, sin afectar sistemas reales. Otra consideración relevante es que, por la naturaleza delicada del tema, sería aconsejable no hacer público el código fuente para prevenir usos inadecuados.

A nivel social, se busca crear conciencia sobre la amenaza que representan las botnets y fomentar la investigación de métodos para combatirlas. Aunque se estudian técnicas maliciosas, la divulgación controlada genera más beneficios para la seguridad informática.

Desde una perspectiva legal, e la botnet se utilizará exclusivamente con fines educativos y de investigación, sin ningún propósito económico ni intención de vulnerar ninguna ley. Su implementación y uso se llevarán a cabo únicamente en entornos controlados y aislados.

En cuanto a la protección de la propiedad intelectual, todas las fuentes de información usadas en el proyecto serán debidamente referenciadas. La autoría del proyecto será exclusivamente del autor y en ningún caso se plagiará trabajo de otros autores. Además, se garantiza el uso legal de recursos de terceros, asegurando hacerlo de acuerdo con las licencias y los términos de uso correspondientes.

Respecto al impacto ambiental, al ser un proyecto informático es limitado. Para la realización de las pruebas se empleará unicamente un ordenador con varias máquinas virtuales y se limitará el tiempo de las pruebas al estrictamente necesario, con el objetivo de minimizar el consumo energético, que puede ser elevado por el minado de criptomonedas.

1.8. Planificación temporal

La planificación de este proyecto se ha diseñado considerando una serie de de especial relevancia a la hora de adaptar el plan a las necesidades específicas de la investigación y desarrollo. Los elementos que han influido en la planificación incluyen:

Investigación Preliminar: Del 12 de septiembre al 26 de septiembre se llevó a cabo una investigación preliminar sobre las botnets. Esto permitirá agilizar las tareas de investigación.

Fechas de Entrega: La planificación temporal debe adaptarse a las fechas de entrega parcial de la asignatura:

- PEC1: Entrega el 11 de octubre. Se llevará a cabo el plan de trabajo. (50h de dedicación)
- PEC2: Entrega el 7 de noviembre. Se llevará a cabo el estudio de mercado y el diseño de la aplicación. (75h de dedicación).

- PEC3: Entrega el 5 de diciembre. Describir la implementación de la aplicación y las pruebas realizadas. (75h de dedicación).
- PEC4: Entrega el 9 de enero. Se llevará a cabo la entrega de la memoria final y el producto resultante. (75h de dedicación).
- PEC5: Entrega el 16 de enero. Se llevará a cabo la entrega de la presentación. (25h de dedicación).
- PEC6: Entrega el 26 de enero. Se llevará a cabo la defensa ante el jurado. (5h de dedicación).

Tiempo estimado PECs: Para realizar la planificación se ha tenido en cuenta el tiempo estimado de realización de cada PEC en el enunciado de la actividad, siendo 50h para la PEC1, 75h para la PEC2, PEC3 y PEC4, un tiempo de 20h para la PEC5 y 5h para la PEC6.

Disponibilidad de Material: Se garantiza la disponibilidad continua de los materiales necesarios a lo largo del proyecto.

Disponibilidad Temporal: Se ha asignado un tiempo diario de 4 horas para trabajar en el proyecto, con posibilidad de ser ampliado a 6 horas durante la fase de desarrollo de la botnet. Adicionalmente se reservan 12 días de vacaciones como salvaguardia en caso de posibles retrasos.

Festividades: En los días festivos se dispondrá de tiempo adicional para avanzar el proyecto.

Las festividades contempladas son:

- Jueves 12 de octubre (Fiesta Nacional de España)
- Miércoles 1 de noviembre (Todos los Santos)
- Miércoles 6 de diciembre (Día de la Constitución Española)
- Viernes 8 de diciembre (Inmaculada Concepción)
- Lunes 25 de diciembre (Navidad)
- 1 de enero (Año Nuevo)
- 6 de enero (Día de Reyes)

Experiencia en Malware: La experiencia previa en temas de malware permitirá acelerar las fases de investigación relacionadas con botnets y malware.

Experiencia en Desarrollo de C++: Dada la experiencia en desarrollo de C++, se podrá recortar tiempo en los desarrollos.

Pruebas de regresión de la botnet: Dado la complejidad que supone probar la botnet en un entorno con varias máquinas es posible que surjan imprevistos. Es por esto necesario dar un tiempo extra prudencial a esta tarea.

Flexibilidad temporal para el análisis de las botnets y malware: Dado que esta parte no es necesario realizarla hasta la entrega final, puede empezar a realizarse tras la recopilación de información.

Redacción: Se ha proporcionado margen de tiempo adicional en las tareas que involucren redacción de contenidos debido a la falta de experiencia previa en este campo.

Teniendo en cuenta lo expuesto anteriormente, se propone la siguiente planificación temporal:

Name	Begin date	End date	Durati...
Planificación TFM	9/27/23	1/26/24	122
PEC1	9/27/23	10/10/23	14
Plan de Trabajo	9/27/23	10/10/23	14
PEC2	10/11/23	11/6/23	27
Estudio de Mercado	10/11/23	10/20/23	10
Diseño de la Botnet	10/21/23	11/6/23	17
PEC3	11/7/23	12/5/23	29
Implementación de la Botnet	11/7/23	11/27/23	21
Pruebas Unitarias	11/7/23	11/27/23	21
Pruebas de Regresión	11/28/23	11/30/23	3
Documentación del Desarrollo	12/1/23	12/3/23	3
Documentación de Pruebas	12/4/23	12/5/23	2
PEC4	12/6/23	1/9/24	35
Recopilación de Información	12/6/23	12/14/23	9
Lectura, Resumen y Organización del Material	12/15/23	12/22/23	8
Estudio Teórico Botnets	12/23/23	12/31/23	9
Estudio Teórico malware	1/1/24	1/2/24	2
Redacción de la memoria final	1/3/24	1/7/24	5
Revisión y edición	1/8/24	1/9/24	2
PEC5	1/10/24	1/16/24	7
Preparación de la Presentación	1/10/24	1/16/24	7
PEC6	1/17/24	1/26/24	10
Preparación de la Defensa	1/17/24	1/26/24	10

Figure 1: Planificación temporal

Puede consultar el Diagrama de Gantt en el el siguiente enlace:

Anexo C: Diagrama de Gantt

1.9. Organización de la memoria

Capítulo 1: Introducción y Planificación Presenta los antecedentes y justificación del trabajo, objetivos, enfoque metodológico, planificación temporal y descripción de contenidos.

Capítulo 2: Contexto y Estado del Arte. Analiza el contexto mediante una revisión del estado del arte en temas relacionados.

Capítulo 3: Fundamentos Teóricos. Recopila los conceptos teóricos necesarios sobre botnets y malware para el desarrollo del proyecto.

Capítulo 4: Diseño. Detalla los elementos de diseño: solución propuesta, elección de tecnologías, arquitectura, algoritmos y componentes.

Capítulo 5: Implementación. Documenta el proceso de construcción e integración del sistema.

Capítulo 6: Resultados y Evaluación. Presenta las pruebas realizadas y evalúa los resultados obtenidos.

Capítulo 7: Conclusiones y Trabajo Futuro. Sintetiza conclusiones y plantea posibles mejoras y líneas de continuación.

2. Contexto y Revisión del Estado del Arte

2.1. Desglose del problema

Aunque ya hemos explorado la problemática de las botnets en el planteamiento inicial, es esencial desglosar este desafío específico. Mi proyecto se centra en la creación de una botnet P2P diseñada para el robo de criptomonedas. Este problema se divide en tres componentes interconectados:

Botnets: Las botnets representan una amenaza significativa en el ciberespacio. Estas redes son peligrosas por varias razones. En primer lugar, pueden llevar a cabo ataques masivos distribuidos de denegación de servicio (DDoS), paralizando sitios web y servicios en línea críticos. Además, las botnets son instrumentos eficaces para el robo masivo de datos.

Uso de P2P en botnets: La implementación de tecnología P2P en botnets complica significativamente su desmantelamiento y la identificación del botmaster. Al utilizar comunicación P2P, las botnets se vuelven altamente descentralizadas y autónomas, lo que dificulta la detección y el seguimiento. Los nodos en una red P2P se comunican directamente entre sí, evitando puntos centrales de control que las autoridades podrían atacar para desmantelar la red.

Robo de criptomonedas: El robo de criptomonedas se ha convertido en un peligro creciente debido a varios factores. En primer lugar, la falta de legislación unificada a nivel internacional permite a los ciberdelincuentes operar en un entorno legalmente ambiguo. Las criptomonedas, siendo descentralizadas y pseudoanónimas, facilitan el robo sin dejar un rastro claro. Además, los delincuentes pueden utilizar servicios de mezcla (mixers) para ocultar la ruta de las transacciones, lo que dificulta aún más el seguimiento del dinero robado.

2.2. Posibles soluciones al problema

Aunque la elección final de la temática del proyecto se centra en el desarrollo de una botnet educativa, es relevante nombrar que enfoques de proyecto alternativos se pueden llevar a cabo para solucionar el problema:

Investigación de Técnicas Actuales de Desarrollo: Explorar técnicas de desarrollo modernas ayuda a entender las últimas tendencias en la creación de botnets, permitiendo prepararse mejor contra amenazas emergentes.

Medidas de Mitigación y Protección: Este enfoque analiza tecnologías y prácticas de seguridad existentes para protegerse contra botnets.

Simuladores de Botnets: Estos entornos controlados permiten modelar ataques de botnets y evaluar estrategias de seguridad.

Desarrollo de Botnet Educativa: Crear una botnet educativa para concienciar sobre amenazas y prácticas de seguridad puede ser una herramienta poderosa.

Sistema de Protección para Carteras Criptográficas: Investigar medidas de seguridad para proteger las carteras criptográficas es crucial para prevenir el robo de criptomonedas.

2.3. Temas de estudio

El estado del arte para este proyecto se centrará en tres áreas principales: botnets educativas, uso de P2P en botnets y robo de criptomonedas mediante malware. Estos temas son relevantes debido a:

Botnets educativas

El desarrollo de una botnet con fines formativos es una parte central del proyecto. Analizar trabajos previos sobre botnets educativas aportará conocimientos valiosos en aspectos como:

- Objetivos y motivaciones para desarrollar este tipo de amenazas y simulaciones.
- Arquitecturas y comportamientos implementados en otros proyectos educativos.
- Resultados y conclusiones derivadas de experiencias prácticas con botnets controladas.

Uso de P2P en botnets

Dado que la botnet a desarrollar utilizará un protocolo P2P, es esencial investigar en profundidad el uso específico de P2P en el ámbito de las botnets:

- Ejemplos de botnets P2P e implementaciones o propuestas innovadoras de uso de P2P en botnets en el ámbito académico
- Ventajas que provee frente a arquitecturas centralizadas: dificultad de desmantelamiento, anonimato, etc.

Robo de criptomonedas mediante malware

Al estar orientada específicamente al robo de criptomonedas, es necesario investigar:

- Técnicas actuales empleadas para el robo de criptoactivos mediante malware.
- Ejemplos de malware que emplee estas técnicas.

2.4. Criterios de inclusión

Relevancia Temática: La fuente está relacionada con los termas que se tratarán.

Actualidad: Se priorizan fuentes de los últimos años
Credibilidad y Fiabilidad: Se priorizan fuentes publicadas en revistas científicas, conferencias reconocidas o editoriales académicas y se priorizan fuentes revisadas y validadas por expertos en el campo.

Metodología: La fuente utiliza metodologías sólidas y fiables en su investigación.

Contribución: La fuente aporta nuevos conocimientos, descubrimientos o enfoques innovadores y ofrece análisis detallados conclusiones que enriquecen el estudio.

2.5. Fuentes de investigación

En esta investigación, se consultan diversas fuentes de información para construir una base sólida de conocimientos. Las principales fuentes utilizadas incluyen:

Google Académico: A través de este buscador en línea, se recuperan textos especializados de diversas áreas mediante búsquedas avanzadas.

Páginas web de Bibliotecas Universitarias: Se exploran las bibliotecas de instituciones académicas nacionales e internacionales.

IEEE Xplore: Este recurso ofrece una amplia gama de documentos académicos en el campo de la ingeniería y la tecnología.

Repositorios Institucionales: Se utilizan repositorios institucionales de diversas universidades y centros de investigación para acceder a trabajos académicos y documentos relevantes.

Repositorios Cooperativos: Se consultan repositorios cooperativos, como Tesis Doctorales en Red y RECERCAT, que recopilan una variedad de documentos de diferentes instituciones.

2.6. Síntesis del estado del arte

2.6.1. Botnets educativas P2P

"A Study on Social Network based P2P Botnet"[4]

Este estudio presenta un modelo novedoso de botnet que utiliza microblogs como servidores C&C, combinando características de redes sociales con redes P2P. La investigación detalla el análisis de control y comando, la estructura y el cifrado de comandos. Se introduce un algoritmo dinámico para generar microblogs y un método sólido de codificación para integrarlos en los nodos zombi. Los resultados destacan la eficiencia y robustez de esta botnet, aunque señalan que su alcance puede aumentar el tráfico y la detección debido a la relación entre eficiencia y alcance, y la cantidad de datos durante la transmisión de comandos.

“A Reputation-Based Resilient and Recoverable P2P Botnet”[5]

Este documento describe TRBot, una botnet P2P diseñada para abordar las vulnerabilidades de las botnets híbridas frente a los ataques de contaminación de listas de pares. Destaca por su arquitectura basada en la reputación, que asegura la confianza entre bots y su capacidad de autorreparación. El estudio detalla su diseño, enfocándose en un procedimiento de inicio de dos pasos, un mecanismo de construcción de lista de pares

basado en la reputación y un sistema de autorreparación. Aunque innovador, el documento carece de datos empíricos y evaluaciones comparativas con otras botnets.

2.6.2. Botnets educativas blokchain

"ZombieCoin3.0: On the Looming of a Novel Botnet Fortified by Distributed Ledger Technology and Internet of Things"[6]

El estudio presenta ZombieCoin3.0, una infraestructura de botnet que utiliza Distributed Ledger Technology (DLT) y dispositivos IoT para mayor resiliencia y anonimato del Botmaster. Elimina la dependencia de listas de vecinos, enfocándose en transferencia de archivos y retroalimentación de datos en C&C. Fusiona botnet P2P con DLT, usando el protocolo Gossip en el DLT IOTA's Tangle, con canales CDC y DUC.

Los resultados demuestran mayor eficiencia en la comunicación con baja latencia y costos mínimos en comparación con Bitcoin y Ethereum. Destaca en la gestión de transferencia de archivos y retroalimentación de datos. A pesar de sus logros, se enfatiza la necesidad de medidas de seguridad y privacidad, especialmente al usar dispositivos IoT como mantenedores.

"Botract: abusing smart contracts and blockchain"[7]

El artículo presenta Botract, una botnet innovadora que utiliza contratos inteligentes y blockchain para C&C. Enfoca la lógica C&C en la blockchain Ethereum para crear conciencia sobre la necesidad de auditar y defenderse contra estas botnets emergentes.

La metodología involucra una prueba de concepto en entornos de prueba en la red Ethereum. Se analizan ventajas, desafíos y las implicaciones de seguridad y privacidad al desplegar botnets basadas en blockchain.

Los resultados confirman la viabilidad del enfoque mediante la implementación exitosa de la prueba de concepto. Se identifica una limitación relacionada con el almacenamiento necesario para descargar y mantener la blockchain, restringiendo su implementación en dispositivos con recursos limitados.

"D-LNBot: An In-Depth Analysis of a Scalable, Cost-Free, and Covert Hybrid Botnet on Bitcoin's Lightning Network"[8]

El artículo introduce D-LNBot, una botnet híbrida que utiliza la Lightning Network de Bitcoin para comunicaciones encubiertas. Su arquitectura de dos capas envía comandos del botmaster a través de pagos, transmitidos a los bots por servidores C&C.

El objetivo es proponer una nueva generación de botnets eficientes en la Lightning Network, analizando su viabilidad y escalabilidad en términos de retraso y costo.

La metodología incluye el diseño e implementación en la red Lightning Network con nodos reales en la red de pruebas de Bitcoin. Se realizaron pruebas de concepto y análisis de rendimiento, discutiendo contramedidas para detectar y minimizar el impacto de D-LNBot.

Los resultados validan la viabilidad y eficacia de D-LNBot en la red Lightning, demostrando la operación gratuita y la propagación eficiente de comandos. Sin embargo, se limita a pruebas, siendo necesario más análisis para su implementación real.

2.6.3. Simuladores botnet P2P

"An Overview of the Botnet Simulation Framework"[9]

El artículo introduce el Botnet Simulation Framework (BSF), una herramienta implementada en OMNeT++ que simula botnets P2P en diversos escenarios para evaluar estrategias de monitoreo y mitigación.

El objetivo principal es presentar y demostrar la utilidad de BSF en la simulación de botnets y evaluación de mecanismos de detección. La metodología detalla los componentes clave del framework y explora la generación de datos de detección y la inyección de tráfico de botnets en archivos PCAP existentes.

BSF destaca por su versatilidad para simular botnets P2P con configuraciones flexibles. Puede manejar grandes botnets y exportar comunicaciones simuladas para inyectar tráfico en capturas PCAP reales mediante ID2T.

A pesar de su potencial, BSF tiene limitaciones como la configuración manual de parámetros y demanda de recursos para simular grandes botnets y generar conjuntos de datos extensos.

"BotsideP2P: A Peer-to-Peer Botnet Testbed"[10]

El estudio presenta BotsideP2P, un entorno de laboratorio especializado para botnets P2P que emplea una tabla hash distribuida (DHT) basada en Kademlia DHT para encriptar la comunicación entre bots. Detalla la estructura de la botnet y los procedimientos de registro.

Los objetivos incluyen proporcionar un entorno de prueba especializado, fomentar la comprensión de las botnets P2P y servir como herramienta educativa. La metodología implica la implementación de la botnet P2P en BotsideP2P usando Kademlia DHT y Asyncio.

Aunque demuestra la implementación, el estudio señala limitaciones en la escalabilidad y desafíos de compatibilidad entre bibliotecas. Este trabajo aporta comprensión sobre botnets P2P y ofrece un enfoque alternativo para la simulación y monitorización de amenazas.

2.6.4. Botnets basadas en IA

“DeepC2: AI-powered Covert Command and Control on OSNs”[1]

El estudio presenta DeepC2, una botnet que emplea inteligencia artificial (IA) para su sistema de comando y control (C&C) en redes sociales.

Su objetivo principal es desarrollar un sistema de C&C encubierto en redes sociales mediante IA, superando la reversibilidad y detección de comandos anormales. La metodología implica entrenar una red neuronal para extraer características de avatares del atacante, identificándolo encubiertamente, y el uso de colisiones de hash y tweets embebidos para evitar la detección.

Los experimentos en Twitter validan la efectividad, mostrando cómo evita la reversibilidad al identificar al atacante y evitar la detección de comandos anormales. No obstante, depende de las redes sociales y se limita a Twitter, lo que puede afectar su aplicación en otras

plataformas y su vulnerabilidad a medidas de seguridad más estrictas.

2.6.5. Malware educativo robo criptomonedas

No se han encontrado trabajos de investigación en los que se lleve a cabo malware educativo o simuladores enfocados en el robo de criptomonedas.. No obstante, el robo de criptomonedas es una amenaza real y se pueden encontrar noticias de malware que lleva a cabo estos ataques.[11]

2.7. Conclusiones estado del arte

El uso de P2P en botnets está muy extendido por las ventajas que ofrecen, como la resiliencia y la preservación del anonimato del botmaster, en comparación con las botnets centralizadas. Tal y como se ha podido observar en el estado del arte existen lineas de investigación que buscan nuevas posibles formas de implementar botnets P2P, así como arquitecturas híbridas que intentan unir lo mejor de las arquitecturas centralizadas y descenralizadas, como es el caso del malware que opera en las redes blockchain.

El uso de P2P en botnets lleva siendo habitual desde hace bastantes años. Para que el proyecto aporte valor de investigación es necesario que el protocolo P2P usado aporte alguna innovación respecto a los usados por las botnets actuales, dando a conocer así nuevas posibles amenazas y concienciando sobre como con pequeños cambios de implementación se pueden hacer botnets con características diferentes que pueden dificultar su detección

En un principio se pensó en la posibilidad de diseñar una botnet que opere en la red blockchain, por el foco mediático en las criptomonedas y porque puede ocultarse más fácilmente el robo de las criptocarteras que haga la botnet y usar los comandos/respuestas con el botmaster para transferir el dinero, pero tal y como se ha podido observar en el estado del arte existen muchos trabajos que tratan el tema de los sistemas de C&C usando blockchain.

Aunque se conoce de la existencia de malware orientado al robo de criptomonedas no hay estudios de investigación que implementen pruebas de concepto ni simulaciones de estos ataques. Es por lo tanto una linea de investigación que merece la pena explorar.

3. Marco teórico

3.1. Botnets

3.1.1. Definición de Botnet

Una red botnet es un conjunto de dispositivos informáticos infectados con malware, controlados remotamente por un atacante de manera coordinada y sin el consentimiento de los usuarios, con el fin de llevar a cabo actividades ilícitas [1].

Las botnet están compuestas por una extensa lista de computadoras conectadas a Internet desde diferentes partes del mundo (bots) que son controlados por un cibercriminal (botmaster) a través de un sistema de Comando y Control (C&C).

3.1.2. Características de las Botnet

Red de Dispositivos Coordinados: Las botnets se destacan por su capacidad para coordinar y controlar una red extensa de dispositivos comprometidos, permitiendo acciones concertadas y eficientes por parte del atacante.

Modelo de Control: El modelo de control de una botnet varía, siendo común el uso de un servidor C&C centralizado. Sin embargo, algunas botnets adoptan enfoques descentralizados, lo que mejora la resiliencia al eliminar la dependencia de un único punto de control.

Capacidad de Escala: La capacidad de las botnets para adaptarse y expandirse eficientemente les permite incorporar nuevos dispositivos a su red, incrementando su capacidad para llevar a cabo acciones maliciosas.

Diversidad de Acciones Coordinadas: Las botnets son versátiles y pueden ser utilizadas para una amplia variedad de acciones coordinadas, desde ataques DDoS hasta la propagación de malware secundario, adaptándose a los objetivos específicos del atacante.

Persistencia y Adaptabilidad: Las botnets son altamente persistentes, resistiendo intentos de eliminación y adaptándose a cambios en su entorno digital mediante el uso de técnicas de ocultamiento y evasión.

Capacidad de Auto organización: Algunas botnets exhiben características de autoorganización, lo que les permite reorganizarse y adaptarse a cambios en la red, aumentando su resiliencia.

Utilización de Redes de Bots: La fuerza impulsora de las botnets es la red de bots que trabajan conjuntamente, permitiendo acciones coordinadas maliciosas a gran escala.

3.1.3. Componentes de una Botnet

Botmaster: El botmaster, también conocido como operador o controlador de la botnet, desempeña un papel crucial en la orquestación y dirección de las actividades. Este individuo o grupo ejerce control sobre la botnet, emitiendo comandos y estrategias para alcanzar objetivos específicos. La relación del botmaster con la botnet puede ser directa o indirecta, y su identidad a menudo se oculta detrás de diversas capas de anonimato.

Servidor de Comando y Control (C&C): El servidor de comando y control actúa como el punto central de coordinación. Los bots infectados se comunican con este servidor para recibir instrucciones y enviar información sobre sus actividades. El C&C proporciona al botmaster una interfaz centralizada para gestionar la botnet, permitiendo la emisión de comandos, actualizaciones y la recopilación de datos.

Bots (Ordenadores Infectados): Los bots representan los nodos comprometidos dentro de la botnet. Estos dispositivos pueden incluir computadoras personales, servidores o dispositivos conectados a la red. La infección de los bots se lleva a cabo mediante la introducción de malware diseñado para operar de manera encubierta. Los bots permiten al botmaster controlar remotamente las acciones de estos dispositivos.

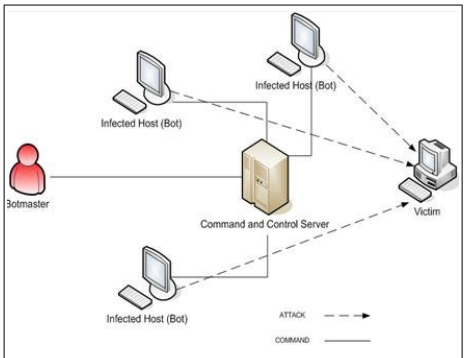

Figure 2: componentes botnetma[25]

3.1.4. Arquitectura de las Botnets

Centralizadas: En esta configuración, los nodos infectados se conectan directamente a un C&C único, que ejerce control sobre la red y recopila datos. Esto ofrece un control directo y comunicación rápida entre los nodos y el C&C. Sin embargo, esta estructura es vulnerable, ya que la desactivación del C&C central puede inutilizar toda la botnet, y es más detectable debido a su centralización.


Figure 3: C&C centralizado[23]

Descentralizadas (P2P): En este modelo, los nodos se conectan directamente entre sí, sin un servidor de comando central. Cada nodo actúa como cliente y servidor simultáneamente. Esta configuración ofrece resiliencia, ya que carece de un punto central de control, facilita la autoorganización y adaptación a cambios. No obstante, su gestión se vuelve más compleja al carecer de un punto central, y coordinar acciones entre los nodos puede resultar desafiante.

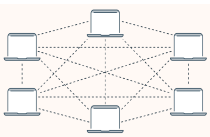
Figure 4: C&C descentralizado[23]

3.1.5. Tipos de botnets

Botnets HTTP: Las botnets HTTP se valen del protocolo de transferencia de hipertexto para establecer comunicación entre los nodos y el C&C. Este enfoque a menudo utiliza el tráfico web convencional, dificultando su detección.

Botnets IRC: Las botnets IRC utilizan canales de chat para la coordinación y comunicación entre nodos. Esta arquitectura se ha utilizado históricamente y proporciona un medio efectivo para que los bots reciban comandos y compartan información.

Botnets POP3: Las botnets POP3 usan correos electrónicos para coordinar acciones maliciosas, a menudo aprovechando la apariencia legítima del tráfico de correo.

Botnets P2P: Estas botnets basadas en protocolos P2P aprovechan redes descentralizadas, para facilitar la comunicación y coordinación entre nodos. Eliminan la dependencia de un servidor central, mejorando la resistencia y dificultando la detección.

Botnets en Redes Sociales: Las botnets que operan en plataformas de redes sociales utilizan perfiles comprometidos para propagar malware, difundir enlaces maliciosos o llevar a cabo actividades de ingeniería social.

Botnets Internet of Things (IoT): Las botnets IoT se enfocan en dispositivos conectados a Internet, como cámaras de seguridad y electrodomésticos inteligentes. La falta de seguridad en muchos de estos dispositivos los hace susceptibles a infecciones.

Botnets Blockchain: Las botnets basadas en blockchain aprovechan la descentralización inherente a esta tecnología para dificultar la identificación y neutralización de nodos. Pueden utilizar contratos inteligentes para coordinar actividades maliciosas.[7]

Botnets en dispositivos móviles: Las botnets que se basan en mensajes multimedia (MMS) o mensajes de texto (SMS) para la coordinación y propagación. Pueden aprovechar vulnerabilidades en dispositivos móviles y redes para llevar a cabo acciones maliciosas.

Botnets de máquinas virtuales y servidores: Botnets como "Mantis" emplean máquinas virtuales y servidores potentes, otorgando a cada bot una gran cantidad de recursos computacionales. Estas botnets pueden llevar a cabo ataques masivos a pesar de contar una cantidad limitada de bots.[12]

3.1.6. Vectores de infección de las botnets

Descarga de Software Malicioso: Los usuarios, al descargar e instalar software desde fuentes no seguras, como programas pirateados o versiones comprometidas, pueden introducir malware en sus sistemas inadvertidamente, facilitando la formación de una botnet.

Emails con Adjuntos Maliciosos: El envío de correos electrónicos con archivos adjuntos maliciosos es una táctica común. Los usuarios que abren estos archivos pueden desencadenar la descarga e instalación de malware, propiciando la entrada de sistemas a la red de una botnet.

Explotación de Vulnerabilidades en Sitios Web (Exploit Kits): Mediante el aprovechamiento de vulnerabilidades presentes en sitios web, las botnets utilizan exploit kits

para inyectar código malicioso en los dispositivos de los usuarios que visitan estas páginas comprometidas.

Vulnerabilidades en el Sistema Operativo o Programas: La explotación de vulnerabilidades conocidas permite que las botnets se infiltren en sistemas que no han aplicado las debidas actualizaciones de seguridad.

Dispositivos Extraíbles Infectados: Los dispositivos extraíbles pueden actuar como portadores de malware.

Dispositivos con Contraseñas por Defecto: La utilización de dispositivos con contraseñas predeterminadas o débiles proporciona una puerta de entrada para las botnets.

Anuncios Maliciosos: Los anuncios maliciosos, pueden llevar a la descarga de malware cuando los usuarios hacen clic en banners engañosos o anuncios comprometidos mientras navegan por la web.

Ataques de Ingeniería Social: Las botnets pueden emplear tácticas de ingeniería social para manipular a los usuarios y hacer que realicen acciones que faciliten la infección, como descargar ficheros maliciosos

3.1.7. Usos criminales de las Botnets

Ataques Distribuidos de Denegación de Servicio (DdoS): Este ataque apunta a abrumar un servicio en línea al inundarlo con tráfico malicioso desde una red distribuida de dispositivos, causando la interrupción del servicio.

Extorsión: Las botnets a veces se utilizan para llevar a cabo campañas de extorsión, amenazando con revelar información confidencial o lanzar ataques DDoS a menos que se pague un rescate.

Robo de Datos: Las botnets pueden ser empleadas para recopilar datos sensibles, como información personal, financiera o empresarial, con el objetivo de utilizarla de manera fraudulenta o con fines lucrativos.

Phishing: Un método de engaño en línea donde las botnets se utilizan para enviar correos electrónicos falsos o mensajes diseñados para engañar a las personas y obtener información confidencial, como nombres de usuario y contraseñas.

Keylogging: Las botnets pueden emplear keyloggers para registrar de manera clandestina las pulsaciones de teclas en dispositivos comprometidos, capturando información confidencial como contraseñas y datos de acceso.

SPAM: Utilizando una red de dispositivos comprometidos, las botnets pueden inundar correos electrónicos no deseados o mensajes en foros y redes sociales, promoviendo productos, servicios o contenido malicioso.

Click Fraud: Las botnets pueden generar clics falsos en anuncios en línea para inflar artificialmente los ingresos publicitarios o perjudicar a competidores.

Fuerza Bruta: Se emplea para descifrar contraseñas mediante la prueba de combinaciones

hasta encontrar la correcta, otorgando acceso no autorizado a sistemas protegidos.

Minado de Criptomonedas: Las botnets pueden aprovechar la potencia de procesamiento de dispositivos infectados para minar criptomonedas de manera ilícita.

Ransomware: Las botnets distribuyen malware que cifra archivos en dispositivos comprometidos, exigiendo un rescate para su restauración.

Influencia Mediática[2]: Se utiliza para difundir desinformación, influenciar opiniones políticas o sociales y manipular la percepción pública a través de campañas en redes sociales.

Descarga de Programas Maliciosos: Las botnets pueden distribuir y propagar malware, incluyendo virus, gusanos u otros programas maliciosos, afectando la integridad de sistemas y datos.

Ataques de Ingeniería Social: Las botnets pueden usar tácticas como distribuir antivirus falsos que muestran alertas de seguridad ficticias para engañar a las víctimas y hacer que paguen por servicios fraudulentos

3.1.8. Técnicas de ocultamiento de botnets

Uso de Redes Anónimas: Al dirigir la comunicación a través de redes anónimas, como Tor, las botnets pueden ocultar la ubicación real de los servidores C&C y dificultar el rastreo de la actividad maliciosa.

Uso de Protocolos Encapsulados: Las botnets pueden emplear protocolos de comunicación encapsulados, como el uso de SSL/TLS, para cifrar el tráfico y dificultar la detección por parte de sistemas de seguridad.

Uso de Dominios Dinámicos o Fast-Flux[13]: Las botnets pueden cambiar dinámicamente los registros DNS asociados con los nodos de la red, dificultando la tarea de bloquear o rastrear la infraestructura maliciosa.

Comunicación P2P (Peer-to-Peer): La comunicación entre nodos de la botnet a través de redes P2P en lugar de depender de un servidor centralizado ayuda a distribuir la responsabilidad y dificulta el seguimiento.

Uso de Canales Encubiertos: Las botnets pueden utilizar protocolos y servicios legítimos para pasar desapercibidas.

Técnicas de Robo de Dominio: Incluye estrategias para adquirir el control de dominios legítimos, como el secuestro de cuentas de registradores contra propietarios de dominios.

Cloud Botnets[14]: Los controladores de botnets se disfrazan como usuarios legítimos de servicios en la nube, utilizando máquinas virtuales proporcionadas por los proveedores de servicios en la nube.

3.1.9. Métodos de detección de Botnets

Análisis de Comportamiento de Red: Monitorización del tráfico de red para identificar patrones de comportamiento asociados con la actividad de botnets, como la comunicación frecuente con servidores C&C, tráfico no habitual, o la presencia de comandos maliciosos.

Análisis de Tráfico P2P: Identificación de patrones de tráfico característicos de las redes peer-to-peer, comunes en botnets descentralizadas.

Detección basada en firmas[15]: Utiliza patrones de tráfico conocidos y firmas de bots existentes para su detección.

Detección basada en minería de datos[16]: Aplica técnicas de machine learning y clustering para identificar patrones de tráfico C&C.

Análisis de Comportamiento del Sistema: Monitoreo del comportamiento de los sistemas para detectar actividades sospechosas, como la ejecución de comandos remotos o la instalación no autorizada de software.

Detección de Patrones de Ataque: Identificación de patrones de ataque específicos asociados con botnets, como intentos de fuerza bruta o ataques DDoS coordinados.

HoneyPots[17]: Los honeypots son sistemas aislados que simulan ser máquinas vulnerables para atraer atacantes y analizar su comportamiento.

Detección basada en SPAM[16]: Analiza patrones en emails para identificar envíos automatizados de bots spammers.

Análisis de Redes Sociales: Monitorización de redes sociales y foros en busca de discusiones o actividades relacionadas con la coordinación de botnets.

Detección de Infraestructura de Dominio y C&C: Identificación de dominios y servidores utilizados como puntos de control y comando (C&C) por botnets.

Monitoreo de nicknames IRC[15]: Analiza los nicknames utilizados por los bots en los canales IRC para detectar similitudes que indiquen la presencia de una botnet.

Aprendizaje automático: Aplica algoritmos de aprendizaje automático para distinguir tráfico legítimo y de botnets.

Extracción de grafos de relaciones[16]: Crea grafos de relaciones entre hosts infectados observando patrones similares de comunicación y actividad maliciosa para identificar posibles bots dentro de una misma botnet.

Análisis Forense de Botnets: Realizar análisis forense en sistemas comprometidos para recopilar evidencia contra botnets y entender su funcionamiento interno.

3.1.10. Métodos de mitigación de botnets

Implementación de Firewalls y Filtros de Paquetes: Configurar firewalls y filtros de paquetes para bloquear el tráfico malicioso conocido asociado con botnets.

Actualización Regular de Software: Mantener el software actualizado, incluyendo sistemas operativos, aplicaciones y programas de seguridad, para corregir vulnerabilidades conocidas.

Uso de Sistemas de Detección de Intrusiones (IDS) e Intrusion Prevention Systems (IPS): Implementar soluciones IDS e IPS para detectar y prevenir actividades sospechosas asociadas con botnets.

Filtrado de Tráfico DNS: Utilizar sistemas de filtrado de tráfico DNS para bloquear consultas a dominios maliciosos asociados con botnets.

Endurecimiento de Configuraciones de Seguridad: Aplicar configuraciones de seguridad sólidas en servidores y dispositivos para reducir la superficie de ataque.

Educación y Concienciación del Usuario: Proporcionar capacitación regular a los usuarios sobre prácticas de seguridad en línea y cómo identificar posibles amenazas.

Software Antimalware: Utilizar software antimalware actualizado para detectar y eliminar malware, incluyendo componentes de botnets. Estos programas pueden escanear sistemas en busca de firmas conocidas, comportamientos maliciosos y otros indicadores de infección.

3.1.11. Métodos de desmantelamiento de botnets

Poisoning de reconocimiento[13]: Envenena las consultas que realizan los bots a listas negras DNS (DNSBL) durante la fase de reconocimiento previa a un ataque, provocando respuestas falsas que desactivan la botnet [49].

Colaboración con Proveedores de Servicios de Internet: Trabajar en estrecha colaboración con los ISP para identificar y desconectar servidores C&C y nodos infectados.

Coordinación con Organizaciones de Seguridad: Colaborar con organizaciones de seguridad y agencias gubernamentales para compartir inteligencia y coordinar acciones contra botnets.

Desactivación de Dominios C&C: Colaborar con registradores de dominios y autoridades para desactivar dominios utilizados como servidores C&C.

Sinkholing[18]: Esta técnica implica redirigir el tráfico de la botnet a servidores controlados por investigadores o entidades de seguridad. Esto puede ayudar a monitorear la actividad de la botnet, recopilar información sobre dispositivos infectados y prevenir la comunicación entre los bots y los servidores C&C originales.

Ataques de denegación de servicio (DDoS): En algunos casos, se pueden emplear ataques DDoS contra los servidores C&C de la botnet para sobrecargarlos, interrumpir su funcionamiento y dificultar su capacidad para controlar los dispositivos infectados.

Infiltración y engaño: En algunos casos, los expertos en seguridad pueden infiltrarse en la botnet, haciéndose pasar por parte de ella para obtener información sobre su funcionamiento interno y, potencialmente, desbaratar su estructura desde adentro.

Interrupción de la infraestructura financiera: En algunos casos, se puede trabajar para

interrumpir la infraestructura financiera que respalda la operación de la botnet, como el bloqueo de cuentas bancarias utilizadas por los operadores.

Investigación y seguimiento de Bitcoin y criptomonedas: Dado que las transacciones de criptomonedas pueden utilizarse para financiar operaciones de botnets, el seguimiento de las transacciones en blockchain puede proporcionar pistas sobre los operadores y su financiamiento.

3.2. Malware

3.2.1. Métodos de ocultamiento de malware

Anti-Análisis: Implementa técnicas para detectar si el malware está siendo ejecutado en un entorno de análisis, alterando su comportamiento o permaneciendo inactivo para evadir la detección.

Polimorfismo: Cambia la apariencia del código malicioso sin modificar su funcionalidad básica, generando variantes únicas que dificultan la detección basada en firmas.

Metamorfismo: Reorganiza la estructura interna del código malicioso, más avanzado que el polimorfismo, complicando aún más la detección por parte de soluciones basadas en firmas.

Uso de Empaquetadores y Compresores: Altera la apariencia del malware mediante técnicas de compresión y empaquetado, dificultando la detección por parte de los motores antivirus.

Ofuscación de Código: Modifica el código fuente o de máquina del malware para hacerlo más difícil de entender, incluyendo cambios en nombres de variables y funciones, así como la introducción de instrucciones redundantes.

Uso de Rootkits: Oculta la presencia del malware en el sistema, interceptando llamadas al sistema y modificando datos para mantener la persistencia y dificultar la detección.

Detección de Emulación: Incluye rutinas que detectan si el malware se está ejecutando en un entorno emulado, haciendo que el malware se comporte de manera diferente para evitar la detección.

Detección de Entorno de Sandbox: Incorpora rutinas que detectan la ejecución en un entorno de sandbox o de análisis, modificando su comportamiento para parecer inofensivo durante la evaluación de seguridad.

Inyección en Procesos Legítimos: Inserta su código malicioso en procesos legítimos del sistema, lo que dificulta su identificación, ya que parece ser parte de un proceso confiable.

Uso de Cifrado: Encripta partes del código malicioso o las comunicaciones para dificultar la inspección y la detección basada en patrones conocidos.

3.2.2. Métodos de detección de malware

Análisis de Firmas: Utiliza bases de datos de firmas conocidas de malware para identificar patrones específicos en archivos o procesos. Esta técnica se basa en comparar el código con firmas previamente identificadas.

Análisis de Comportamiento: Monitoriza continuamente el comportamiento de los procesos en ejecución para detectar actividades maliciosas, como cambios no autorizados en archivos o comunicación sospechosa.

Análisis de Tráfico de Red: Examina el tráfico de red en busca de patrones asociados con la transmisión de malware, como la propagación de malware a través de la red.

Emulación y Sandboxing: Ejecuta archivos sospechosos en un entorno aislado para observar su comportamiento sin afectar el sistema principal. Esto permite detectar acciones maliciosas sin comprometer la seguridad del sistema.

Análisis de Entropía: Mide la complejidad de la información en un archivo. Los archivos maliciosos a menudo tienen patrones de entropía diferentes a los archivos legítimos, lo que puede indicar su naturaleza maliciosa.

Análisis de Firmas: Examina patrones de código de malware conocidos.

Detección de Rootkits: Identifica la presencia de rootkits, que son herramientas que ocultan la actividad maliciosa al modificar o sustituir partes fundamentales del sistema operativo.

Monitorización de Cambios en el Registro del Sistema: Vigila cambios en el registro del sistema que podrían ser indicativos de una infección por malware.

3.2.3. Inyección de procesos

La inyección de procesos es una técnica que implica la inserción o ejecución de código en un proceso en curso. La inyección de procesos se puede lograr mediante varias técnicas, algunas de las cuales son:

Inyección de código: Esta estrategia implica la inserción directa de código, ya sea malicioso o funcional, en la memoria de un proceso en ejecución. Esto se puede realizar aprovechando vulnerabilidades conocidas en el software, técnicas de ingeniería inversa o modificaciones en el espacio de memoria del proceso a través de funciones del sistema operativo.

Inyección de DLL: En este caso, se carga una DLL manipulada o maliciosa en el espacio de memoria de un proceso. Esta técnica es común en ataques de malware, donde una DLL comprometida se introduce en un proceso legítimo para ejecutar acciones no autorizadas.

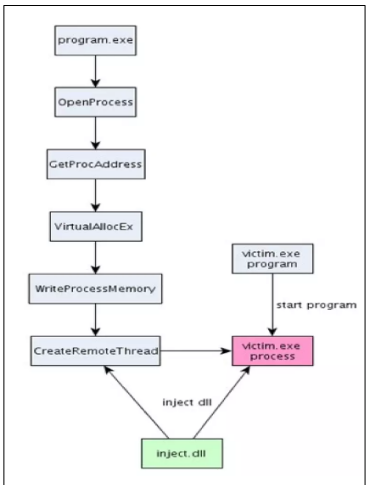

Figure 5: process injection[22]

La inyección de proceso puede tener los siguientes objetivos:

Evadir detección: Al infiltrarse en un proceso legítimo, el malware puede evadir la detección de software de seguridad.

Modificación del comportamiento: La inyección de procesos permite al malware modificar el comportamiento de la aplicación en la que se ha infiltrado. .

Captura de información sensible: Uno de los propósitos principales de la inyección de procesos en el ámbito del malware es la captura de información confidencial.

3.2.4. Persistencia

La persistencia es crucial para que el malware se mantenga activo y operativo en un sistema después de reinicios. Las técnicas de persistencia son variadas y se adaptan para garantizar que el malware permanezca oculto y continúe sus operaciones sin ser detectado fácilmente. Se detallan algunas de las técnicas más comunes utilizadas por el malware para persistir en un sistema:

Registros de inicio y servicios: Muchos tipos de malware se insertan en los registros de inicio del sistema operativo o crean servicios maliciosos que se ejecutan automáticamente durante el inicio. Esto les permite ser activados cada vez que se reinicia el sistema.

Programación de tareas: Algunos malware utilizan la programación de tareas del sistema operativo para ejecutarse en momentos específicos o de manera periódica, permitiéndoles mantenerse activos y realizar acciones sin la necesidad de estar constantemente en ejecución.

Bootkits: Estos son malware que infectan componentes críticos del sistema de arranque, lo que les permite ser activados incluso antes de que el sistema operativo se inicie. Esto hace que sea extremadamente difícil de detectar y eliminar.

Persistencia en el registro y archivos ocultos: Algunos malware modifican claves del registro o crean archivos ocultos en ubicaciones estratégicas del sistema para asegurar su persistencia. Estos archivos o claves pueden ser difíciles de encontrar y eliminar manualmente.

Carpeta de inicio (Startup folder): Los malware pueden colocar accesos directos o archivos ejecutables en las carpetas de inicio del sistema operativo, lo que provoca que se ejecuten al iniciar sesión o al arrancar el sistema. Esto asegura que el malware se active cada vez que un usuario inicie sesión en el sistema.

Reemplazo de DLLs: Algunos malware sustituyen bibliotecas de enlace dinámico (DLLs) legítimas con versiones modificadas o maliciosas. Esto les permite mantener su presencia activa y persistente, ya que las aplicaciones legítimas cargarán la DLL maliciosa cuando sean ejecutadas.

CLSID Hijacking: Aprovechando el registro de clases (CLSID) en Windows, el malware puede reemplazar las rutas de ejecución de los objetos registrados, apuntándolos hacia archivos maliciosos. De esta manera, cuando se invoca una funcionalidad asociada a ese

CLSID, en realidad se ejecuta el malware.

3.2.5. Hookeo de funciones

El hookeo de funciones, conocido como hooking, es una técnica crítica utilizada por el malware y otras aplicaciones para manipular el comportamiento de las funciones del sistema operativo o de las aplicaciones.

En el proceso de hookeo se crea una redirección hacia el código malicioso, el cual al finalizar sus acciones redirige el flujo de ejecución hacia la función original para mantener su funcionalidad intacta. Este puente permite al malware ejecutar operaciones adicionales o maliciosas antes o después de que la función original se ejecute, permitiendo así un control más extenso y discreto sobre las actividades del sistema o de las aplicaciones.

Existen varias tácticas de hookeo. Las más comunes son las siguientes:

IAT Hooking (Import Address Table Hooking): Modifica la Tabla de Direcciones de Importación (IAT) de un programa, reemplazando las direcciones originales de las funciones con direcciones apuntando al código malicioso.

EAT Hooking (Export Address Table Hooking): Similar al IAT hooking, pero en lugar de la IAT, modifica la Tabla de Direcciones de Exportación (EAT) de una DLL para redirigir las llamadas a funciones exportadas hacia el código malicioso.

Hookeo de la tabla de Delay Import: Esta técnica manipula la Tabla de Direcciones de Importación Diferida (Delay Import Table), retrasando la carga de las bibliotecas hasta que sean necesarias, permitiendo redirigir las llamadas de función a código malicioso antes de que se cargue la biblioteca.

Byte Patching: El Byte Patching es una técnica que implica modificar directamente los bytes en el código de un programa. Esta alteración puede abarcar la sustitución de instrucciones específicas por otras, permitiendo redirigir el flujo de ejecución o realizar ajustes en tiempo de ejecución.

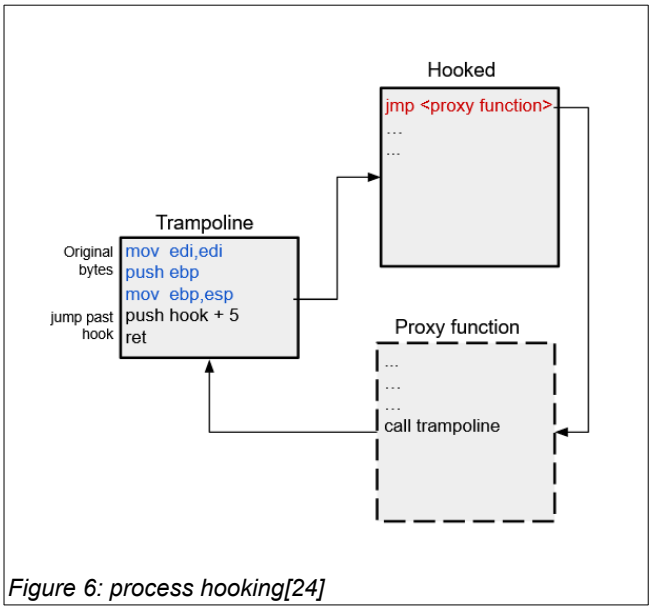

Figure 6: process hooking[24]

4. Fase de diseño

4.1. Solución Planteada

Este proyecto propone una solución innovadora, centrándose en el desarrollo de una botnet educativa. La principal finalidad de esta botnet es permitir un análisis profundo de su funcionamiento interno desde la perspectiva del atacante.

La botnet a desarrollar hará uso del protocolo P2P de Tox y tendrá como finalidad el robo de criptomonedas. La comunicación del botmaster con los bots se realizará mediante un programa de consola. En siguientes apartados se tratan los detalles de la solución.

A continuación, se detallan los objetivos del desarrollo propuesto:

Concienciación del Usuario: Uno de los objetivos fundamentales de este proyecto es la concienciación del usuario en relación con las amenazas cibernéticas. A través de la creación de una botnet educativa, se pretende demostrar que su implementación está al alcance de cualquier persona con conocimientos técnicos de programación y ciberseguridad.

Comprendiendo al Atacante: La botnet también tiene como objetivo proporcionar una visión detallada del proceso de implementación y el funcionamiento de una botnet desde la perspectiva del atacante. Comprender cómo opera un cibercriminal al crear y gestionar una botnet es esencial para desarrollar nuevas soluciones de seguridad y estrategias de mitigación efectivas.

Robo de Criptomonedas: Otro aspecto destacado de esta botnet es su enfoque en el robo de criptomonedas. Este tipo de robo es especialmente crítico debido a la dificultad de rastrear las transacciones de criptomonedas y a la falta de identificación directa de los usuarios, lo que complica aún más la persecución de los ciberdelincuentes. Además, persisten lagunas en la legislación internacional que complican la persecución de estos delitos cibernéticos.

Valor Distintivo: Este proyecto destaca frente a otras investigaciones en varios aspectos clave. En primer lugar, la botnet propuesta adopta un enfoque innovador al utilizar el protocolo P2P Tox para sus comunicaciones. Esta elección añade una capa significativa de complejidad a la amenaza, dificultando considerablemente la identificación del botmaster y la erradicación de la botnet. La selección de un protocolo P2P poco conocido ejemplifica cómo las tecnologías avanzadas pueden amplificar la peligrosidad de las amenazas cibernéticas, gracias a su capacidad para evadir la detección y persistir en entornos seguros.

Por otro lado, este proyecto se destaca por abordar una brecha en la investigación: la falta de trabajos que examinen el robo de criptomonedas desde una perspectiva ofensiva. Para lograr esto, se implementarán dos mecanismos de robo de criptomonedas en la botnet. El primer ataque implica la exfiltración de las carteras wallet.dat. El segundo mecanismo se basa en el criptoclipping, una técnica mediante la cual el atacante monitorea el portapapeles del usuario y, al detectar la introducción de una dirección de cartera criptográfica, la sustituye por la del atacante.

Herramienta para el Futuro: Este proyecto no solo tiene un propósito educativo, sino que también pretende convertirse en una herramienta valiosa para futuros trabajos de investigación. La botnet desarrollada puede ser mejorada y utilizada para simular ataques y pruebas de penetración, lo que contribuirá al desarrollo de soluciones más efectivas para la detección y desmantelamiento de botnets en el futuro.

4.2. Elección del Lenguaje de Programación

Se han considerado C++, Golang y Python como lenguajes de programación, puesto que son de los mas usados para el desarrollo de malware y tienen las siguientes características que los hacen interesantes para este propósito:

Lenguaje	Ventajas	Desventajas
C++	• Compilado, difícil interpretar que hace el código • Fácil de camuflar, muchos ejecutables de sistema de Windows están en C++ • Muchas librerías públicas de malware disponibles • Velocidad de ejecución al ser compilado • Tamaño reducido del ejecutable • Facilidad de uso de la API de Windows •	• Mayor tiempo de desarrollo
Golang	• Programación más ágil • Compilado, difícil interpretar que hace el código • Velocidad de ejecución al ser compilado • Algunas librerías públicas de malware disponibles • Funcionalidad maliciosa se camufla bien entre todos los imports estáticos	• Tamaño de ejecutable grande al importar las librerías estáticamente • Un ejecutable muy grande puede ser sospecho • No es común un ejecutable de sistema de Windows en Go
Python	• Programación más ágil • Algunas librerías públicas de malware disponibles	• Más lento al ser interpretado • Fácil saber que hace el código al ser interpretado • No es común ejecutables de sistema de Windows en Python

En este contexto, la elección de C++ es la más acertada, ya que es un lenguaje de programación versátil y es el más cómodo para interaccionar con funciones de la API de Windows y punteros. Por otro lado, el hecho de que sea compilado es un factor importante a la hora de dificultar el análisis de malware.

4.3. Elección librería P2P

Para la realización del proyecto, se consideraron varias tecnologías con bibliotecas públicas en C++:

Libp2p: Una biblioteca de redes P2P con primitivas de pubsub para una eficiente difusión de mensajes.

Ricochet: Una herramienta P2P para mensajería instantánea anónima que utiliza difusión epidémica para la distribución de mensajes.

GNUnet: Un framework para construir redes P2P privadas y anónimas, que incluye funciones de broadcast nativas.

Tox: Una plataforma P2P para mensajería instantánea, voz y video que permite la difusión de mensajes a través de grupos públicos abiertos.

Si bien todas estas tecnologías tienen las funcionalidades necesarias para la realización del proyecto, se decidió descartar libp2p y GNUnet debido a la necesidad de implementar completamente la red P2P desde cero, lo cual sería poco realista en el plazo establecido para el proyecto, ya que requeriría la implementación de mecanismos de redundancia y enrutamiento, entre otros.

Ricochet, si bien ofrece una API de programación sencilla para el envío y recepción de mensajes entre nodos y gestiona su propio protocolo, carece de la capacidad de chats grupales, lo que lo convierte en una opción menos adecuada para el propósito del proyecto, que implica el envío de mensajes de difusión. Implementar mensajes de difusión en Ricochet requeriría una red superpuesta adicional y, por lo tanto, trabajo adicional.

Finalmente, se optó por Tox, que utiliza la biblioteca c-toxcore. Tox proporciona mensajería grupal y ofrece numerosas ventajas en comparación con otras tecnologías. A continuación, se detallan los beneficios y limitaciones de Tox:

Beneficios de usar Tox[19]

Tox se distingue por ofrecer un cifrado extremo a extremo , garantizando que los mensajes solo sean legibles para su destinatario.

Además, Tox incorpora un sistema de enrutamiento denominado "onion routing", que permite transmitir un mensaje desde un nodo A a un nodo B a través de tres nodos intermedios de Tox. Este proceso se lleva a cabo de manera cifrada entre los nodos, lo que impide que B conozca la identidad de A.

Por otro lado, tox está preparado para ser usado junto a Tor, VPNs y otros sistemas de anonimato

Por último, tox esta diseádo diversas amenazas, entre las que se incluyen ataques MitM, ataques Sybil, ataques DDoS, fingerprinting de tráfico, suplantación de identidad, ataques de repetición, envenenamiento y envío de SPAM.

Beneficios de usar grupos tox para el C&C[19]

Los grupos de Tox se organizan en una estructura de anillo, donde cada nodo mantiene vínculos con dos nodos adyacentes a la derecha y otros dos a la izquierda. Este enfoque brinda redundancia y, al mismo tiempo, resguarda la identidad de cada nodo, ya que solo los nodos vecinos directos poseen información sobre las direcciones IP de otros miembros del grupo.

Un aspecto destacado radica en que el algoritmo interno de los grupos Tox se encarga de eficazmente distribuir los mensajes a todos los nodos, incluso a aquellos que estuvieran desconectados previamente.

La resiliencia de los grupos Tox también se manifiesta en su capacidad para resistir ataques de denegación de servicio, ya que la eliminación de un grupo solo se produce si todos los miembros salen del mismo. En caso de fallos de nodos, el algoritmo de Tox se activa para redistribuir la red de forma automática.

Además, Tox ofrece opciones de seguridad adicionales al permitir que los grupos se protejan mediante contraseñas, lo que limita el acceso únicamente a aquellos con la clave adecuada.

Tox también brinda herramientas para gestionar la integridad de los grupos, incluyendo mecanismos para ignorar pares no deseados, además de la capacidad de establecer roles y aplicar funciones de moderación.

Limitaciones de Tox[19]

En un contexto de grupos Tox, es relevante mencionar que el enfoque de red de vecinos en forma de anillo puede presentar ciertos desafíos. Si un nodo experimenta problemas al retransmitir un mensaje, como dificultades de conexión, la lógica interna de Tox interviene para redirigir el tráfico y resolver posibles obstáculos. Sin embargo, es importante reconocer que esta optimización puede resultar en una leve variación en el tiempo necesario para la entrega de mensajes. En este sentido, es fundamental destacar que este ligero retraso no representa un inconveniente crítico en situaciones donde la comunicación en tiempo real no es una prioridad.

Es importante tener en cuenta que aunque Tox proporciona un nivel considerable de anonimato, no puede garantizarse por completo, ya que las conexiones entre los nodos se establecen directamente mediante direcciones IP y puertos. Además, la red podría estar sujeta a monitoreo o ataques potenciales. Por lo tanto, es recomendable que el botmaster use sistemas de anonimato tales como VPNs o proxies.

4.4. Decisiones de diseño uso Tox

En el diseño propuesto, se ha optado por implementar un grupo público en Tox, ya que la creación de grupos privados requiere invitaciones específicas asociadas a nodos particulares, lo cual requeriría de hacer un sistema centralizado de invitaciones y podría convertirse en un punto de falla. Es importante destacar que el funcionamiento de los grupos públicos en Tox se basa en el uso de anuncios DHT. Cuando un fundador crea un grupo público, ciertos nodos en la red Tox almacenan información esencial para permitir que otros nodos se unan al grupo, lo que incluye la dirección IP del fundador. A medida que más nodos

se unen al grupo, el nodo que mantiene estos anuncios DHT adquiere conocimiento de las nuevas direcciones IP.[19]

Si bien es poco probable que un usuario externo pueda interceptar o extraer información de manera sencilla del nodo que contiene estos anuncios DHT, se recomienda que el botmaster utilice herramientas de anonimato, como proxies, VPN o Tor, al crear el grupo Tox. Esto agregará una capa adicional de seguridad y privacidad al proceso.

El diseño propuesto implica que tanto el botmaster como los bots aprovecharán el grupo de Tox para enviar mensajes de difusión (broadcast) y así mantener una comunicación efectiva. En este esquema:

Cuando el botmaster envíe un comando, este se cifrará utilizando AES y se transmitirá al grupo.

- Si un bot desea responder a un comando, cifrará su respuesta utilizando la clave pública RSA del botmaster y la compartirá en el grupo. Esto garantiza que solo el botmaster pueda descifrar y comprender el contenido del mensaje.
- En el caso de que un bot desee enviar datos robados de una cartera criptográfica, también cifrará esta información con la clave pública del botmaster antes de transmitirla en el grupo.

4.5. Mecanismo robo criptocartera

Como se ha comentado anteriormente, la botnet esta orientada al robo de criptomonedas. Para llevar a cabo su cometido se usarán dos técnicas:

Robo de criptocartera

El malware escanea la computadora de la víctima en busca de archivos de carteras criptográfica (wallet.dat) de los gestores de carteras criptográficas instalados en el equipo. Para llevarlo a cabo realiza una búsqueda recursiva de los ficheros del equipo.

Clipboard Hijacking

El malware puede supervisar el portapapeles de la víctima y, cuando detecta una dirección de criptomoneda copiada, la reemplaza por la dirección del atacante. De esta manera, la víctima enviará el dinero al cibercriminal pensando que es el receptor legítimo.

Para llevar a cabo el ataque se hará un monitoreo de la función GetClipboardData, esperando que el usuario copie una dirección de criptocartera. En ese punto, se reemplazara por la dirección del atacante usando SetClipboardData.[20]

4.6. Ocultación DLL maliciosa

La lógica maliciosa de la botnet no se ubicará en el programa principal, sino que se ocultará cifrada dentro del ejecutable, con el objetivo de dificultar el análisis de malware y evitar la detección por parte de los antivirus, a través de firmas o mediante análisis de las funciones importadas, entre otras.

Existen miles de formas de ocultar la DLL maliciosa. A continuación se muestran las diferentes opciones que se han barajado:

- Como recurso legítimo
- Al final del código
- Al final de una sección
- Descargándola de internet al momento de ejecutar el programa

Se ha optado por esconderla como recurso cifrado. Aunque es una estrategia muy común, en la práctica sigue siendo efectiva.

Para llevar a cabo la carga de esta DLL en memoria se realizará una implementación manual de LoadLibrary, que no notifique al sistema operativo de la carga (reflective DLL Loading). De este modo la DLL no se verá en la lista de módulos cargados por el proceso, en caso de que se realice análisis de malware.

4.7. Persistencia

Para que el programa malicioso siga funcionando una vez el usuario reinicie o apague el equipo, es necesario dotar al bot de un mecanismo de persistencia en el sistema. Se han barajado las siguientes opciones:

Clave Run del Registro: La clave *"Run"* en el Registro de Windows contiene programas que se ejecutan automáticamente al iniciar sesión. A

Carpeta de Inicio (Startup Folder): La carpeta de inicio es un directorio especial en el sistema operativo donde los programas y archivos pueden colocarse para ejecutarse automáticamente al iniciar sesión.

Tarea programada: Las tareas programadas son procesos automatizados que se ejecutan en un sistema operativo en momentos específicos o a intervalos regulares. Estas tareas pueden ser configuradas para ejecutarse con diferentes niveles de privilegios y hacerlas pasar desapercibidas para el usuario.

Se ha optado por el uso de una Tarea Programada debido a su simplicidad, eficacia y capacidad para operar de manera discreta. Esta técnica es flexible, ya que se puede configurar para ser ejecutada con diversos niveles de usuario asignarle diversas condiciones de ejecución.

4.8. Tarea programada

La tarea programada tendrá como acción “Start a program Program”. El programa a ejecutar será el ejecutable de la botnet.
El usuario que la ejecutará será el usuario con sesión iniciada y correrá con los permisos de éste. El trigger que ejecutará la tarea programada será el inicio de sesión.

4.9. Configuración

Es necesario almacenar datos tales como la configuración que se usará para comunicarse con la red P2P (id de grupo y contraseña de grupo), la clave simétrica de cifrado y la clave pública de cifrado del botmaster, entre otros.

Se han barajado las siguientes ubicaciones para guardar la configuración:

- En un fichero almacenado en el sistema
- En el registro
- Como recurso en la DLL del bot
- Al final del ejecutable del bot

Se ha elegido la primera opción, ya que es una opción fácil de implementar y es la forma común de almacenar configuraciones. La opción del registro también es una buena candidata, pero se ha optado por la primera ya que la botnet no hará uso del registro para nada más. De este modo se evita dejar pistas de las que pueda tirar un analista de malware. Por otro lado se han descartado las dos últimas opciones, ya que modificar el ejecutable del bot hace un ruido innecesario y puede poner el foco del antivirus sobre éste.

El fichero de configuración se guardará cifrado, con el objetivo de que solo el ejecutable de la botnet pueda ver su contenido

4.10. Algoritmos de cifrado

La botnet usará dos algoritmos de cifrado:

Cifrado Asimétrico RSA: Cuando un peer roba una criptocartera, encripta los datos utilizando la clave privada del botmaster y luego envía un mensaje en modo de difusión (broadcast) a todos los peers. Este enfoque asegura que solo el botmaster pueda acceder al contenido de la criptocartera robada, ya que solo él posee la clave privada correspondiente.

Cifrado Simétrico AES: Cuando el botmaster desea enviar un comando a los bots, primero cifra el comando utilizando una clave simétrica que se estableció previamente en el ejecutable del malware. Luego, envía el comando cifrado a todos los peers a través de un mensaje de difusión (broadcast). Dado que los peers tienen acceso a esta clave simétrica compartida, son capaces de descifrar el comando y llevar a cabo la acción requerida.

Del mismo modo, los recursos encriptados (Ejecutable bot, DLL maliciosa y configuración) se cifrarán con AES.

4.11. Instalador

Para instalar el malware en el sistema objetivo, se requiere un instalador que facilite el proceso mientras dificulta el análisis del malware. En este contexto, se optará por desarrollar un ejecutable en lugar de un script ps1 u otros formatos, ya que estos últimos pueden ser fácilmente analizados al estar en texto plano.

El instalador contendrá el ejecutable de la botnet y la configuración almacenados como recursos cifrado, siguiendo un proceso idéntico al utilizado para el propio ejecutable de la botnet, que contiene la lógica maliciosa oculta en un recurso cifrado.

El instalador volcará el fichero de configuración en la ruta de disco especificada. A continuación desencriptará la DLL de la botnet y la volcará también en disco. Por último, creará la tarea programada correspondiente para ejecutar la botnet en el sistema objetivo y la ejecutará.

Es importante mencionar que la ejecución de la tarea programada se omitirá si es detectada por el antivirus durante las pruebas. Dicha detección podría ocurrir debido al comportamiento sospechoso de ejecutar la tarea programada desde el ejecutable.

4.12. Consola de C&C

La Consola de Comando y Control (C&C) es una parte central de la botnet que permite al operador o botmaster controlar y administrar los bots conectados. Proporciona una interfaz de línea de comandos a través de la cual se pueden emitir comandos a los bots. A continuación, se describen las funcionalidades y comandos que tendrá disponibles:

- Obtener bots online: permite listar el identificador de los bots conectados.
- Comando obtener información del sistema: Este comando permite obtener detalles de la configuración de sistema de los bots, tales como la dirección IP, el sistema operativo, la RAM instalada y el procesador, entre otros.
- Actualizar configuración de los bots: actualiza la configuración de los bots
- Consultar carteras de criptomonedas robadas: Este comando permite al operador buscar y consultar carteras de criptomonedas robadas

4.13. Niveles de privilegio

Para la elección del nivel de privilegio de la botnet se ha debatido entre los dos siguientes:

Ring 0 (Kernel)

Control Total: Ring 0 brinda acceso completo al hardware y al núcleo del sistema, otorgando un mayor nivel de control.

Mayor Riesgo: El control elevado implica la posibilidad de daños graves al sistema y lo convierte en una opción de alto riesgo.

Instalación Más Compleja: Requiere permisos excepcionales para su instalación, lo que dificulta su implementación.

Ring 3 (Usuario)

Restricciones de Seguridad: Con restricciones de seguridad, las amenazas en Ring 3 tienen menos capacidad para dañar el sistema.

Menor Riesgo: Esto reduce significativamente los riesgos asociados y hace que sea una

opción menos peligrosa.

Instalación Sencilla: La facilidad de instalación en Ring 3 es un punto positivo, lo que lo convierte en una elección preferida.

Ring 3 es la elección para llevar a cabo lel desarrollo de la botnet. Este nivel de privilegio es suficiente para llevar a cabo las funciones de la botnet. Es más efectivo para demostrar lo fácil que puede ser instalar una botnet en un sistema, en comparación con Ring 0, que sería considerablemente más complicado. Por otro lado, la implementación será más sencilla.

4.14. Ejecutables

Generador: El generador es una herramienta esencial que se encarga de crear el instalador del malware. Su función principal consiste en cifrar y agregar como recursos la DLL del bot y la configuración necesaria. El generador prepara el terreno para la posterior infección de las máquinas víctimas, garantizando que el malware esté listo para ser distribuido y ejecutado en los sistemas comprometidos.

Instalador: El instalador es la pieza de software que se encarga de llevar a cabo la instalación del bot en el ordenador de la víctima. Una vez que el generador ha preparado el malware, el instalador se asegura de infiltrar el bot en el sistema de la víctima sin levantar sospechas. Su función es esencial para que el malware sea efectivamente ejecutado en la máquina objetivo.

Programa Bot Zombie: El programa bot zombie es el malware propiamente dicho que se instala en el ordenador de la víctima. Este componente es la parte activa de la botnet y se encarga de comunicarse con el botmaster, recibir y ejecutar comandos, y enviar las carteras criptográficas robadas. Su operación se realiza en segundo plano, oculto al usuario, y se asegura de llevar a cabo las acciones maliciosas programadas.

Consola Botmaster: La consola del botmaster es una interfaz que permite al usuario interactuar con los bots. Su función principal es enviar comandos a los bots zombie y recibir las carteras criptográficas robadas por parte de los programas bot zombie. Actúa como el centro de mando y control de la botnet, proporcionando al botmaster la capacidad de gestionar y supervisar las operaciones de los bots.

4.15. Enfoque de implementación

- La configuración seguirá un patrón singleton o similar, para asegurar que solo exista una instancia.
- Las clases de la aplicación usarán un patrón de diseño RAII, para garantizar que se liberen los recursos correctamente en el destructor
- Se aplicaran los principios SOLID, tales como el principio de responsabilidad única y el principio de inversión de dependencia.
- Se asegurará el manejo correcto de los errores, poniendo comprobaciones de error en todas las funciones que lo requieran,
- Se añadirán comentarios para la legibilidad del código
- Siempre que sea posible se usarán clases de la std moderna (vector, thread, unique_ptr…)

4.16. Requisitos de la aplicación

A continuación, se muestran los requisitos funcionales y no funcionales de la aplicación. Estos están separados por categoría y estan priorizados por orden de importancia:

Requisitos funcionales prioritarios:

- Comunicación P2P cifrada entre nodos
- Consola comunicación botmaster con nodos
- Sustracción de carteras de criptomonedas (wallet.dat)
- Mecanismo para reemplazar dirección bitcoin copiada al portapapeles

Requisitos funcionales secundarios:

- Comando obtención de información detallada del sistema (CPU, RAM, IPs, etc.) de los equipos infectados mediante comando
- Almacenamiento de los datos extraídos en el nodo botmaster (en local)
- Persistencia de los nodos tras reinicios
- Comando actualización configuración en los bots

Requisitos no funcionales prioritarios

- Rendimiento, compatibilidad y escalabilidad
    - La red P2P debe poder operar sin servidores centralizados.
    - La consola de C&C debe ser capaz de controlar todos los nodos de forma simultánea
    - La botnet tendrá como mínimo compatibilidad con sistemas operativos Windows

11 de 64 bits.

- La botnet debe poder escalar como mínimo a 1000 nodos.

- Seguridad
  - Comandos P2P cifrados con AES-128
  - Datos exfiltrados deben ser encriptados con RSA de 2048 bits
  - Se debe proteger la identidad del botmaster
- Disponibilidad
  - Debe asegurarse que el sistema no caiga si se eliminan nodos de la red
  - Detección de nodos caídos
  - El sistema debe reubicar la red en función de los nodos caídos
  - Los bots deben poder reconectarse automáticamente a la red si se pierde la conexión.
- Legalidad
  - El malware no buscará expandirse a otros equipos.
  - Las pruebas se realizarán en entornos controlados.

Requisitos no funcionales secundarios

- Rendimiento, compatibilidad y escalabilidad
  - El tiempo de respuesta para comandos remotos debe ser inferior a 5 minutos
  - El código deberá seguir buenas prácticas y patrones de diseño que faciliten su mantenimiento.
- Seguridad
  - Se limita que información conocen unos nodos sobre otros y sobre la estructura actual de la red (solo los nodos vecinos que tienen que establecer conexiones directas conocen la IP).
  - Se evitará que usuarios ajenos puedan acceder al grupo de comunicación del C&C

4.17. Pseudocódigo componentes

```
# installer_main - Instalador
Función installer_main():
    ejecutable_bot = LeerRecursoCifrado("ejecutable_bot.exe")
    desencriptar(ejecutable_bot, clave_AES)
    GuardarArchivo(ejecutable_bot, ruta_especificada)
    configuracion = LeerRecursoCifrado("configuracion.txt")
    desencriptar(configuracion, clave_AES)
    GuardarArchivo(configuracion, ruta_especificada)

# bot_main - Bot
Función bot_main():
    dll_maliciosa = LeerRecursoCifrado("malware.dll")
    desencriptar(dll_maliciosa, clave_AES)
    CargarDLLenMemoria(dll_maliciosa)
    InicializarFuncionalidadMaliciosa()

# backdoor_maindll - DLL Maliciosa
Función backdoor_maindll():
    loadConf()
    InicializarClaseCommunicationTox()
    InicializarClaseReemplazoPortapapeles()
    InicializarClaseBusquedaCriptocarteras()
    InicializarThreadConsultaComandos()

Función ThreadConsultaComandos():
    Mensajes = LeerMensajesGrupoTox()
    Para Cada Mensaje en Mensajes:
        Comando = DesencriptarMensaje(Mensaje, clave_AES)
        EjecutarComando(Comando)
        Si HayTCarterasCapturadas:
            CifrarYEnviarDatos(DatosRobados, clave_RSA_publica_botmaster)
            Dormir(10000)

Función CifrarYEnviarDatos(datos, clave_RSA_botmaster):
    Mensaje = CrearMensajeCifrado(datos, clave_RSA_publica_botmaster)
    EnviarMensaje(Mensaje)

Función ejecutarComando(comando):
    Si comando == "actualizar conf":
        NuevaConfiguracion = LeerNuevaConfiguracionDesdeDisco()
        UpdateConf(NuevaConfiguracion)
    Sino Si comando == "getSystemInfo":
        RespuestaSystemInfo = ObtenerSystemInfo()
        MensajeRespuesta = CrearStructRespuesta(RespuestaSystemInfo)
        MensajeCifrado = CifrarMensajeConAES(MensajeRespuesta, clave_AES)
        EnviarMensajeGrupoTox(MensajeCifrado)

Función loadConf():
    configuracion = LeerConfiguracionDesdeDisco("configuracion.ini")
    Desencriptar(configuracion, clave_AES)
    CargarConfiguracionEnMemoria(configuracion)

Función updateConf(nuevaConfiguracion):
    GrabarNuevaConfiguracionEnMemoria(nuevaConfiguracion)
    Encriptar(nuevaConfiguracion, clave_AES)
    GuardarConfiguracionEnDisco(nuevaConfiguracion)

Clase ReemplazoPortapapeles:
    ObtenerDatosPortapapeles
    Si (datos == expresionRegularBitcoin):
        ReemplazarDirección()
    Dormir(500);

Clase ClaseBusquedaCriptocarteras:
    BusquedaRecursivaEquipo(ruta_ inicial, extension)
    Si(carterasRobadas):
        GuardarCarteraRobadaEnMemoria();

Clase CommunicatiónTox():
    inicializarCallbacks()
    ConectarGrupoTox()
    CrearThreadLecturaEnvioMensajes()

# generator_main - Generador
Función generator_main(argumentos):
    configuracion = GenerarConfiguracionDesdeArgumentos(argumentos)
    Cifrar(configuracion, clave_AES)
```

```
    AlmacenarConfiguracionComoRecurso(configuracion, "configuracion.txt")

    dll_maliciosa = LeerArchivo("malware.dll")
    Cifrar(dll_maliciosa, clave_AES)
    AlmacenarDLLComoRecurso(dll_maliciosa, "malware.dll")

# console_main - Interfaz de Usuario
Función console_main():
    Mientras Verdadero:
        MostrarListaComandos()
        Comando = LeerComandoUsuario()

        Si Comando == "consultar  bots online":
            BotsOnline = ObtenerBotsOnline()
            MostrarMensaje("Número de bots online: " + NumeroBotsOnline)

        Si Comando == "actualizar conf":
            RutaNuevaConf = LeerRutaNuevaConfiguracionDesdeUsuario()
            NuevaConfiguracion = LeerNuevaConfiguracionDesdeArchivo(RutaNuevaConf)
            StructConfCifrada = CrearStructConComandoYCifrar(NuevaConfiguracion, "actualizar conf", clave_AES)
            EnviarMensajeGrupoTox(StructConfCifrada)
            MostrarMensaje("Configuración actualizada exitosamente.")

        Si Comando == "getSystemInfo":
            StructComandoCifrado = CifrarStructComando("getSystemInfo", clave_AES)
            EnviarMensajeGrupoTox(StructComandoCifrado)
            MensajesRecibidos = LeerMensajesGrupoTox()

            Para Cada MensajeCifrado en MensajesRecibidos:
                MensajeDesencriptado = DesencriptarMensajeConRSA(MensajeCifrado, clave_RSA_botmaster)
                MostrarMensaje("Mensaje recibido de bot: " + MensajeDesencriptado)

        Si Comando == "leer_carteras_robadas":
            MensajesCifrados = LeerMensajesGrupoTox()

            CarterasRobadas = []
            Para Cada MensajeCifrado en MensajesCifrados:
                MensajeDesencriptado = DesencriptarMensajeConRSA(MensajeCifrado, clave_RSA_botmaster)
                CarteraRobada = ExtraerCarteraDesdeMensaje(MensajeDesencriptado)
                CarterasRobadas.Agregar(CarteraRobada)

            CarterasAnteriores = LeerCarterasDesdeArchivo()
            CarterasRobadas += CarterasAnteriores
            MostrarCarterasRobadas(CarterasRobadas)
        Dormir(10000)
```

5. Fase de implementación

5.1. Flujo de trabajo en la implementación

La implementación del proyecto se llevó a cabo siguiendo un proceso iterativo basado en metodologías ágiles. Se dividió en diversas fases que abarcaron desde la planificación inicial hasta la entrega final del producto. El flujo de trabajo se estructuró en las siguientes etapas principales:

Desarrollo de Prototipos

Se procedió a la creación de prototipos y versiones preliminares del producto para validar el diseño propuesto. Se implementaron funcionalidades básicas para verificar la viabilidad técnica del enfoque elegido y se realizaron pruebas de concepto.

Iteraciones y Desarrollo Incremental

El desarrollo se realizó de manera incremental y en iteraciones cortas. Se priorizaron las características fundamentales del producto y se implementaron en ciclos de desarrollo que abarcaban aproximadamente dos semanas cada uno. En cada iteración se llevaban a cabo reuniones de revisión para evaluar el progreso y realizar ajustes según el feedback obtenido.

Integración y Pruebas Continuas

A medida que se completaban las funcionalidades, se procedía a su integración en el sistema general. Se realizaron pruebas continuas para asegurar el correcto funcionamiento de las nuevas implementaciones y su compatibilidad con las partes ya existentes del producto.

Optimización y Refactorización

Se dedicó una fase específica para la optimización del código y la refactorización de componentes. Se identificaron áreas de mejora en el rendimiento y la eficiencia del sistema, implementando cambios para garantizar un funcionamiento óptimo.

Entrega y Documentación

Finalmente, se preparó la entrega del producto, incluyendo la documentación correspondiente, tanto a nivel de usuario como de desarrollo. Se generaron manuales, guías de usuario y se documentaron los procesos técnicos clave para facilitar el mantenimiento y la comprensión del sistema.

5.2. Componentes del proyecto

BotmasterConsole

Permite la interacción con los bots a través de comandos específicos. Su implementación se basa en un bucle interactivo que espera y procesa comandos introducidos por el usuario, ejecutando acciones correspondientes sobre los bots y el sistema.

Funcionalidades Principales

- Listar Bots: Muestra la lista de bots online.
- Obtener Carteras Robadas: Recupera y almacena localmente carteras robadas.
- Obtener Información del Sistema: Solicita y visualiza información del sistema de un bot específico.
- Configurar Parámetros: Facilita el cambio de configuraciones del sistema.

El código de BotmasterConsole se encarga de gestionar la interacción con los bots y la configuración, utilizando clases como Configuration, CryptRSA, ToxManager, entre otras, para realizar estas acciones.

Generator

Se encarga de generar y preparar recursos cruciales para la instalación y configuración de los bots en máquinas objetivo. Está diseñado para ejecutarse en un entorno de línea de comandos y requiere tres parámetros: commanderToxId, groupPassword, y btc_replace_addr.

Proceso de Generación:

1. Generación de Claves de Cifrado
    - Genera una clave AES aleatoria y un par de claves RSA para el sistema.
    - Guarda la clave privada RSA en un archivo local.
2. Configuración del Sistema
    - Configura los datos generados, incluyendo las claves, contraseñas y direcciones requeridas.
    - Cifra y guarda la configuración para el botmaster.
3. Cifrado de Recursos
    - Cifra y prepara recursos críticos como archivos DLL y ejecutables para su distribución.
    - Los recursos cifrados se incorporan al instalador final installer.exe para su implementación en máquinas objetivo.

Este componente utiliza las clases Configuration, CryptAES, CryptRSA, y otras, para realizar el cifrado, generación y preparación de recursos necesarios para la instalación de los bots.

Installer

Es responsable de la configuración e implementación de los recursos necesarios en las máquinas objetivo para el funcionamiento de los bots. Este programa, ejecutado al inicio, se encarga de realizar las siguientes tareas:

1. **Creación de Directorios:** Verifica y crea directorios esenciales, como el directorio de AppData, si no existen previamente.
2. **Extracción y Configuración:** Extrae la configuración cifrada necesaria para la operación de los bots y la almacena en la ruta de AppData.
3. **Desencriptado de Recursos:** Recupera recursos encriptados, como ejecutables cifrados, y procede a su desencriptado para su uso en la instalación.
4. **Almacenamiento y Programación:** Guarda el recurso desencriptado en la ruta de AppData y programa una tarea programada utilizando la función createScheduleTask.

DllCore

El componente DllCore constituye el núcleo principal del bot, encargado de ejecutar funciones esenciales para su operatividad en las máquinas objetivo. Sus funciones principales son:

1. I**nicialización de Configuración:** Carga la configuración esencial del bot desde un archivo en la ruta de AppData y la inicializa para el funcionamiento del bot.
2. **Comunicación P2P:** Inicia el sistema de comunicación punto a punto (P2P) utilizando ToxManager, basado en los datos de configuración proporcionados.
3. **Manejo de Mensajes:** Inicia un hilo de ejecución (messageHandlingThread) para manejar mensajes entrantes.
4. **Funcionalidades Específicas:** Utiliza la herramienta ClipboardMonitor para supervisar y realizar operaciones específicas con el portapapeles.
5. **Búsqueda de Criptocarteras:** Realiza búsquedas en el equipo de posibles criptocarteras utilizando FileSearcher.

Este componente se integra con diferentes clases como Variables, Utilities, FileSearcher, y otras utilidades de manipulación de archivos y comunicación para llevar a cabo sus funciones fundamentales.

Pe_Wrapper

Se trata de un ejecutable que encapsula el núcleo del bot (DllCore), permitiendo que este pase desapercibido en análisis de malware u otros procesos de inspección. Sus principales funcionalidades son:

1. **Encapsulación de DllCore:** Este ejecutable encapsula y ejecuta el núcleo del bot (DllCore) después de desencriptarlo y cargarlo en memoria.
2. **Desencriptación y Carga:** Lee el recurso protegido que contiene la información cifrada de DllCore y procede a desencriptarlo utilizando las claves incrustadas

(embeddedAESKey). Luego, carga esta información desencriptada en la memoria.

3. **Ejecución de Funciones del Bot:** Una vez que se ha cargado el núcleo del bot en la memoria, se ejecutan las funciones y tareas correspondientes a través de la librería MemoryModule.

Este componente se enfoca en eludir técnicas de detección comunes, encapsulando el código esencial del bot y permitiendo su ejecución de manera más discreta y menos perceptible en análisis de seguridad.

ToxCore

El proyecto ToxCore se ha desarrollado a partir del código base de la librería Tox, con el objetivo de construir una versión independiente que pueda ser compilada como una librería estática (*.lib). Esta librería estática se integra como un componente esencial en el proyecto principal, eliminando la necesidad de depender de DLLs externas de Tox durante la ejecución.

Objetivos del Proyecto Independiente:

- **Librería Estática Autocontenida:** El propósito principal es generar una librería estática de Tox que pueda ser utilizada en el proyecto principal sin requerir DLLs externas durante la ejecución.
- **Facilidad de Depuración y Debugging:** Permite un análisis más detallado y un proceso de debugging más sencillo al desvincularse de las dependencias externas, facilitando la identificación y corrección de errores específicos.
- **Resolución de Problemas y Mejoras:** Se arreglaron errores funcionales identificados previamente en ToxCore, y se adaptó la librería a las necesidades particulares del proyecto.
- **Independencia de Dependencias Externas:** Elimina la necesidad de las DLLs externas de Tox, garantizando una integración completa y autónoma de la funcionalidad de ToxCore en el proyecto principal.

CommonLib

Es una biblioteca diseñada para centralizar y proporcionar un conjunto de funcionalidades comunes utilizadas en varios proyectos. Esta biblioteca encapsula un conjunto de clases y funciones que abarcan áreas de múltiples proyectos, facilitando la reutilización y el mantenimiento de funcionalidades clave.

Componentes y Funcionalidades:

- **Configuration (Configuración):** Gestiona la configuración de los proyectos, facilitando su carga y modificación.
- **CryptRSA y CryptAES (Cifrado RSA y AES):** Ofrece métodos y herramientas para el cifrado y descifrado de información utilizando los algoritmos RSA y AES, respectivamente.
- **Message (Mensaje):** Administra la estructura y el procesamiento de mensajes utilizados en la comunicación entre distintos componentes del proyecto.

- **Resource (Recurso):** Proporciona funciones para la manipulación de recursos, como la lectura y escritura de archivos embebidos.
- **ToxManager (Gestor de Tox):** Facilita la interacción con la biblioteca Tox, ofreciendo métodos para la gestión de la comunicación P2P.
- **Utilities (Utilidades):** Contiene una serie de funciones de utilidad genéricas que se utilizan en varios contextos del proyecto.
- **Variables (Variables):** Agrupa y gestiona variables globales o constantes fundamentales para los diferentes aspectos del proyecto.

5.3. Decisiones de la implementación

Durante el desarrollo, se tomaron decisiones para asegurar la robustez, seguridad y funcionalidad del proyecto. Algunas de las decisiones clave incluyen:

Validación y Comprobación de Errores: Se implementó un sistema de funciones de inicialización para verificar errores, permitiendo un control exhaustivo sobre el flujo de ejecución y posibles fallos en las operaciones. Se optó por devolver códigos de error DWORD y gestionar las excepciones con precisión utilizando mensajes a través de std::cerr para una mejor trazabilidad en caso de fallo.

Manejo de Recursos: Se aplicó el patrón RAII en las clases para garantizar la liberación ordenada y segura de recursos. Esto asegura una gestión adecuada de memoria y recursos del sistema.

Sincronización y Seguridad: Para asegurar operaciones seguras de lectura y escritura, se empleó el uso de std::mutex para sincronizar hilos, evitando posibles problemas de concurrencia.

Estructuración de Datos y Comunicación: Se implementó una estructuración adecuada de los datos utilizando message.h, lo que permitió dividir grandes conjuntos de información en chunks para su envío y recepción. Se utilizó un enfoque de envoltorio digital para los comandos entre los bots y el botmaster, cifrando la clave AES con la clave pública RSA asociada al botmaster.

Cifrado y Camuflaje: La configuración del bot y la DLL maliciosa fueron cifradas y almacenadas como recursos en el instalador. La DLL se camufló dentro de Pe_Wrapper como recurso cifrado para dificultar su identificación en análisis de malware.

Abstracción de Funcionalidades: Se creó la clase ToxManager para facilitar el uso de los grupos de Tox en la implementación. Esto permitió un manejo simplificado del sistema de comunicación, abstrayendo detalles técnicos y optimizando la interacción con los grupos publicos de Tox.

Uso de la std moderna: Se emplearon las clases modernas de la librería estándar de C++, como std::vector, std::map, y std::thread, para mejorar la claridad del código.

5.4. Integración de tecnologías

Toxcore: Proporciona la comunicación P2P

Librería Bcrypt: Utilizado para la encriptación AES y MD5.

Librería Wincrypt: Ofrece la generación de llaves y la encriptación RSA-

Librería taskschd: Necesario para la creación y gestión de tareas programadas en el sistema operativo.

Librería comsupp: Utilizada para trabajar con el Component Object Model (COM). Usado en la creación de la tarea programada

MemoryModule de Joachim Bauch[21]: Necesaria para cargar la DLL maliciosa en memoria, permitiendo su ejecución sin ser almacenada en disco, lo que contribuye al camuflaje y la persistencia del bot.

5.5. Gestión de datos

La gestión de datos se lleva a cabo de manera cifrada y estructurada para garantizar la seguridad y confidencialidad en todas las etapas del sistema. La configuración se almacena con cifrado AES en la máquina objetivo (bots), específicamente en *%APPDATA%/microsoft/windows/winupdate/winupdate.log.*

Los bots y el botmaster manejan la configuración de manera distinta: en los bots, se almacena sin la clave privada RSA, mientras que en el botmaster se conserva la configuración completa, incluyendo la clave privada RSA. Ambos se almacenan en la misma ruta que la consola de comandos para una gestión eficiente.

Además, las carteras robadas se almacenan progresivamente en el directorio wallet del botmaster a medida que se reciben de los bots, facilitando su gestión y análisis.

Los mensajes hacia el botmaster se envían de manera encriptada con envoltorio digital, mientras que los comandos hacia los bots se cifran utilizando AES. Estos mensajes se encapsulan en las estructuras de datos definidas en message.h, que incluyen campos esenciales para su procesamiento, como números de secuencia, información del receptor y emisor.

El PE_wrapper se guarda en el disco del ordenador objetivo (bot) en %APPDATA%/microsoft/windows/winupdate/winupdate.exe, mientras que la DLL maliciosa se almacena como recurso cifrado dentro de PE_Wrapper.exe, utilizando también AES para su protección.

Tanto PE_Wrapper.exe como la configuración se guardan como recursos cifrados con AES en el instalador. La clave AES para descifrar la configuración y los recursos está incrustada en el código dentro de variables.h en embeddedAESKey, asegurando un acceso seguro y controlado a los datos.

5.6. Aspectos de seguridad

Cifrado y Protección de Comunicaciones: El uso de AES para cifrar los comandos hacia los bots, asegurando que, incluso si un tercero accede al grupo, el contenido de los mensajes permanezca inaccesible.

La protección de los grupos Tox mediante contraseñas, impidiendo el acceso de usuarios externos no autorizados.

Confidencialidad de Mensajes Críticos: La implementación de envolturas digitales en los mensajes hacia el botmaster, garantizando que solo el destinatario pueda acceder a información sensible como carteras robadas o respuestas a comandos fundamentales.

Cifrado de Recursos y Configuración: La aplicación de AES para cifrar tanto la configuración como los recursos del ejecutable, asegurando que el contenido permanezca inaccesible para personas no autorizadas.

Niveles de Seguridad Adicionales Provistos por Tox: Tox proporciona mecanismos de cifrado de extremo a extremo y brinda un grado de anonimato, añadiendo capas de seguridad a las comunicaciones.

Consideraciones sobre Seguridad No Abordadas: Los comandos, al estar protegidos solo por AES, podrían ser susceptibles a la manipulación por parte de un bot malicioso. Esta vulnerabilidad permite la potencial inserción de comandos falsos o maliciosos dentro de la red. Este enfoque se ha mantenido deliberadamente, para demostrar cómo una botnet podría ser atacada por las autoridades.

Seguridad y Memoria: El uso de las clases modernas de la librería estándar de C++ reduce significativamente las posibilidades de fugas de memoria y mejora la seguridad del sistema, ya que su gestión de memoria es más segura y predecible.

Gestión de Recursos Críticos: La implementación emplea el patrón RAII en las clases para asegurar la liberación adecuada de recursos cruciales. Esto asegura que los recursos se liberen cuando ya no son necesarios, evitando posibles fugas de memoria.

Persistencia de los Grupos Tox: Los grupos Tox no pueden ser eliminados a menos que se den de baja todos los usuarios del grupo. Esto garantiza la persistencia del grupo y la información en él, asegurando la continuidad de la red incluso en circunstancias adversas.

Gestión de Errores: Todas las funciones han sido diseñadas para comprobar sus valores de retorno y gestionar los errores de manera adecuada, asegurando así una respuesta resiliente ante situaciones imprevistas.

5.7. Rendimiento y escalabilidad

Escalabilidad del Bootstrapping: El Botmaster actúa como nodo bootstrap para emitir invitaciones al grupo. Ante un aumento en la red, la capacidad de cómputo del Botmaster puede resultar crítica.

Tiempo de Respuesta y Limitaciones de Escalabilidad: La velocidad de respuesta está intrínsecamente ligada a la velocidad de la red Tox. Si bien proporciona tiempos de respuesta aceptables, el envío de mensajes como mensajes grupales puede limitar la escalabilidad. Tox administra el envío de mensajes mediante una estructura de anillo entre los nodos, lo que puede presentar limitaciones en términos de escalabilidad bajo ciertas condiciones de carga.

5.8. Desafíos y soluciones

Grupos Tox Públicos: Los grupos Tox públicos generaron problemas de estabilidad y conexión inconsistente. Se migró a grupos por invitación y se implementó un sistema de invitaciones a través del Botmaster para garantizar una mayor estabilidad en la conexión y funcionamiento de los grupos.

Métodos de Tox: Identificación de errores en funciones específicas de la biblioteca Tox. Se realizó la corrección de estos errores mediante modificaciones directas en el código de la biblioteca Tox para asegurar su correcto funcionamiento.

Identificación de Usuarios en Grupos Tox: Problemas con la identificación precisa de usuarios dentro de los grupos Tox, resultando en identificaciones erróneas. Se implementaron estructuras de mensajes más detalladas y metadatos específicos (message.h) para mejorar la precisión y confiabilidad en la identificación de los usuarios.

Compilación de ToxCore: Dificultades en la compilación de ToxCore desde el repositorio oficial. Se creó un proyecto personalizado para la compilación local de ToxCore, asegurando una correcta integración y funcionalidad en el proyecto principal.

Tamaño de Mensajes: Problemas al manejar mensajes de gran tamaño que excedían las limitaciones de ToxCore. Se desarrolló un sistema para fragmentar y enviar mensajes por partes, permitiendo el envío adecuado de mensajes grandes dentro de los límites establecidos.

Conexión de Bots: Bots que presentaban dificultades ocasionales para conectarse al grupo. Se implementó un mecanismo que monitorea y restablece automáticamente la conexión de los bots al grupo en caso de una desconexión inconsistente.

Acceso a Carteras Criptográficas: Limitaciones al intentar leer el contenido de las criptocarteras cuando el gestor de wallets estaba abierto. Se logró acceder a las criptocarteras realizando copias de seguridad de los archivos y accediendo a los datos desde dichas copias para evitar bloqueos o problemas de acceso.

Lectura de Configuración: Dificultades al leer configuraciones con elementos de tamaño

variable. Se implementaron funciones específicas de serialización y deserialización para manejar adecuadamente la lectura de configuraciones con elementos variables, asegurando una lectura correcta y coherente.

5.9. Clases y componentes específicos

botmasterCommands: Contiene funciones esenciales que definen las interacciones del botmaster con los bots en la red de la botnet, permitiendo operaciones como listar bots, cambiar configuraciones, obtener carteras robadas y obtener información del sistema de los bots.

configuration: Gestiona la configuración global de la botnet, incluyendo datos críticos como IDs de Tox, contraseñas de grupo y direcciones Bitcoin para reemplazo. Contiene funciones para cargar, actualizar y guardar la configuración, además de proveer métodos para encriptar y desencriptar los datos de configuración.

cryptAES y cryptRSA: Proporcionan funciones criptográficas para cifrar y descifrar mensajes entre el botmaster y los bots, garantizando la confidencialidad y la integridad de la comunicación.

toxmanager: Administra las comunicaciones y operaciones en la red de la botnet a través de la plataforma Tox. Provee métodos para enviar y recibir mensajes, inicializar tanto el rol de comandante como de pares (peers), actualizar la contraseña del grupo y configurar el estado del grupo. También gestiona la persistencia de datos de Tox para el commander, así como las interacciones con otros nodos en la red.

message: Define estructuras y funciones para la creación, cifrado y descifrado de mensajes dentro de la red de la botnet. Ofrece métodos para la encapsulación y manipulación de mensajes, incluyendo la estructura de los mensajes cifrados y sus componentes, así como la lectura y escritura de mensajes entre el botmaster y los bots en la red.

variables y utilities: Contienen funciones y variables auxiliares para la gestión de datos y operaciones esenciales en la botnet, proporcionando recursos útiles y compartidos a través del sistema.

resource: Contiene funciones para manipular recursos dentro de un ejecutable, permitiendo la extracción, lectura y adición de datos en forma de recursos. Facilita la gestión de recursos asociados al ejecutable, como la extracción de datos específicos, la lectura de recursos y la adición de nuevos datos a los recursos existentes.

consoleMain: Es el main del botmaster. Este archivo permite ejecutar comandos desde la consola, como listar bots, obtener carteras robadas, obtener información del sistema de los bots y modificar la configuración del programa.

botCommands: Este archivo gestiona la recepción y el procesamiento dcomandos para los bots en la red.

clipboardMonitor: Clase encargada de supervisar el portapapeles, con el objetivo de identificar cuando se copia una dirección de bitcoin y reemplazarla por la dirección que le

especifiquemos.

dllCoreMain: es el controlador principal de la DLL, manejando la configuración desde archivos, inicializando comunicaciones P2P, administrando hilos para mensajes, buscando criptocarteras en el equipo y monitorizando el portapapeles esperando que copie direcciones bitcoin.

FileSearcher: Es una clase que busca y captura archivos de extensiones específicas en rutas designadas, en nuestro caso archivos de criptocarteras. Explora directorios definidos, identifica archivos con extensiones definidas, y almacena los archivos encontrados.

GetSystemInfo: obtiene información del sistema utilizando el comando de windows systeminfo. Abre un pipe para ejecutar el comando, lee la salida línea por línea y convierte cada línea en bytes para almacenarla en un vector, representando así la información del sistema.

GenetatorMain: Se encarga de generar las claves de cifrado, los archivos de configuración y de preparar el instalador que se usará en la máquina objetivo.

TaskScheduler: interactúa con el servicio de Programador de tareas de Windows para crear tareas programadas con los parámetros que le especifiquemos.

InstallerMain: incluye la lógica para instalar el malware en la máquina objetivo

PE_Wrapper: programa aparentemente legítimo que carga la dllCoreMain en memoria

MemoryModule: contiene la lógica para cargar la dll maliciosa en memoria.

5.10. Documentación adicional

En la carpeta “documentation” incluida en la entrega, puede consultar la documentación adicional, que contiene toda la documentación de las funciones y clases de la aplicación, así como diagramas de éstas.

6. Pruebas y evaluación

6.1. Pruebas realizadas

En el proceso de desarrollo de la aplicación, se llevaron a cabo diversas pruebas para garantizar la funcionalidad, seguridad y rendimiento del sistema. Estas pruebas se diseñaron para evaluar cada componente de manera individual y también para verificar su funcionamiento conjunto. A continuación, se detallan las pruebas realizadas:

Pruebas de Funcionalidad:

- Evaluación del proceso de generación e instalación del malware
- Verificación del reemplazo exitoso de direcciones Bitcoin copiadas en el portapapeles.
- Confirmación de la extracción y envió correctos de carteras criptográficas extraídas de los bots
- Evaluación de la capacidad del botmaster para enviar comandos y recibir respuestas de los bots.
- Verificación del correcto funcionamiento de los comandos getsystemInfo, cambio de configuración y listar bots.

Pruebas de Integración:

- Evaluación de la coherencia entre los componentes de los distintos elementos de la botnet

Pruebas de Seguridad:

- Resistencia a la detección por parte de herramientas antivirus y sistemas de seguridad.
- Capacidad de comunicarse con el Firewall activo
- Verificación de que no sea posible unirse al grupo sin la contraseña correcta y una invitación del botmaster.
- Realización de pruebas de penetración para evaluar la resistencia del sistema contra intentos de ataque externo, incluyendo:
    - Denegación de servicio a un bot específico mediante el envío masivo de mensajes "getsysteminfo" (suplantando al botmaster).
    - Denegación de servicio mediante el comando de cambio de configuración, suplantando al botmaster (cambio de contraseña).

Pruebas de Estrés y Escalabilidad:

- Sometimiento del sistema a envío masivo de mensajes para evaluar su rendimiento bajo condiciones de uso intensivo.

Pruebas de Recuperación:

- Introducción de interrupciones en la comunicación botmaster-bot para evaluar la capacidad del sistema para recuperarse y reanudar operaciones normales.

Pruebas de Rendimiento:

- Verificación de la recepción de mensajes dentro del intervalo de tiempo especificado.
- Comprobación de que el sistema no se ve afectado al monitorizar o reemplazar el portapapeles.
- Evaluación del comportamiento del sistema al buscar y extraer el contenido de las carteras criptográficas.
- Confirmación de que el sistema no experimenta alteraciones al ejecutar la consola del botmaster o el ejecutable del bot.
- Verificación de que el rendimiento del sistema no se ve afectado al ejecutar comandos.
- Comprobación de que la red Tox funciona sin afectar el rendimiento general del sistema.
- Toma de métricas de rendimiento del bot y el botmaster (consumo de CPU y RAM)

Pruebas de persistencia:

- Verificación continua de la capacidad del malware para mantener una comunicación persistente y estable con el botmaster a lo largo del tiempo.
- Comprobación de que todas las funciones continúan funcionando correctamente después de reiniciar la máquina.

Pruebas componentes individuales:

Consola Botmaster:

- Comprobación de la capacidad de la consola para ejecutar comandos seleccionados por el usuario con los parámetros correctos.
- Verificación de la recepción correcta de respuestas a los comandos ejecutados.

Generador:

- Prueba de recepción correcta de argumentos.
- Comprobación de la generación exitosa del instalador.
- Verificación, mediante Resource Hacker, del almacenamiento adecuado de los recursos cifrados en el instalador.

PE_Wrapper:

- Verificación de la capacidad del programa para obtener la DLL maliciosa del recurso cifrado y descifrarla correctamente.
- Comprobación de la carga exitosa del malware en memoria y su ejecución mediante

Memorymodule.

Instalador:

- Comprobación de la instalación correcta del sistema al ejecutar el instalador en la máquina virtual.
- Verificación de la creación exitosa de la tarea programada.
- Confirmación de que los archivos de configuración y del malware se almacenan en la ruta especificada.
- Comprobación de la ejecución inmediata del malware sin necesidad de reiniciar la máquina.

DLL Core (DLL Maliciosa):

- Verificación de la capacidad de recibir y procesar comandos correctamente.
- Comprobación de la ejecución exitosa de acciones asociadas a los comandos, como la actualización de la configuración y la obtención de información del sistema
- Confirmación de que, al obtener carteras criptográficas, estas se envían correctamente a través de la red Tox.

Clase ToxManager:

- Comprobación de la creación del grupo Tox si no está creado.
- Verificación de la correcta configuración de los parámetros del grupo, como la tipología y la contraseña de acceso.
- Confirmación del correcto funcionamiento del sistema de invitación al grupo
- Comprobación del envío y recepción exitosos de mensajes entre los usuarios del grupo.
- Verificación de que el botmaster conserva las credenciales de Tox, incluido el Tox ID y el grupo creado, incluso después de reinicios.
- Configuración:
- Comprobación de la capacidad de la clase para guardar la configuración correctamente, incluida la serialización y encriptado de la misma.
- Verificación de que los datos se almacenan correctamente en la estructura de datos.

EncryptAES:

- Confirmación de que el texto de prueba se cifra y descifra correctamente.

CryptRSA:

- Comprobación de la generación y exportación correctas del par de claves.
- Verificación de que el texto de prueba se cifra y descifra correctamente.

Clase Message:

- Comprobación de la correcta serialización de los mensajes, incluida la introducción de metadatos en la estructura y el cifrado del mensaje.
- Verificación de la partición y reconstrucción adecuadas de mensajes grandes.

Clase ClipboardMonitor:

- Comprobación de la monitorización adecuada del portapapeles.
- Verificación de la correcta sustitución de direcciones de Bitcoin en el portapapeles cuando se detecta una copia.

Filesearcher:

- Comprobación de la capacidad de la clase para encontrar todas las carteras a partir de la ruta especificada.
- Verificación de la obtención de información sobre las criptocarteras incluso si están abiertas por otro proceso.
- Confirmación de que no se vuelve a capturar una cartera que ya ha sido capturada.

Comando getsysteminfo:

- Verificación de que el comando es recibido y procesado por el bot indicado como parámetro.
- Comprobación de la ejecución exitosa del comando systeminfo de Windows
- Confirmación de que el botmaster recibe correctamente la información del sistema.

Comando de cambio de configuración:

- Comprobación de que todos los bots reciben la configuración correctamente.
- Verificación de que todos los bots actualizan la configuración correctamente.
- Confirmación de que el botmaster actualiza la configuración correctamente.
- Comprobación de que, si hay un cambio de contraseña de grupo Tox, este se actualiza correctamente, incluido el cambio en la red Tox con "tox_group_founder_set_password".
- Verificación de que, si se cambia el Tox ID del botmaster, los usuarios se comunican con el nuevo ID.
- Confirmación de que, si se cambia la dirección de reemplazo de Bitcoin, esta se actualiza correctamente.

Comando de listar bots:

- Confirmación del envio del mensaje y recepción de todos los Ids por parte del botmaster

6.2. Resultados y análisis

Comportamiento de la Botnet:

Todas las funcionalidades del botmaster y del malware en las máquinas virtuales operan correctamente, cumpliendo con los requisitos de la fase de diseño. Se han realizado con éxito todas las pruebas establecidas.

Detección y Evasión:

Las pruebas se llevaron a cabo con Windows Defender activo en las máquinas virtuales y en el sistema operativo anfitrión. En ningún momento la botnet fue detectada, ya sea en su funcionamiento habitual o durante la ejecución de comandos. La tarea programada puede ejecutarse en el instalador sin reiniciar el equipo sin ser detectada por el software de seguridad. La comunicación se realiza de manera exitosa y no es bloqueada por el Firewall de Windows.

Impacto en el Sistema:

No se observaron variaciones de rendimiento en las máquinas infectadas con la botnet corriendo sin ejecutar comandos. Tampoco hubo cambios en el rendimiento al ejecutar comandos, capturar criptocarteras, o monitorear o reemplazar el portapapeles.

Eficiencia y Rendimiento:

El proceso de la botnet en la máquina virtual muestra un aumento del % de procesador y un consumo de memoria de X MB. La consola del botmaster no afecta el rendimiento del sistema. Se estableció un tiempo de espera de 15 segundos para el envío y recepción de comandos, cumpliendo con los requisitos de la fase de diseño.

Validación de Objetivos:

Se lograron todos los hitos previstos en la metodología del proyecto, demostrando con éxito la viabilidad de la creación de una botnet P2P basada en el robo de criptomonedas, incluyendo ataques de clipboard hijacking y robos de criptocarteras (wallet.dat).

Limitaciones del Estudio:

La versión actual de la botnet no puede capturar archivos wallet.dat protegidos con contraseña. Además, utiliza grupos públicos de Tox con invitación, lo que, aunque no afecta significativamente la funcionalidad, podría ser modificado para mejorar la escalabilidad del sistema.

Actualmente, la botnet utiliza grupos públicos de Tox con invitación, lo que implica que el botmaster cumple el papel de nodo bootstrap para invitar al bot al grupo. Aunque esta configuración no representa una limitación crítica y se ha diseñado considerando la seguridad del protocolo Tox ante posibles ataques de denegación de servicio, se reconoce que, en términos de escalabilidad, podría resultar más eficiente modificar el C&C para permitir que varios nodos de la red asuman la función de nodo bootstrap.

Por otro lado, es posible realizar ataques de denegación de servicio, mediante el envío de comandos que suplanten al botmaster. Esto es posible ya que el botmaster cifra los

comandos usando la clave AES que también comparten los bots. Una posible solución consiste en que el botmaster firme el mensaje con su clave RSA.

7. Conclusiones y Trabajo Futuro

7.1. Conclusiones

La facilidad con la que se pueden perpetrar robos de criptomonedas es asombrosa, desde técnicas simples como el Clipboard Hijacking hasta sistemas de robo de archivos wallet.dat, accesibles incluso para quienes no son expertos en la materia. Cabe destacar que el programa es capaz de evitar la detección de Windows defender a pesar de no usar técnicas de evasión de malware sofisticadas.

La creación de botnets P2P es viable con herramientas comunes, aunque garantizar el anonimato en Tox requiere medidas adicionales, como el uso de VPNs o Tor. Destaca la robustez de los grupos, siendo necesario eliminar por completo las credenciales para eliminar un grupo, incluyendo las del botmaster. La resistencia a la denegación de servicio en la comunicación botmaster-grupo es notable debido a que no es posible eliminar al botmaster del grupo y que este sigue pudiéndose conectar a la red Tox desde cualquier equipo mientras conserve sus archivos de credenciales.

La funcionalidad de grupos públicos DHT (sin invitación) en Tox es una adición relativamente reciente. Aunque inicialmente se consideró su uso, la implementación final se inclinó hacia los grupos públicos con invitación debido a que el funcionamiento de la primera era inconsistente

Por último, mencionar que solo es posible capturar criptocarteras que no estén encriptadas con contraseña. Esto se debe a que aún no se ha implementado un sistema para capturar las teclas introducidas por el usuario.

7.2. Trabajo futuro

Multiples nodos Bootstrap: Considerar ceder la responsabilidad de gestionar las invitaciones al grupo entre múltiples bots de la red para mejorar la escalabilidad y distribuir la carga de manera equitativa.

Almacenamiento de Mensajes y Carteras Robadas: Explorar la posibilidad de almacenar mensajes y carteras robadas en los bots cuando el botmaster está desconectado, incluso mediante un sistema de almacenamiento P2P.

Persistencia de Datos en Bots: Implementar un sistema para almacenar los hashes de carteras robadas en el disco de los bots, de manera que cuando estos reinicien no vuelvan a enviar las criptocarteras que ya fueron enviadas anteriormente.

Captura de Contraseñas mediante Keylogging: Incorporar la funcionalidad de capturar las teclas pulsadas por la víctima, de tal forma que la botnet sea capaz de robar carteras protegidas con clave. Para llevar a cabo esta funcionalidad es necesario realizar una inyección de proceso al gestor de criptocarteras y posteriormente hacer un hookeo a las funciones asociadas a la captura de teclas, usualmente PeekMessage / GetMessage.

Firma digital de los comandos por parte del botmaster: Como se ha comentado, firmar los comandos con la clave RSA del botmaster impediría la posibilidad de que los bots envíen mensajes suplantando al botmaster.

8. Anexos

8.1. Anexo A: Glosario

Malware: Software dañino diseñado con propósitos maliciosos, como robar datos o dañar sistemas.

Botnet: Red de dispositivos infectados con malware que son controlados de forma remota por un atacante para ejecutar acciones maliciosas de forma coordinada.

C&C (command and control): El sistema utilizado por el botmaster para comunicarse con los bots e impartir órdenes.

P2P (Peer-to-Peer): Un modelo de red donde cada dispositivo o nodo tiene la capacidad tanto de actuar como cliente como servidor, permitiendo compartir recursossin depender de un servidor centralizado.

Keylogging: Es la acción de registrar o monitorear las pulsaciones de teclas en un dispositivo.

DLL (Dynamic Link Library): Librería de enlace dinámico que contiene código ejecutable.

Hooking: Técnica para interceptar llamadas a funciones del sistema e insertar código personalizado.

DdoS (Distributed Denial of Service): Ataque distribuido de denegación de servicio.

SPAM: Mensajes electrónicos no solicitados enviados en masa. Un uso común de las botnets es el envío de SPAM.

Phishing: Técnica para engañar a los usuarios haciéndoles creer que un sitio fraudulento es legítimo para robar información personal.

Fluxing: Técnica utilizada en botnets para variar continuamente la infraestructura de Comando y Control, dificultando su detección y eliminación.

Mixers: Servicios para ocultar el origen de fondos en criptomonedas al mezclar transacciones de múltiples usuarios.

Tor: Red que permite comunicaciones anónimas ocultando la localización e identidad de los usuarios.

DLT (Distributed Ledger Technology): Sistema descentralizado de registro y almacenamiento de datos que se distribuye en múltiples nodos.

IoT (Internet of Things): Dispositivos interconectados con acceso a internet como electrodomésticos, wearables, sensores, etc.

LN (Lightning Network): Capa de pago construida sobre blockchain para transacciones rápidas y económicas.

DHT (Distributed Hash Table): Tabla de hash distribuida para almacenar datos en una red P2P de manera descentralizada.

Broadcast: Difusión de un mensaje a todos los nodos de una red.

Kernel: Núcleo del sistema operativo que gestiona el hardware y los recursos. Se corresponde con ring 0

Ring 3: Nivel de ejecución de aplicaciones de usuario en un sistema operativo.

IDS (Intrusion Detection System): Un sistema que detecta actividades no autorizadas o maliciosas en una red o sistema informático.

IPS (Intrusion Prevention System): Similar al IDS, pero el IPS no solo detecta amenazas, sino que también toma medidas activas para bloquear o prevenir posibles intrusiones.

8.2. Anexo B: Guía de usuario

Ubicación de los Entregables: Todos los archivos necesarios se encuentran en la carpeta 'bin'. Los entregables incluyen Generator.exe, DllCore.dll, BotmasterConsole.exe, Installer.exe y Pe_Wrapper.exe.

Pasos a seguir:

1. Ejecutar BotmasterConsole: En su primera ejecución, se conectará a la red tox y mostrará el identificador toxId generado, que será el identificador del botmaster en adelante.
2. Ejecutar el Generador: Se debe ejecutar Generator.exe con los siguientes parámetros:

   Generator.exe <commanderToxId> <groupPassword> <btc_replace_addr>

3. Esto almacenará los datos necesarios para la instalación en installer.exe, junto con la configuración del botmaster, guardada en el mismo directorio de ejecución de Generator.exe.
4. Instalación del Malware: Para instalar el malware en la máquina objetivo, simplemente ejecuta Installer.exe. Este activará la tarea programada en el lugar, poniendo el malware en funcionamiento de inmediato, sin necesidad de reiniciar la máquina.

Interacción con el Malware

BotMasterConsole: Al ejecutarlo desde la línea de comandos, mostrará un menú con las siguientes opciones:

'listar': Obtiene los identificadores de los bots conectados al grupo.

'carteras': Permite obtener las carteras robadas hasta el momento.

'getsysteminfo': Recibe el identificador de un bot obtenido previamente con 'listar' y devuelve el resultado del comando systeminfo ejecutado en la máquina del bot.

'configurar': Permite actualizar la configuración. Requiere los nuevos parámetros de configuración.

'salir': Cierra el programa.

8.3. Anexo C: Diagrama de Gantt

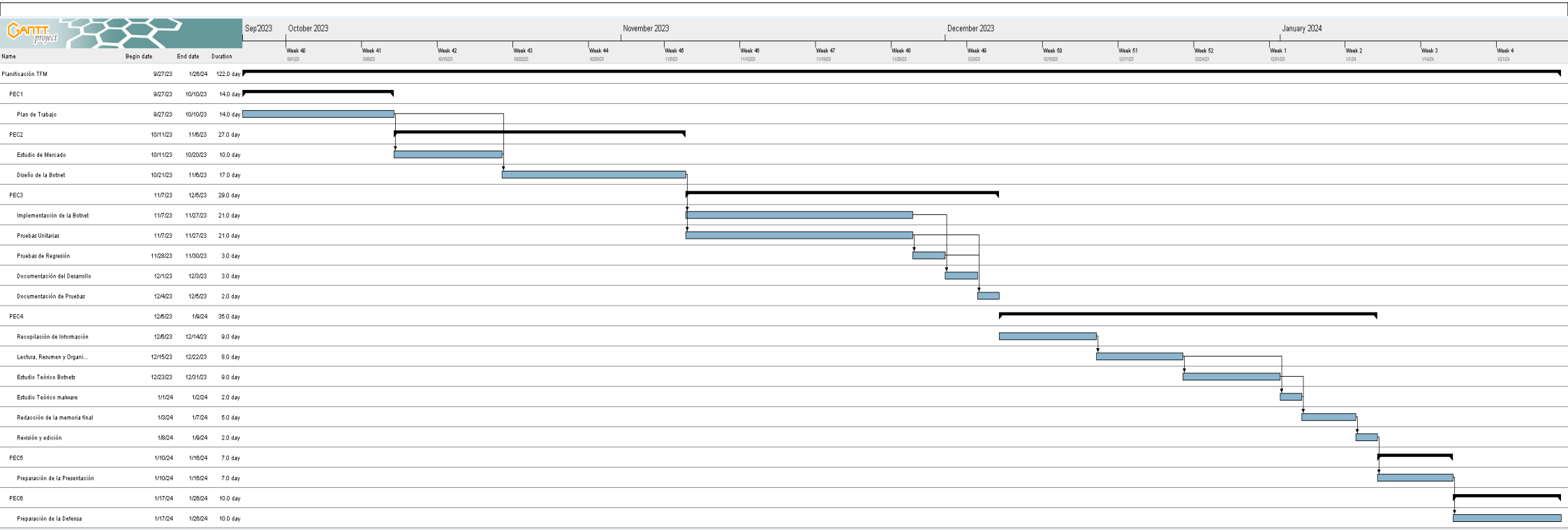

Figure 7: Diagrama de Gantt

9. Referencias

[1] Z. Wang *et al.*, «DeepC2: AI-Powered Covert Command and Control on OSNs», en *Information and Communications Security*, C. Alcaraz, L. Chen, S. Li, y P. Samarati, Eds., en Lecture Notes in Computer Science. Cham: Springer International Publishing, 2022, pp. 394-414. doi: 10.1007/978-3-031-15777-6_22.

[2] S. Yadav, «Political Propagation of Social Botnets: Policy Consequences». arXiv, 10 de mayo de 2022. doi: 10.48550/arXiv.2205.04830.

[3] S. Ikeda, «Nation-State DDoS Attacks May Be the “New Normal”; Leaked Documents Reveal Russia’s FSB Is Seeking to Build a Massive IoT Botnet», CPO Magazine. Accedido: 6 de noviembre de 2023. [En línea]. Disponible en: https://www.cpomagazine.com/cyber-security/nation-state-ddos-attacks-may-be-the-new-normal-leaked-documents-reveal-russias-fsb-is-seeking-to-build-a-massive-iot-botnet/

[4] J. Gao y M. Liu, «A Study on Social Network based P2P Botnet», vol. 2, n.º 3.

[5] J. Yin, X. Cui, y K. Li, «A Reputation-Based Resilient and Recoverable P2P Botnet», en *2017 IEEE Second International Conference on Data Science in Cyberspace (DSC)*, jun. 2017, pp. 275-282. doi: 10.1109/DSC.2017.20.

[6] H. Gao *et al.*, «ZombieCoin3.0: On the Looming of a Novel Botnet Fortified by Distributed Ledger Technology and Internet of Things», en *2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, dic. 2021, pp. 1625-1634. doi: 10.1109/HPCC-DSS-SmartCity-DependSys53884.2021.00240.

[7] O. Alibrahim y M. Malaika, «Botract: abusing smart contracts and blockchain for botnet command and control», *Int. J. Inf. Comput. Secur.*, vol. 17, n.º 1-2, pp. 147-163, ene. 2022, doi: 10.1504/ijics.2022.121295.

[8] A. Kurt, E. Erdin, K. Akkaya, S. Uluagac, y M. Cebe, «D-LNBot: A Scalable, Cost-Free and Covert Hybrid Botnet on Bitcoin’s Lightning Network», *IEEE Transactions on Dependable and Secure Computing*, pp. 1-18, 2023, doi: 10.1109/TDSC.2023.3300738.

[9] L. Böck, S. Karuppayah, M. Mühlhäuser, y E. Vasilomanolakis, «An Overview of the Botnet Simulation Framework», *The Journal on Cybercrime & Digital Investigations*, pp. 1-10 Pages, dic. 2020, doi: 10.18464/CYBIN.V6I1.25.

[10] A. Beauchaine, O. Collins, y M. Yun, «BotsideP2P: A Peer-to-Peer Botnet Testbed», en *2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, dic. 2021, pp. 0236-0242. doi: 10.1109/UEMCON53757.2021.9666641.

[11] alexeybu, «Phorpiex botnet is back with a new Twizt: Hijacking Hundreds of crypto transactions», Check Point Research. Accedido: 6 de noviembre de 2023. [En línea]. Disponible en: https://research.checkpoint.com/2021/phorpiex-botnet-is-back-with-a-new-twizt-hijacking-hundreds-of-crypto-transactions/

[12] «Mantis - the most powerful botnet to date», The Cloudflare Blog. Accedido: 5 de enero de 2024. [En línea]. Disponible en: https://blog.cloudflare.com/mantis-botnet

[13] C. Ortega, «Botnets: La amenaza fantasma».

[14] N. P. Selvaraj *et al.*, «Exposure of Botnets in Cloud Environment by Expending Trust Model with CANFES Classification Approach», *Electronics*, vol. 11, n.º 15, Art. n.º 15, ene. 2022, doi: 10.3390/electronics11152350.

[15] M. Feily, A. Shahrestani, y S. Ramadass, «A Survey of Botnet and Botnet Detection», en *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, Athens/Glyfada, Greece: IEEE, 2009, pp. 268-273. doi: 10.1109/SECURWARE.2009.48.

[16] A. Karim, R. Salleh, M. Shiraz, S. Shah, I. AWAN, y N. Anuar, «Botnet detection techniques: review, future trends and issues», vol. 15, nov. 2014, doi:

10.1631/jzus.C1300242.
[17] H. R. Zeidanloo, M. J. Z. Shooshtari, P. V. Amoli, M. Safari, y M. Zamani, «A taxonomy of Botnet detection techniques», en *2010 3rd International Conference on Computer Science and Information Technology*, jul. 2010, pp. 158-162. doi: 10.1109/ICCSIT.2010.5563555.
[18] B. Rahbarinia, R. Perdisci, M. Antonakakis, y D. Dagon, «SinkMiner: Mining Botnet Sinkholes for Fun and Profit».
[19] «The TokTok Project - Protocol». Accedido: 6 de noviembre de 2023. [En línea]. Disponible en: https://toktok.ltd/spec
[20] «Clipboard Hijacker Dropped By STOP Ransomware – SonicWall». Accedido: 6 de noviembre de 2023. [En línea]. Disponible en: https://securitynews.sonicwall.com/xmlpost/clipboard-hijacker-dropped-by-stop-ransomware/
[21] «Figure 1. The main component of the botnet», ResearchGate. Accedido: 7 de enero de 2024. [En línea]. Disponible en: https://www.researchgate.net/figure/The-main-component-of-the-botnet_fig1_339352098
[22] «ShellCon 2018: Hacking con Python». Accedido: 7 de enero de 2024. [En línea]. Disponible en: https://es.slideshare.net/jdaanial/shellcon-2018-hacking-con-python
[23] «What Is a Botnet?», What Is a Botnet? Accedido: 7 de enero de 2024. [En línea]. Disponible en: https://www.avast.com/c-botnet
[24] J. Hurst, «Basic Windows API Hooking», Geek Culture. Accedido: 7 de enero de 2024. [En línea]. Disponible en: https://medium.com/geekculture/basic-windows-api-hooking-acb8d275e9b8