Prototipo de sistema de detección temprana de software malintencionado por medio de comportamientos anómalos en Windows 10

Carlos Eduardo Guerrero Jaramillo

Universidad del Valle
Escuela de Ingeniería en Sistemas y Computación
Tuluá
2025

Prototipo de sistema de detección temprana de software malintencionado por medio de comportamientos anómalos en Windows 10

Carlos Eduardo Guerrero Jaramillo
Código: 202060216
carlos.eduardo.guerrero@correounivalle.edu.co

Director
José Luis Unás Gómez, Msc
jose.unas@correounivalle.edu.co

Universidad del Valle
Escuela de Ingeniería en Sistemas y Computación
Tuluá
2025

Trabajo de grado presentado por
Carlos Eduardo Guerrero Jaramillo
Como requisito para la obtención del título de Ingeniero de Sistemas

José Luis Unás Gómez
Director

Jurado

Jurado

Agradecer es honrar el camino recorrido; es mirar atrás y reconocer que cada paso estuvo sostenido por manos visibles e invisibles. Es comprender que, sin ese algo, sin ese alguien, esta meta no sería más que un sueño inacabado.

Hoy, al llegar a este momento tan esperado, levanto la voz del corazón para agradecer.

A mi madre, por sus sacrificios silenciosos y sus oraciones incansables. Por cuidar de mí como se cuida una llama en la tormenta, por enseñarme con paciencia y amor, y por llenar mi vida con la ternura de su fe inquebrantable.

A mi padre, por estar siempre presente, aun cuando el cansancio se le aferraba a los pies y no había bálsamo que calmara sus dolores. Por su lucha diaria, por su esfuerzo constante, y por ser ejemplo silencioso de fortaleza.

A mí mismo, por no rendirme. Por no tirar la toalla cuando el viento soplaba en contra, por sostener la mirada en el horizonte y perseguir con firmeza esta meta que hoy se hace real.

Y a mi compañera de vida, por habitar los días grises y convertirlos en luz. Por caminar a mi lado, por comprender los silencios, y por ser faro cuando la niebla era espesa.

Este trabajo no es solo un documento: es un testimonio de amor, de lucha, de presencia y de gratitud.

Tabla de Contenidos

Lista de Figuras

Lista de Tablas

Resumen

Este proyecto se realizó con el objetivo de desarrollar un prototipo de sistema para la detección temprana de comportamientos anómalos en el sistema operativo Windows 10, con el fin de prevenir posibles ataques de ransomware de cifrado y la pérdida de datos que estos conllevan. La justificación de este trabajo radica en la necesidad de herramientas que permitan identificar patrones de comportamiento sospechosos en sistemas informáticos, específicamente en Windows 10, lo que posibilitará recolectar la información necesaria para detectar cuándo se está ejecutando un ataque de ransomware y desplegar una respuesta efectiva en el sistema operativo, reduciendo significativamente el riesgo de pérdida de información. Finalmente, se consiguió desarrollar un prototipo de detección funcional con un alto nivel de asertividad en la detección de comportamientos anómalos por medio de la aplicación de puntaje por entropía, lo cuál cumple con los objetivos planteados.

Palabras clave: Ciberseguridad, Ciberataque, Integridad de datos, Seguridad Informática, Ransomware, Ransomware de cifrado, Windows 10.

Abstract

This project was carried out with the aim of developing a prototype system for the early detection of anomalous behavior in the Windows 10 operating system, in order to prevent possible encryption ransomware attacks and the loss of data that these entail. The rationale behind this work lies in the need for tools that can identify suspicious behavior patterns in computer systems, specifically in Windows 10, which will make it possible to collect the information necessary to detect when a ransomware attack is being executed and deploy an effective response in the operating system, significantly reducing the risk of information loss. Finally, a functional detection prototype was developed with a high level of assertiveness in detecting anomalous behavior through the application of entropy scoring, which meets the objectives set.

Keywords: Cybersecurity, Cyberattack, Data integrity, Computer security, Ransomware, Encryption ransomware, Windows 10.

Información sobre los capítulos

En la Tabla 1 se muestra una corta descripción de cada capítulo del documento.

Tabla 1: Estructura de capítulos. **Fuente: Elaboración propia.**

Capítulo	**Descripción**
Capítulo 1: Introducción	En este capítulo se introduce el contexto del proyecto, el problema a resolver, el objetivo general y los objetivos específicos del proyecto, así como el planteamiento del problema.
Capítulo 2: Marco referencial	En este capítulo se dan a conocer las teorías, los conceptos y los antecedentes para entender el contexto en el que se va a resolver el problema del proyecto.
Capítulo 3: Alcance	En este capítulo se define el alcance del proyecto.
Capítulo 4: Análisis de Fundamentos	En este capítulo se desarrolla el **objetivo específico 1**, el cuál se realiza el análisis del funcionamiento de un ransomware y el análisis de las técnicas y modelos de detección de ransomware.
Capítulo 5: Diseño del prototipo	En este capítulo se desarrolla el **objetivo específico 2**, el cual se realiza el diseño de la arquitectura del prototipo, los modelos de datos propuestos y el diseño de la interfaz de usuario.
Capítulo 6: Implementación del prototipo	En este capítulo se desarrolla el **objetivo específico 3**, el cual se realiza la implementación de la arquitectura del prototipo, los modelos de datos propuestos y la interfaz de usuario, con el propósito de obtener el prototipo de detección temprana.
Capítulo 7: Pruebas aplicadas al prototipo	En este capítulo se desarrolla el **objetivo específico 4**, el cual se realiza las pruebas de uso del prototipo.
Capítulo 8: Conclusiones y trabajos futuros	En este capítulo se presentan las conclusiones finales del proyecto y los trabajos futuros posibles para aplicar a este prototipo

Capítulo 1
Introducción

Introducción

La inseguridad informática se ha convertido en una preocupación fundamental, ya que afecta desde usuarios individuales hasta organizaciones a nivel mundial; las vulnerabilidades en los sistemas informáticos han generado un impacto significativo en la cantidad de ataques que se realizan anualmente, generando impactos económicos sin precedentes. Según el informe "Cost of a Data Breach 2025" de IBM, el costo promedio mundial de una violación de datos alcanzó los 4,4 millones de dólares en 2025 [13], cifra que refleja no únicamente pérdidas financieras directas, sino también daños reputacionales, interrupciones operativas prolongadas y costos de recuperación que pueden extenderse durante años. Este escenario ha posicionado a la ciberseguridad como una disciplina crítica, cuyo objetivo principal consiste en identificar anomalías en los sistemas informáticos para determinar su naturaleza potencialmente maliciosa. Una anomalía se define como cualquier comportamiento o patrón que se desvía significativamente de la actividad normal esperada, manifestándose a través de irregularidades a nivel de archivos, procesos, memoria o tráfico de red [12].

Entre las diversas amenazas que pueden comprometer los sistemas informáticos, el *ransomware* se destaca como una de las formas más devastadoras de ataques cibernéticos. Este es un tipo de *malware* que se aprovecha de ciertas vulnerabilidades de los sistemas operativos para llevar a cabo su propósito: impedir el acceso a datos o sistemas completos y exigir un rescate a cambio de restaurarlos [6]. A diferencia de otros tipos de malware, el ransomware una vez infecta un dispositivo utiliza métodos criptográficos para evitar que el usuario pueda recuperar sus archivos de manera convencional, haciendo que las víctimas deban pagar sumas de dinero (generalmente en criptomonedas) para recuperar sus archivos. Este panorama revela que los ciberatacantes obtienen beneficios económicos significativos de estos ataques, lucrándose a través de los pagos que exigen para liberar los sistemas infectados. A medida que el ransomware evoluciona, los autores de programas maliciosos están redoblando sus esfuerzos para frustrar la detección por parte del software antimalware [4].

Según la encuesta global "*The State of Ransomware 2024*" de Sophos, realizada entre enero y febrero de 2024 con la participación de 5 000 responsables de TI en 14 países, entre el 60 % y el 68 % de las organizaciones de 11 de 15 sectores reportaron incidentes de ransomware en el último año [33]. Estos resultados demuestran no sólo la frecuencia de los ataques, sino también el alto impacto económico y operativo que conllevan.

Windows 10 concentra hoy más del 50 % del mercado de sistemas de escritorio a nivel mundial [34], convirtiéndolo en el blanco preferido de los ciberdelincuentes. Su amplio uso a nivel mundial abren múltiples vulnerabilidades que pueden aprovechar para realizar un ataque de *ransomware*. Por ello, es necesario diseñar mecanismos de detección temprana basados en el análisis de comportamientos anómalos, capaces de identificar patrones de cifrado antes de que se complete el

ataque.

En este contexto, el presente trabajo propone la creación de un prototipo de sistema de detección temprana de comportamientos anómalos en el Sistema Operativo (SO) Windows 10, con el objetivo de minimizar el riesgo de ataques de ransomware y proteger la integridad de la información del usuario.

1.1. Descripción del problema

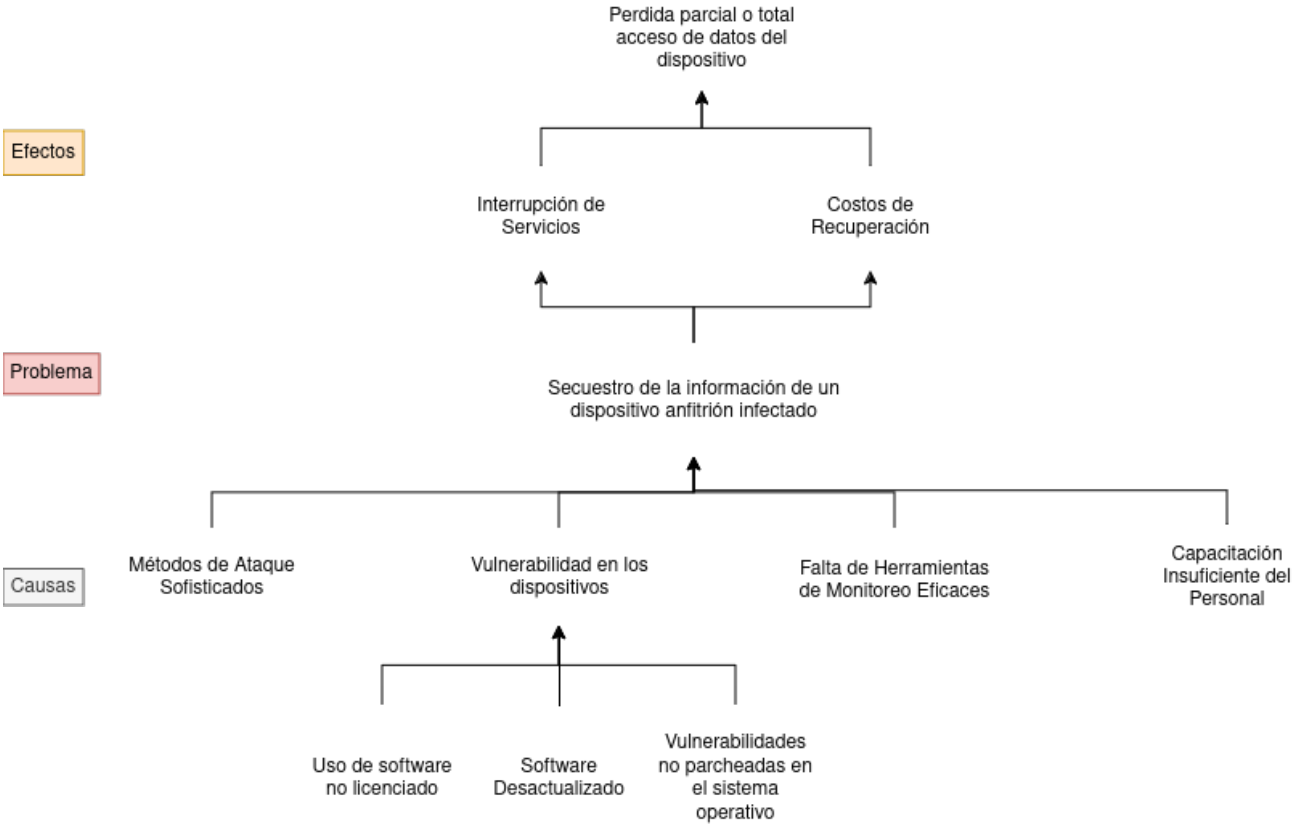
El crecimiento exponencial de los ataques de ransomware ha revelado gravemente la vulnerabilidad de los sistemas informáticos, con un impacto especialmente severo en los sistemas operativos basados en Windows. Este tipo de malware, diseñado para cifrar archivos críticos del usuario y exigir un rescate para su liberación, ha evolucionado para explotar diversas brechas de seguridad tanto en el sistema operativo como en el software relacionado.

La prevalencia de Windows como el sistema operativo más utilizado a nivel mundial lo convierte en un objetivo principal para los ciberdelincuentes, quienes desarrollan continuamente nuevas variantes de ransomware para evadir las defensas existentes. Un artículo del blog oficial de Kaspersky determinó, a través de un estudio basado en metadatos anónimos de Kaspersky Security Network, que casi una cuarta parte (22 %) de los usuarios de PCs a nivel global aún utiliza Windows 7, un sistema operativo que dejó de recibir asistencia técnica en enero de 2020 [7]. Esta situación complica aún más el panorama, ya que un sistema operativo sin soporte no recibe parches de seguridad, lo que expone al dispositivo que lo utiliza a un mayor riesgo de ciberataques de ransomware, como se ilustra en una de las causas de este problema en la Figura 1.1 (Software Desactualizado).

A esta problemática, se suma la falta de experiencia de los usuarios en la adopción de buenas prácticas de ciberseguridad, ya sea por ignorar las advertencias o por no contar con una capacitación adecuada. La ausencia de medidas preventivas a tiempo incrementa considerablemente la probabilidad de sufrir un ciberataque. Según un artículo de Forbes, se concluye que las empresas han sufrido al menos un incidente cibernético en los últimos dos años debido a la falta de personal calificado [18].

En el ámbito económico, tanto personas como empresas y organizaciones enfrentan pérdidas sustanciales no solo por el pago de rescates, sino también por los costos asociados a la interrupción de servicios, la recuperación de datos y la implementación de medidas de mitigación. Los ciberdelincuentes emplean métodos como las monedas virtuales para solicitar pagos a cambio de la liberación de archivos. La creciente adopción de estas monedas a nivel global ha despertado el interés de los atacantes, ya que su facilidad de uso y la dificultad para rastrearlas constituyen la base sobre la cual se diseñan muchos de estos malware [14].

Figura 1.1: Árbol del problema. **Fuente: Elaboración propia.**



1.2. Formulación del problema

El problema principal radica en la falta de herramientas efectivas para detectar y prevenir ataques de ransomware de manera correcta y/o inteligente en el Sistema Operativo (SO) Windows 10. La falta de capacidad de los sistemas operativos para identificar patrones de comportamiento anómalos en tiempo real deja a los usuarios vulnerables a los ataques de cifrado de datos.

Esto conduce al siguiente problema:
¿Cómo diseñar un prototipo de detección de software malintencionado por medio de comportamientos anómalos para Windows 10?

Con el problema establecido, se plantea la necesidad de desarrollar un prototipo de sistema de detección temprana de ransomware en Windows 10, que permita identificar y mitigar los riesgos asociados con este tipo de amenazas. Este prototipo buscará proporcionar una respuesta efectiva ante comportamientos sospechosos en el sistema operativo, reduciendo así los efectos del ataque de ransomware.

1.3. Justificación

La implementación de un prototipo de detección temprana de software malintencionado en Windows 10 se justifica por múltiples factores que convergen en el panorama actual de ciberseguridad, los cuales requieren de respuestas innovadoras y efectivas ante esta creciente amenaza.
Los datos estadísticos revelan la urgencia de desarrollar sistemas de detección proactivos. Según la encuesta global "The State of Ransomware 2024" de Sophos [33], entre el 60 % y 68 % de las organizaciones en 11 de 15 sectores industriales reportaron incidentes de ransomware durante el último año. Esta prevalencia no solo demuestra la frecuencia alarmante de los ataques, sino que también evidencia que las medidas de seguridad tradicionales resultan insuficientes para contener esta amenaza en constante evolución.
Por otra parte, la predominancia de Windows 10, que concentra más del 50 % del mercado de sistemas de escritorio a nivel mundial [34], convierte a este sistema operativo en el objetivo principal de los ciberdelincuentes. Esta amplia adopción, combinada con la diversidad de configuraciones y aplicaciones que ejecuta, genera múltiples vectores de ataque que los atacantes explotan sistemáticamente para desplegar ransomware.
El desarrollo de este prototipo contribuirá al avance del conocimiento en el campo de la ciberseguridad, especialmente en áreas como: implementación de mecanismos de respuesta que minimicen el daño potencial de los ataques de *ransomware* y el desarrollo de métodos de detección temprana con bajo impacto en el rendimiento del sistema. Los algoritmos y metodologías implementados en este prototipo pueden servir como base para futuras investigaciones y desarrollos, como se menciona a lo largo de este trabajo.

1.4. Objetivos

1.4.1. Objetivo general

Desarrollar un prototipo de sistema de detección temprana de ataques de ransomware para Windows 10, que permita reducir significativamente el riesgo de pérdida de datos y otros daños causados por este tipo de ataques.

1.4.2. Objetivos específicos

- Identificar las posibles entradas o vulnerabilidades que utiliza un ransomware para infectar un dispositivo.
- Permitir al usuario interpretar las posibles amenazas que se encuentran en el dispositivo de manera gráfica.
- Evaluar la eficacia del modelo de detección de comportamientos anómalos en la identificación de ataques de ransomware en diferentes escenarios.
- Garantizar que el prototipo sea compatible con el sistema operativo para un óptimo funcionamiento.

1.5. Resultados obtenidos

Para abordar la sección de los resultados obtenidos, se tienen en cuenta los objetivos específicos observados en la Tabla 1.1, ya que por medio del cumplimiento de estos últimos se llega a alcanzar el cumplimiento del objetivo general.

Tabla 1.1: Resultados obtenidos según los objetivos específicos. **Fuente: Elaboración propia**

Objetivo específico	**Resultado obtenido**
Identificar las posibles entradas o vulnerabilidades que utiliza un ransomware para infectar un dispositivo.	▪ Documentos con comportamientos conocidos del ransomware. ▪ Herramienta de simulación de ransomware.
Permitir al usuario interpretar las posibles amenazas que se encuentran en el dispositivo de manera gráfica.	▪ Mockups de la interfaz. ▪ Prototipo interactivo. ▪ Documentación del diseño.
Evaluar la eficacia del modelo de detección de comportamientos anómalos en la identificación de ataques de ransomware en diferentes escenarios.	▪ Documentación del prototipo. ▪ Código fuente de la implementación. ▪ Informe de eficacia comparado en diferentes entornos.
Garantizar que el prototipo sea compatible con el sistema operativo para un óptimo funcionamiento.	▪ Pruebas de usabilidad y compatibilidad para determinar posibles correcciones.

1.6. Metodología de investigación

En este proyecto se utiliza la metodología de investigación aplicada, con el fin de utilizar bases de conocimiento existentes para el desarrollo del prototipo; esta consta de tres fases: planeación, ejecución y comunicación de resultados. A continuación, se detallan cada una de estas en relación con los niveles de madurez tecnológica del proyecto.

Fase 1: Planeación

- TRL 1 - Observación de los principios básicos
  - Realizar una revisión extensa de la literatura sobre ransomware, técnicas de detección de comportamientos anómalos, y metodologías de protección en sistemas operativos, especialmente Windows 10.
  - Identificación de los principios fundamentales de los ataques de ransomware y los mecanismos de detección existentes.
- TRL 2 - Formulación del concepto tecnológico
  - Formulación del concepto y diseño preliminar del sistema de detección temprana.
  - Definición de los objetivos específicos y la arquitectura general del sistema, incluyendo los módulos principales que lo compondrán (monitoreo de archivos, supervisión de procesos, interfaz de usuario, etc.).

Fase 2: Ejecución

- TRL 3 - Prueba experimental del concepto
  - Desarrollo de prototipos iniciales de los módulos identificados.
  - Realización de pruebas de concepto en entornos controlados para validar la funcionalidad básica de los componentes del sistema.
- TRL 4 - Validación de la tecnología en el laboratorio
  - Integración de los diferentes módulos del sistema en un entorno de laboratorio.
  - Evaluación del prototipo en entornos relevantes que simulen condiciones operativas reales.
- TRL 5 - Validación de la tecnología en entorno pertinente
  - Realización de pruebas para identificar y corregir posibles fallos, y ajustar los parámetros del sistema para mejorar su eficacia.
  - Validación de los componentes y subsistemas integrados mediante pruebas detalladas en condiciones controladas, asegurando la correcta interacción entre ellos.

Fase 3: Comunicación de resultados

- TRL 6 - Demostración en el entorno pertinente
  - Demostración del sistema en un entorno operativo real o lo más cercano posible.
  - Evaluación de la eficacia del sistema en la detección de comportamientos anómalos y la identificación de ataques de ransomware en entornos de prueba controlados.
  - Recopilación de datos y feedback para realizar ajustes finales.

Según lo redactado por el Ministerio de Ciencia, Tecnología e Innovación en el documento de NIVELES DE MADUREZ TECNOLÓGICA (TRL) Y DE MANUFACTURA (MRL) [26], este proyecto se desarrollará hasta el nivel de maduréz tecnológica (TRL) 6.

1.7. Metodología de desarrollo de software

La metodología de desarrollo de este proyecto fue elegida teniendo en cuenta ciertos criterios de evaluación, con el fin de seleccionar la mejor. Para la elección, se tuvieron en cuenta las siguientes metodologías:

- Waterfall (cascada).
- Desarrollo ágil.
- Programación extrema (XP).

En los criterios de evaluación, se tienen los siguientes puntos:

- Documentación.
- Comunidad.
- Popularidad.
- Curva de aprendizaje.

A continuación, se presentará la relación de estos conceptos en una tabla para su estimación.

Tabla 1.2: Metodologías a evaluar

Metodología	**Documentación**	**Comunidad**	**Popularidad**	**Curva de aprendizaje**	**TOTAL**
Waterfall	4	3	3	2	**12**
Desarrollo ágil	5	5	5	4	**19**
XP	4	4	4	3	**15**

Fuente: Elaboración propia

Luego de una evaluación exhaustiva de las metodologías de desarrollo de software seleccionadas, puntuando cada criterio de 0 a 5, se puede concluir que la metodología más adecuada para el proyecto es el *desarrollo ágil*. Esta metodología ha ganado una gran aceptación en la comunidad en los últimos años y ofrece una variedad de marcos de trabajo adaptables a diferentes necesidades y contextos.

1.8. Metodología de gestión de actividades

La metodología de gestión de actividades que se desea implementar para este proyecto es la metodología ágil Kanban inventada por la empresa de automóviles Toyota. Esta metodología está basada en la filosofía de mejora continua, por ello figura un flujo constante de trabajo, del que se extraen las tareas pendientes en un grupo de trabajo que se autogestiona y realiza el trabajo de acuerdo a sus capacidades. Esta estrategia aumenta el liderazgo en amplios niveles ya que no establecen roles para los miembros, lo cual para términos de este proyecto es conveniente, pues el número de integrantes es una persona más el director del proyecto.

La metodología ágil Kanban se implementa mediante tableros, los cuales contienen en un estado básico 5 columnas: Backlog, tareas por hacer, en progreso, bloqueadas y finalizadas. Por estas columnas se desplazan diferentes tarjetas las cuales individualmente encierran una tarea. Para la columna de tareas en progreso se establece un límite con la intención de no sobrecargar al equipo de trabajo.

Figura 1.2: Tablero Kanban. **Fuente: Elaboración propia.**

Capítulo 2
Marco Referencial

2.1. Marco Teórico

2.1.1. Teoría de la Información de Shannon y entropía

La **Teoría de la Información**, formulada por Claude Shannon (1948), ofrece una base formal para cuantificar la información en términos de probabilidad e incertidumbre. Según Shannon, el "valor informacional" de un mensaje se relaciona con lo sorprendente de su contenido: eventos muy probables aportan poca información, mientras que sucesos improbables aportan mucha [32].

Formalmente, la *entropía de Shannon* mide la incertidumbre promedio de una fuente aleatoria. Dada una variable aleatoria discreta X con valores x_i y probabilidades $p_i = P(X = x_i)$, la entropía se define como:

$$H(X) = -\sum_i p_i \log_2 p_i \tag{2.1}$$

Shannon demostró que esta medida representa el límite teórico de compresión de datos de una fuente sin pérdida de información. En el contexto de criptografía, se busca generar claves con una alta entropía (poco predecibles) para así reforzar la seguridad.

En el contexto de detección de *ransomware*, la entropía de un archivo puede indicar actividad sospechosa; los archivos originales tienen una estructura más predecible (menor entropía), mientras que las versiones cifradas de estos archivos durante o después de un ataque de ransomware representan una entropía mucho mayor. Este cambio tan grande en el valor de la entropía puede ser monitoreado para detectar ataques de ransomware en tiempo real.

2.1.2. Modelo REDemption (Kharraz y Kirda, 2017)

El modelo propuesto por Kharraz y Kirda (2017) **REDemption**, es un sistema de defensa en tiempo real diseñado para detectar amenazas de ransomware [16]. Este modelo opera mediante el monitoreo continuo de las operaciones de entrada/salida (E/S) realizadas sobre los archivos de un usuario del sistema operativo, calculando posteriormente un *puntaje de malicia* (*malice score*) para cada proceso activo en el sistema operativo. Para determinar este puntaje de malicia, se tienen en cuenta diferentes características de comportamiento, incluyendo:

- Entropía de lectura/escritura: una gran diferencia en la entropía de bloques leídos y escritos sugiere cifrado.

- Sobrescritura de archivos: un proceso que sobrescribe bloques completos es potencialmente malicioso.
- Eliminación de archivos: borrar archivos originales tras generar versiones cifradas eleva el puntaje de malicia.
- Frecuencia de acceso: operaciones de escritura en rápida sucesión a distintos archivos son sospechosas.

Cada una de estas características recibe un peso específico dentro del modelo y se integra mediante una función lineal que permite al sistema determinar si un proceso particular debe ser suspendido y reportado como potencial ransomware.

2.1.3. Detección basada en llamadas al sistema (Castañaga et al., 2019)

La investigación realizada por Castañaga et al. (2019) propone una estrategia de detección que reune distintos métodos de monitoreo de las llamadas al sistema `read()` y `write()`, funciones nativas del sistema operativo que son utilizadas por los ransomware durante el cifrado de los archivos del usuario [6]. Este modelo se encuentra diseñado para identificar comportamientos anómalos a las llamadas al sistema anteriormente descritas, especialmente cuando ocurre de manera masiva, lo cual representa un patrón comportamental característico de las amenazas de ransomware.

Esta estrategia implementada por Castañaga et al. (2019) presenta una ventaja significativa: puede identificar ransomware incluso cuando no se dispone de información previa sobre su firma digital específica, esto ayuda a detectar nuevas variantes desconocidas de ransomware, ya que no tiene las limitaciones de los métodos tradicionales basados en firmas.

2.2. Estado del arte

A continuación se describe el estado actual en los desarrollos orientados a la detección temprana de ataques de ransomware en el entorno del Sistema Operativo (SO) Windows 19. En este ámbito se han explorado diversas aproximaciones, que van desde sistemas basados en reglas y análisis estadístico hasta métodos impulsados por aprendizaje automático.

2.2.1. Sistemas de detección de anomalías existentes

Métodos basados en reglas: Uno de los enfoques tradicionales para la detección de ransomware se basa en sistemas que emplean reglas predefinidas para identificar comportamientos anómalos. Entre estos destacan:

- Windows Defender Advanced Threat Protection (ATP): Esta plataforma unificada de Microsoft no solo ofrece protección preventiva, sino que también realiza detección post-intrusión, investigación automatizada y respuesta. ATP utiliza un conjunto de reglas de firma para identificar comportamientos sospechosos en Windows 10 [23].
- Sysmon: Se trata de una herramienta de monitoreo del sistema también desarrollada por Microsoft, que permite la configuración de reglas personalizadas para detectar anomalías. Sysmon recopila información detallada sobre la creación de procesos, conexiones de red y modificaciones en los metadatos de archivos, facilitando la identificación de patrones atípicos [23].

Enfoques basados en estadística: Otra línea de investigación ha consistido en el análisis de registros y la detección de anomalías mediante técnicas estadísticas. Algunos ejemplos son:

- OSSEC: Este sistema de detección de intrusiones basado en hosts explota métodos estadísticos para identificar desviaciones en los registros de eventos del sistema, permitiendo detectar accesos y actividades inusuales [30].
- Elastic Stack: Conjunto de herramientas potentes para el análisis y la visualización de datos, que permite el procesamiento y monitoreo en tiempo real de los registros del sistema. Elastic Stack facilita la identificación de patrones anómalos en la actividad del sistema y en otros tipos de datos críticos [9].

Métodos basados en aprendizaje automático: La creciente sofisticación de los ataques de ransomware ha impulsado el desarrollo de sistemas que emplean modelos de aprendizaje automático para detectar comportamientos maliciosos de forma temprana. Entre estos métodos se destacan:

- McAfee Wildfire (Palo Alto): Un sistema de detección de malware que utiliza modelos basados en aprendizaje automático para analizar y clasificar el comportamiento de los archivos, facilitando la identificación de amenazas emergentes [28].

- Adaptive Defense 360 de Panda Security: Esta solución para endpoints utiliza algoritmos de aprendizaje automático para prevenir y detectar ataques de ransomware a través del análisis en tiempo real de patrones de comportamiento [29].

2.3. Antecedentes

2.3.1. Investigaciones previas y brechas identificadas

- Monitoreo en tiempo real (Kharraz & Kirda, 2017) Kharraz y Kirda proponen *Redemption*, un sistema que intercepta todas las operaciones de E/S de archivos en Windows y redirige las escrituras a un buffer protegido, calculando un "malice score" a partir de características como entropía de bloques, sobrescritura de datos y frecuencia de acceso. En pruebas con 29 familias de ransomware, alcanzaron 100 % de detección y solamente 0.8 % de falsos positivos, con una sobrecarga de rendimiento promedio del 2.6 % [16]. Sin embargo, su dependencia de hooks en el kernel de Windows limita su portabilidad a otros sistemas operativos y puede pasar por alto ransomware muy lento o sigiloso que opere por debajo de umbrales de alerta.

- Estrategias con llamadas al sistema (Castañaga et al., 2019) Castañaga *et al.* implementan en Linux un módulo de kernel que cuenta llamadas `read()` y `write()` para detectar patrones masivos de cifrado. Reportan una tasa de detección del 93.3 %, pero observan un elevado número de falsos positivos en procesos legítimos de compresión y edición de archivos, lo que evidencia la necesidad de incorporar métricas adicionales como el tamaño de los bloques modificados o mediciones de entropía [6].

- Análisis estático con YARA (Medhat et al., 2018) Medhat, Gaber y Abdelbaki desarrollan un framework de detección estática basado en reglas YARA que identifican APIs criptográficas y extensiones de archivo asociadas a ransomware. Logran una precisión del 94.14 % en muestras conocidas [22], pero su enfoque falla contra muestras ofuscadas o empaquetadas, lo que justifica la combinación con técnicas dinámicas.

- Clasificación dinámica (*EldeRan*, Sgandurra et al., 2016) Sgandurra *et al.* presentan *EldeRan*, que ejecuta muestras en sandbox y extrae características de comportamiento en fase temprana, tales como llamadas a archivos y cambios en el registro. Entrenando modelos de machine learning, obtienen un área bajo la curva ROC de 0.995 en un conjunto de 582 ransomware y 942 benignos [31]. Sin embargo, su uso de entornos virtualizados dedicados dificulta la integración en sistemas de producción.

- Detección basada en APIs y ML (Almousa et al., 2021) Almousa, Basavaraju y Anwar analizan el ciclo de vida del ransomware en Windows y extraen patrones de llamadas a APIs criticas en un entorno sandbox. Con estos datos entrenan varios algoritmos de machine learning, logrando altas tasas de detección, pero no investigan la latencia real en entornos Windows 10 y dependen de conjuntos de datos etiquetados [2].

- Boundary pre-encryption y ML (Zakaria et al., 2024) Zanoramy *et al.* definen el "límite pre-encryption" como el punto antes de que el ransomware inicie el cifrado en masa. Combinan detección por firmas (SHA-256) con clasificadores de ML sobre llamadas API pre-cifrado, alcanzando un AUC de 0.951 con regresión logística. No obstante, su éxito depende de una correcta demarcación manual de la fase pre-encryption y no aborda técnicas anti-monitoreo avanzadas [36].

- Análisis estático por CNN (Manavi & Hamzeh, 2020) Manavi y Hamzeh convierten los primeros 1024 bytes del encabezado PE de ejecutables en imágenes, y aplican una red neuronal convolucional, logrando una precisión del 93.33 % [20]. Su ventaja es la detección sin ejecutar el malware, pero es vulnerable a ofuscación o empaquetado que modifiquen el encabezado y no captura ninguna señal de comportamiento dinámico.

2.3.2. Evolución de los ataques de ransomware

El ransomware ha recorrido un largo camino desde sus orígenes rudimentarios hasta convertirse en una amenaza cibernética sofisticada y global. Sus primeros indicios se remontan a la década de 1980, cuando los ataques eran rudimentarios y, en muchos casos, más experimentales que lucrativos. No obstante, fue a partir de la década de 2010 cuando este tipo de malware comenzó a ganar terreno en los ciberataques, impulsado por el incremento en la disponibilidad de herramientas de desarrollo de malware y la expansión de internet.

Primeros ataques

Los primeros ejemplos de ransomware eran relativamente simples, centrándose en bloquear el acceso a la información o el sistema sin utilizar técnicas de cifrado robustas. Sin embargo, la evolución de la criptografía aplicada a estos ataques transformó al ransomware en una amenaza que podía impedir el acceso a datos críticos, forzando a las víctimas a pagar rescates elevados para recuperar sus archivos.

Ataques notables

A lo largo de la última década se han producido varios ataques notables que han demostrado la capacidad de los cibercriminales para adaptar y sofisticar sus métodos. Entre los ataques más destacados se incluyen:

- PETYA: Descubierto inicialmente en marzo de 2016, PETYA sobrescribe el registro de arranque (MBR), impidiendo que el sistema operativo se inicie de manera normal. Este ataque ha sido particularmente dañino, ya que se aprovecha de vulnerabilidades en el manejo del MBR y se ha propagado a través de medios legítimos como servicios de almacenamiento en la nube, por ejemplo, Dropbox [27].

- WannaCry: Este crypto-ransomware se caracteriza por cifrar los archivos críticos de la víctima y exigir un rescate para su descifrado. WannaCry logró notoriedad global al propagarse rápidamente por múltiples redes y sistemas, aprovechando una vulnerabilidad en el protocolo SMB de Windows. Su impacto demostró la capacidad destructiva y el alcance global de los ataques de ransomware [6].

- SamSam: Detectado en marzo de 2016, SamSam difiere de otros ataques en su método de propagación, ya que explota vulnerabilidades en servidores que no cuentan con los parches de seguridad correspondientes. En lugar de utilizar técnicas comunes como URL maliciosas

o spam, los actores responsables de SamSam se enfocan en comprometer sistemas críticos para ampliar su red de máquinas infectadas [27].

- CryptXXX: Además, Castañaga et al. han señalado el ataque CryptXXX como otro ejemplo paradigmático dentro de los ransomware. Este malware es conocido por cifrar de forma agresiva los archivos de los usuarios, dejando pistas mínimas sobre el método de recuperación y, en ocasiones, incorporando métodos de encriptación que complican significativamente la remediación sin la clave privada. La sofisticación del CryptXXX reside en su capacidad de adoptar técnicas de cifrado robustas y en su rápida propagación, lo que lo convierte en una amenaza especialmente peligrosa en entornos corporativos y de infraestructura crítica.

La evolución y diversificación de estos ataques demuestran que, a medida que los métodos de detección y las medidas de seguridad han mejorado, los cibercriminales han respondido desarrollando variantes más complejas y evasivas. Este contexto histórico y técnico subraya la necesidad de seguir investigando y desarrollando soluciones efectivas para la detección y mitigación del ransomware.

Capítulo 3
Alcance

3.1. Alcance del proyecto

El presente proyecto se enfoca en el diseño, desarrollo e implementación de un prototipo de sistema para la detección temprana por medio de comportamientos anómalos de ataques de *ransomware* en el sistema operativo Windows 10. El enfoque del sistema se basa en el monitoreo constante de los procesos en ejecución del sistema operativo, con el objetivo de identificar comportamientos anómalos que puedan anticipar un intento de cifrado, consecuencia de un ataque de *ransomware*. Mediante el análisis de comportamientos en tiempo real, el sistema busca medidas de respuesta automática que detengan el ataque antes de que se concrete la pérdida de información.

El prototipo tiene como base una integración de técnicas inspiradas en los modelos propuestos por Castañaga et al. (2019) [6] y Kharraz y Kirda (2017) [16], implementandolos junto a un *framework* que se pueda ejecutar en Windows 10.

Para el desarrollo de este proyecto, se debe tener en cuenta que existen ciertos supuestos y limitaciones que pueden influir en el alcance y la efectividad del prototipo. A continuación, se detallan los supuestos y restricciones que se han considerado:

3.1.1. Supuestos

- **Disponibilidad tecnológica:** Se asume que los equipos que ejecutarán el prototipo cumplen con los requisitos mínimos de hardware y software para su correcto funcionamiento sobre Windows 10.
- **Permisos suficientes:** Se contempla que el sistema contará con los privilegios necesarios para monitorear y analizar los procesos en tiempo real, es decir, tendrá acceso de administrador en el sistema operativo.
- **Muestras representativas:** Durante las pruebas, se utilizarán muestras de ransomware que reflejan escenarios y comportamientos similares a los que se pueden presentar en situaciones reales dentro de Windows 10.
- **Colaboración de los usuarios:** Se da por hecho que los usuarios no restringirán el funcionamiento del prototipo y mantendrán prácticas básicas de seguridad en el entorno de prueba.
- **Entorno controlado:** Inicialmente, todas las pruebas y evaluaciones del sistema se realizarán en un entorno controlado o de laboratorio, lo que permite gestionar los posibles riesgos antes de pensar en una aplicación real.

3.1.2. Restricciones

- **Tiempo de desarrollo:** El tiempo de desarrollo para este proyecto está limitado a ocho (8) meses, lo que equivale a dos (2) periodos academicos.
- **Enfoque exclusivo en Windows 10:** El diseño y desarrollo del prototipo ha sido pensado sólo para este sistema operativo, por lo que no se garantiza su funcionamiento en versiones diferentes o en otras plataformas.
- **Límite en la detección:** El prototipo está centrado en el monitoreo de comportamientos similares al de un *ransomware* de los procesos activos; así, otros tipos de ciberataques — como los que explotan redes o hardware — quedan fuera del alcance de este proyecto.
- **Cobertura de las respuestas automáticas:** La respuesta automática está limitada por los algoritmos implementados; en consecuencia, ataques novedosos o técnicas aún no estudiadas podrían no ser identificados ni detenidos de manera eficaz.
- **Consumo de recursos:** El análisis constante del sistema puede tener un impacto en el rendimiento, especialmente en equipos con bajos recursos.
- **Adaptación de modelos previos:** Se reconoce que la implementación de las propuestas de Castañaga et al. (2019) y Kharraz y Kirda (2017) podría requerir ciertos ajustes técnicos, por lo que la solución desarrollada puede diferir parcialmente de los modelos originales.

Capítulo 4

Análisis de Fundamentos

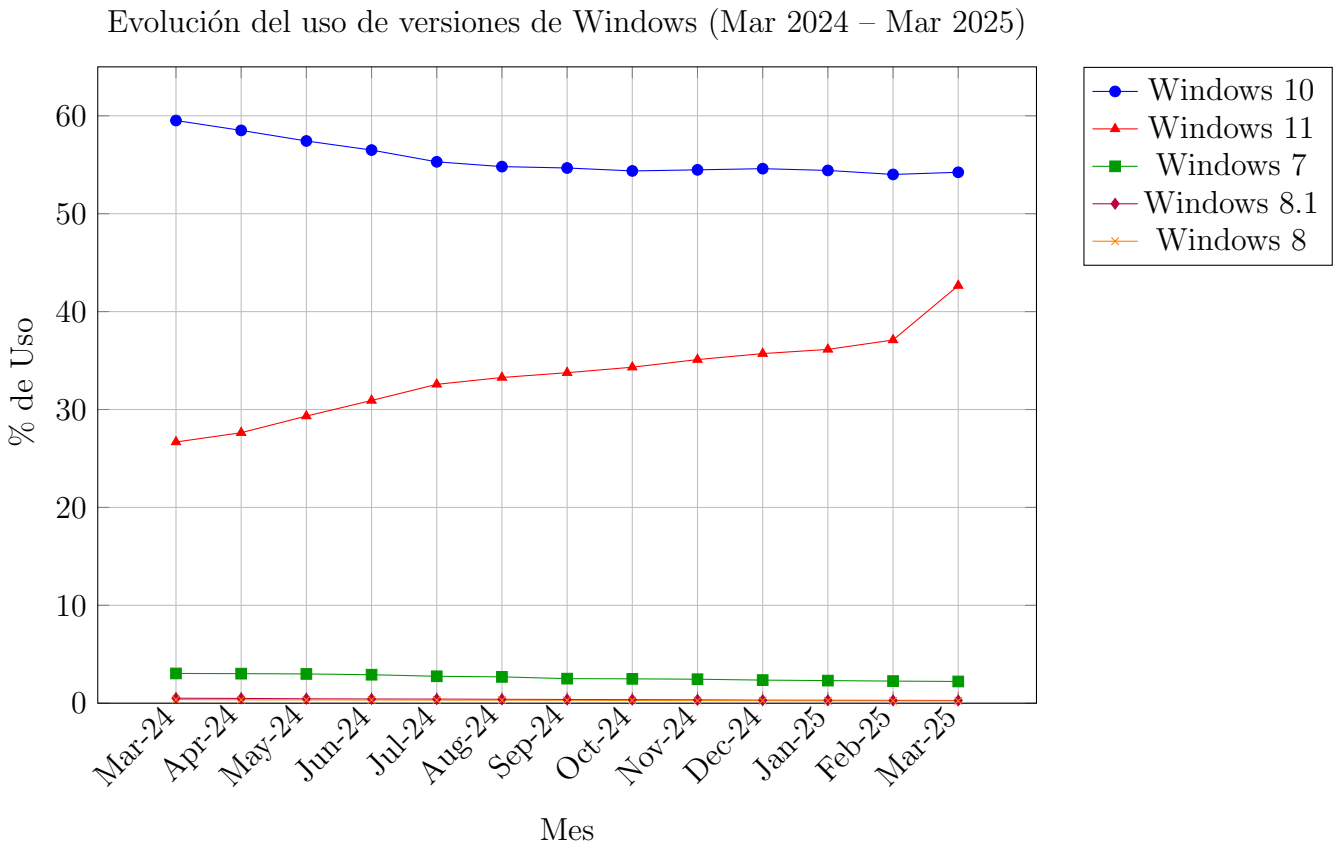
4.1. Selección de la versión del sistema operativo

La presente investigación se centra en el sistema operativo Microsoft Windows 10, debido a su amplia adopción y relevancia en el mercado, superando incluso a las versiones más recientes de Windows. Actualmente, la última actualización disponible es la versión 22H2 [25], lanzada el 18 de octubre de 2022, y con soporte oficial vigente hasta el 14 de octubre de 2025.

La elección de Windows 10 se basa en la significativa cantidad de usuarios que lo utilizan, lo que lo convierte en un objetivo estratégico para los estudios de seguridad.

A continuación, se presenta el gráfico extraído de la página StatCounter Global Stats, que muestra la distribución de uso de las versiones de Windows a nivel mundial. Este gráfico proporciona una visión clara de la popularidad de Windows 10 como sistema operativo en el año 2024.

Figura 4.1: Cuota de mercado de versiones de Windows entre marzo 2024 y marzo 2025. **Fuente: [34]**



En la Figura 4.1 se puede observar que Windows 10 es la versión más utilizada hasta la fecha de corte de este documento (marzo de 2025), con una cuota de mercado del 54.23 % respecto a las otras versiones del sistema operativo. A partir de esta información, se justifica la decisión de seleccionar la versión Windows 10 para el desarrollo del prototipo.

4.2. Funcionamiento de un *ransomware*

Para el inicio de la investigación de este trabajo, se consideró estudiar literatura acerca de las entradas que tiene un *ransomware* para poder infectar un sistema; esto con el proposito de tener una base para empezar a desarrollar el modelo del prototipo de detección temprana.

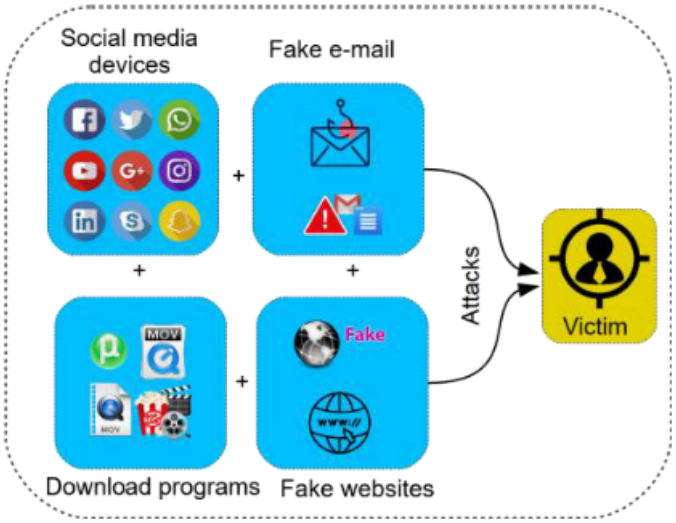
Entradas y/o vulnerabilidades que utiliza un ransomware

Vectores de infección

Un *ransomware* puede llegar a un sistema a través de varios vectores de infección. Según diferentes autores como lo son [36], [31], [19], [10] y [4] el vector de infección en el que convergen y el más común en las victimas en el **phishing**. Según Malik et al., utilizan un correo electrónico sospechoso puede contener un enlace web que aloje una descarga sospechosa o un archivo adjunto con funciones de gestor de descargas de Internet. Si de alguna manera el receptor del correo electrónico es engañado, el ransomware se descarga directamente en una estación de trabajo [19].

Tambien se pueden encontrar otros vectores de infección, como los mencionados en la investigación de Kara, I. y Aydos, M.:

Figura 4.2: Diferentes vectores de infección de un *ransomware*. Fuente: Kara, I. y Aydos, M. [14]



En la Figura 4.2 se pueden observar los diferentes vectores de infección de un *ransomware*, que se detallan a continuación:

- Redes sociales
- Correos electrónicos falsos
- Sitios web falsos
- Descarga de programas de fuentes desconocidas

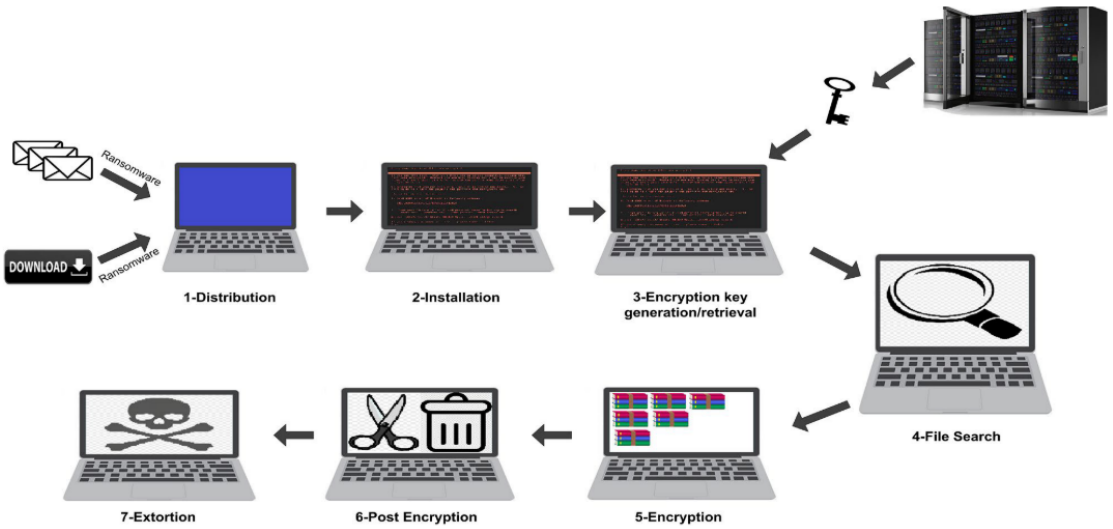
Tipos de *ransomware*

En la investigación realizada por Alqahtani et al. [4] se menciona que existen dos tipos principales de *ransomware*: *crypto-ransomware* y *locker-ransomware*, cada uno con ciertas características que los diferencian:

Locker-ransomware. El objetivo del *locker-ransomware* es bloquear el acceso al sistema operativo, impidiendo que el usuario pueda utilizar el equipo. Por el enfoque que utiliza, este tipo de ransomware no cifra ningún archivo ni afecta a los datos almacenados en el dispositivo [4], por lo cual los datos se pueden recuperar por medio de otros métodos.

Crypto-ransomware. El objetivo del *crypto-ransomware* es cifrar los archivos o incluso toda la base de datos con la que interactúa el sistema [4]. Este tipo de ransomware es el más común y el más peligroso, ya que los datos cifrados no pueden ser recuperados sin la clave de descifrado.

Figura 4.3: Etapas habituales de un *crypto-ransomware*. Fuente: UROOJ, M., et al. [35]



Basado en la Figura 4.3 se pueden distinguir las diferentes fases de un *ransomware*.

1. **Distribución**. El ataque comienza cuando el ransomware logra llegar al sistema víctima. Esto puede ocurrir a través de correos de *phishing*, descargas engañosas, o mediante la explotación de vulnerabilidades en servicios expuestos a Internet. Basta con que el usuario haga clic en un enlace malicioso o abra un archivo adjunto para que se dispare la cadena de infección.

2. **Instalación**. Una vez ejecutado, el malware se instala discretamente en el equipo: se copia en el disco, configura mecanismos de persistencia (por ejemplo, entradas en el Registro de Windows) y, si es posible, intenta desactivar el antivirus o el firewall. Todo este proceso se realiza en segundo plano para evitar levantar sospechas.

3. **Obtención o generación de claves**. En esta etapa, el ransomware obtiene la clave que usará para cifrar los archivos. Esta puede ser generada localmente o descargada desde un servidor remoto controlado por el atacante. Esta clave es crucial: sin ella, la recuperación de los archivos se vuelve prácticamente imposible.

4. **Búsqueda de archivos**. El malware comienza a escanear el sistema en busca de archivos importantes: documentos, hojas de cálculo, correos, fotos y más. No se limita al disco local; también analiza unidades de red y carpetas compartidas. Aunque esta actividad es intensa, muchas veces pasa desapercibida porque se ejecuta de forma silenciosa y se camufla entre procesos legítimos del sistema [35].

5. **Cifrado**. Esta es la fase más destructiva. El ransomware sobreescribe los archivos originales con versiones cifradas, haciendo uso de llamadas como `CreateFile`, `WriteFile` y `SetFileInformation`. La entropía del contenido escrito se dispara, lo cual es un indicador técnico claro de que algo anómalo está ocurriendo. Este comportamiento es precisamente lo que nuestro sistema de detección se esfuerza en identificar.

6. **Acciones posteriores al cifrado**. Una vez que los archivos han sido cifrados, muchas variantes de ransomware intentan impedir cualquier posibilidad de recuperación. Esto incluye borrar copias de seguridad del sistema mediante comandos como `vssadmin delete shadows`, eliminar registros o destruir archivos temporales.

7. **Extorsión**. Finalmente, el malware revela sus intenciones: se muestra una nota de rescate, se cambia el fondo de pantalla o incluso se bloquea la sesión del usuario. Se exige un pago—generalmente en criptomonedas—a cambio de la clave de descifrado. El mensaje suele incluir instrucciones detalladas para realizar la transacción y amenazas si no se cumple el plazo.

4.3. Modelos y técnicas de detección de *ransomware* aplicables

En la Figura 4.1 se detalla la revisión de 20 articulos científicos, publicados en las plataformas IEEE Xplore, Scopus y Google Scholar, con el fin de recolectar la información necesaria para el

desarrollo del prototipo. Esta información se utilizará más adelante para la elaboración del modelo de detección de comportamientos anómalos.

N.°	Título	Autores	Año	Palabras Clave	Resumen
1	**2entFOX**: Framework for high survivable ransomware detection	Ahmadian, M. M.; Shahriari, H. R.	2016	Ransomware, behavioral analysis, high survivability	Marco para detección de ransomware con alta supervivencia basado en análisis de comportamiento y múltiples estrategias de monitoreo.
2	**API-Based Ransomware Detection** using ML threat models	Almousa, M.; Basavaraju, S.; Anwar, M.	2021	API calls, machine learning, Random Forest	Enfoque de detección mediante llamadas API y ML. Random Forest demostró mayor precisión en identificación de amenazas.
3	**Crypto-Ransomware Early Detection** (CRED) Model	Alqahtani, A.; Gazzan, M.; Sheldon, F. T.	2020	Early detection, deep learning, vector space	Modelo CRED combina aprendizaje profundo para detectar patrones antes del cifrado con alta precisión.
4	**Ransomware in Windows Platform**: An Overview	Alzahrani, A.; et al.	2017	Windows ransomware, crypto-ransomware, countermeasures	Revisión de ransomware en Windows: clasificación de tipos, vectores de infección y métodos de defensa.
5	**Estrategias de detección** de ransomware de cifrado	Castañaga, I. et al.	2019	Detección ransomware, análisis comportamiento	Análisis comparativo de estrategias basadas en comportamiento y firmas digitales.
6	**Early mitigation** of CPU-optimized ransomware	Enomoto, S.; et al.	2024	CPU monitoring, encryption instructions	CryptoSniffer: mecanismo para detectar ransomware que usa instrucciones CPU específicas.
7	**Windows malware detection** using machine learning	Hilabi, R.; Abu-Khadrah, A.	2024	XGBoost, Random Forest, PE files	Modelo híbrido XGBoost + Random Forest para detección de malware en Windows.

N.°	Título	Autores	Año	Palabras Clave	Resumen
8	**Enhanced detection** of obfuscated malware	Hossain, Md. A.; Islam, Md. S.	2024	Memory dumps, obfuscated malware	Framework con preprocesamiento para detectar malware ofuscado en volcados de memoria.
9	**Cyber Fraud**: Crypto-Ransomware analysis	Kara, I.; Aydos, M.	2020	Fraud detection, data mining	Análisis de fraudes y comportamiento técnico de ransomware usando minería de datos.
10	**Ransomware Detection** using Random Forest	Khammas, B. M.	2020	Random Forest, PE headers	Modelo Random Forest sobre encabezados PE para detección efectiva.
11	**Redemption**: Real-Time ransomware protection	Kharraz, A.; Kirda, E.	2017	Real-time protection, system recovery	Sistema de protección en tiempo real que revierte cambios no autorizados.
12	**Systematic Ransomware Detection** techniques study	Lee, S.-J.; et al.	2022	EDR, static/dynamic analysis	Revisión de métodos estáticos, dinámicos y ML para detección sistemática.
13	**Ransomware Behavior and Characteristics** survey	Malik, N. A.; et al.	2024	Ransomware behavior, evasion techniques	Encuesta sobre comportamientos, vectores de ataque y técnicas de evasión.
14	**Ransomware Detection** using CNN and PE headers	Manavi, F.; Hamzeh, A.	2020	CNN, PE analysis, feature extraction	CNN sobre encabezados PE para detección estática rápida y efectiva.
15	**Static Detection** using LSTM and PE headers	Manavi, F.; Hamzeh, A.	2021	LSTM, sequential analysis	LSTM para analizar relaciones secuenciales en campos PE.
16	**Static-Based Framework** for ransomware detection	Medhat, M.; et al.	2018	YARA rules, static analysis	Framework de análisis estático y ML para detección pre-cifrado.
17	**Analysis Technique** for ransomware detection	Mukesh, S. D.	2018	Heuristic analysis, Tesla Crypt	Enfoque heurístico mediante análisis de encabezados PE sin ejecución.

N.°	Título	Autores	Año	Palabras Clave	Resumen
18	**Automated Dynamic Analysis** of ransomware	Sgandurra, D.; et al.	2016	Dynamic analysis, behavioral detection	EldeRan: sistema basado en comportamiento con 582 muestras de ransomware.
19	**Adaptive Pre-Encryption** detection model	Urooj, U.; et al.	2021	Pre-encryption, adaptive systems	Modelo adaptativo combinando análisis estático y dinámico.
20	**Early Detection** using ML and pre-encryption boundary	Wira Zanoramy et al.	2024	Pre-encryption boundary, ML	Modelo ML que delimita zonas críticas para detección temprana con alta precisión.

Tabla 4.1: Revisión sistemática de literatura sobre técnicas de detección de ransomware. **Fuente:** Elaboración propia basada en análisis de 20 investigaciones relevantes (2016-2024).

4.4. Elementos seleccionados

4.4.1. Técnicas de detección seleccionadas

A partir de la revisión de literatura realizada, se seleccionaron las siguientes técnicas de detección de ransomware, para implementarlas junto al modelo base del prototipo.

1. **Puntuación de malicia (Malice Score):** Basada en el modelo de Kharraz & Kirda [16], esta técnica asigna una puntuación numérica a los procesos según su comportamiento, considerando factores como la frecuencia de operaciones de I/O y patrones de acceso a archivos.

2. **Conteo de operaciones read/write:** Implementando la metodología propuesta por Castañaga et al. [6], se monitorean y cuantifican las llamadas al sistema de lectura y escritura para detectar patrones anómalos característicos del ransomware.

3. **Análisis de firmas criptográficas:** Generación y comparación de hashes SHA-256 para detectar modificaciones no autorizadas en archivos críticos del usuario.

4.4.2. Tipo de *ransomware* seleccionado para analizar con el prototipo

El prototipo se centra únicamente en los **crypto-ransomware**, esto justificado en los siguientes puntos:

- Es el tipo de ransomware más común y peligroso, como se mencionó anteriormente en el apartado de tipos de *ransomware* (como se detalla en la sección 4.2)

- Por su naturaleza, este tipo de *ransomware* raliza un cifrado irreversible, lo que hace que los archivos sean inaccesibles cuando se completa el ataque; se pueden observar estas fases en el gráfico "fases de un *ransomware*" (como se observa en la Figura 4.3)
- El proceso de cifrado (fase número 5, observada en la Figura 4.3) genera llamadas a la API del sistema operativo, generando registros como: eventos de E/S y cambios de extensión. Estos comportamientos pueden ser cuantificados y tenidos en cuenta a la hora de detectar el ransomware.

4.4.3. Herramienta de simulación de *ransomware*

Para el desarrollo del prototipo de detección, se seleccionó la herramienta de código abierto **RanSim** como simulador de ransomware. La herramienta se escogió debido a que es necesario contar con un entorno de pruebas seguro que permita evaluar la efectividad del prototipo sin los riesgos relacionados a un ransomware real. **RanSim** se encuentra dispobible en el anexo (9)

Capítulo 5
Diseño del Prototipo

5.1. Arquitectura del sistema

5.1.1. Arquitectura general

El prototipo de sistema propuesto tiene una estructura por módulos que consta de las siguientes cuatro capas:

1. **Capa de Monitoreo de Sistema**
   FileManager:
   - Vigila en tiempo real el directorio del usuario (`C:\Users\<usuario>`), donde Windows almacena archivos manipulados como documentos, imágenes y configuraciones.
   - Genera y actualiza firmas SHA-256 para cada fichero.
   - Persiste estas firmas en un archivo cifrado accesible solo mediante credenciales internas.
2. **Capa de Detección de Anomalías**
   AnomalyDetector:
   - Recibe eventos de creación, lectura, modificación y eliminación desde el FileManager.
   - Calcula una puntuación de "malicia" inspirada en Redemption [16].
   - Aplica conteo de llamadas a `read/write` según el modelo de Castañaga et al. [6].
   - Determina si un proceso es sospechoso revisando su patrón de acceso a los archivos.
3. **Capa de Registro y Notificaciones**
   LoggerService y NotificationsService:
   - El servicio **Logger** registra cada evento registrado por el FileManager.
   - El servicio **Notifications** emite alertas al usuario mediante notificaciones tipo pop-up y muestra detalles del proceso sospechoso.
4. **Capa de Mitigación**
   QuarantineService:

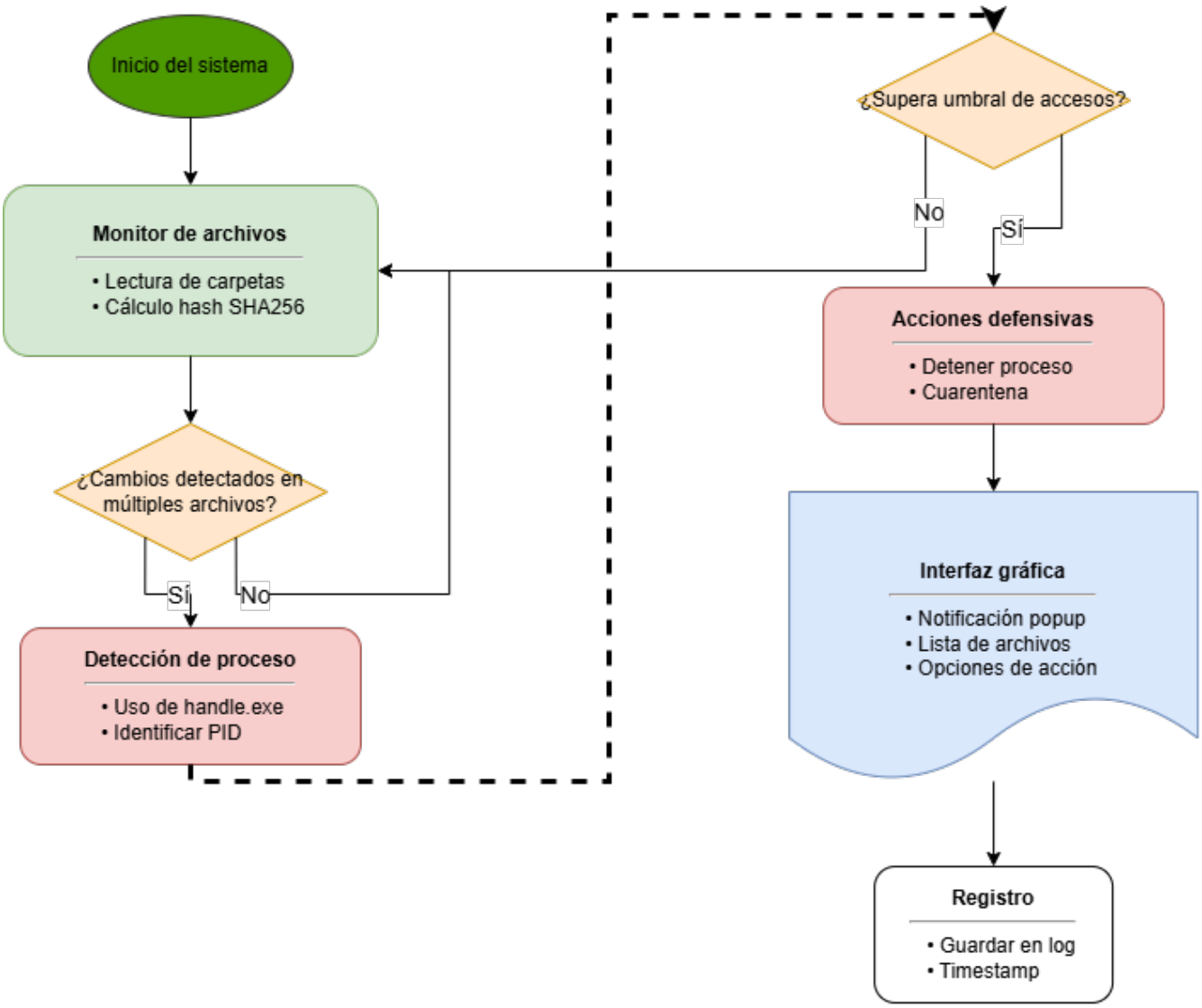
- Termina el proceso sospechoso cuando se determina que es un proceso sospechoso.
- Cifra el ejecutable del proceso.
- Por medio del servicio **Notifications** notifica al usuario sobre la acción tomada.

5.1.2. Flujo de ejecución

1. Inicialización: FileManager carga las firmas SHA-256 y configura el *watcher* sobre `C:\Users\<usuario>`.
2. Cambio detectado: se genera un evento con metadatos (tipo, ruta, tamaño).
3. Evaluación: el detector calcula la puntuación de malicia por cada proceso según:
   - Cantidad de escrituras realizadas en un intervalo de tiempo.
   - Recuento alto del uso de funciones `read/write` del Sistema Operativo.
   - Cambios en la entropía de los archivos.
4. Registro y alerta: si supera el umbral, se muestra alerta sonora.
5. Mitigación: el proceso se detiene, se aísla y se registra el evento.

A continuación, en la Figura 5.1 se puede observar el flujo de ejecución en forma de bloques, para la comprensión del mismo:

Figura 5.1: Flujo de ejecución. **Fuente: Elaboración propia.**



5.2. Modelos de Datos Propuestos

En esta sección se presentan los modelos de datos que utiliza el prototipo de detección para representar y gestionar la información. Se han definido cuatro modelos principales:*Firmas de Archivos Logs*, *Alertas* y *Estadísticas*, cada uno con características específicas que facilitan el monitoreo y análisis del sistema.

5.2.1. Modelo de Firmas

El modelo de firmas se utiliza para almacenar y gestionar las firmas SHA256 de los archivos del sistema operativo. Este modelo funciona como un repositorio que representa por medio de firmas los archivos del usuario.

La estructura de cada registro consta de:

- **Path**: Ruta del archivo en el sistema de archivos del usuario.
- **Hash**: Firma del archivo en formato SHA256.

5.2.2. Modelo de Logs

Para el registro de eventos durante la ejecución del sistema, se implementó un mecanismo basado en archivos de texto plano. El archivo `ErrorLog.txt` funciona como repositorio centralizado donde se almacenan los mensajes clasificados según su nivel de importancia: `INFO`, `WARNING` y `ERROR`.

La estructura de cada registro incluye:

- **Timestamp**: Momento exacto en que se produjo el evento.
- **Level**: Clasificación del mensaje según su criticidad (`INFO`, `WARNING`, `ERROR`).
- **Message**: Descripción detallada del evento registrado.

5.2.3. Modelo de Alertas

Las alertas constituyen el núcleo informativo sobre las amenazas identificadas durante los procesos de escaneo. Cada registro de alerta encapsula información crítica sobre los incidentes de seguridad detectados.

Los componentes de este modelo son:

- **Threat**: Identificación del archivo o patrón de comportamiento clasificado como malicioso.
- **Timestamp**: Momento preciso en que se identificó la amenaza.
- **Action**: Respuesta aplicada por el sistema (opciones incluyen: `Quarantined`, `Deleted`, `Ignored`).

5.2.4. Modelo de Estadísticas

El componente estadístico del sistema captura instantáneas del estado operacional, generando métricas que permiten evaluar el rendimiento y la efectividad de los procesos de detección. Estas mediciones se toman en intervalos regulares para construir un perfil temporal del sistema.

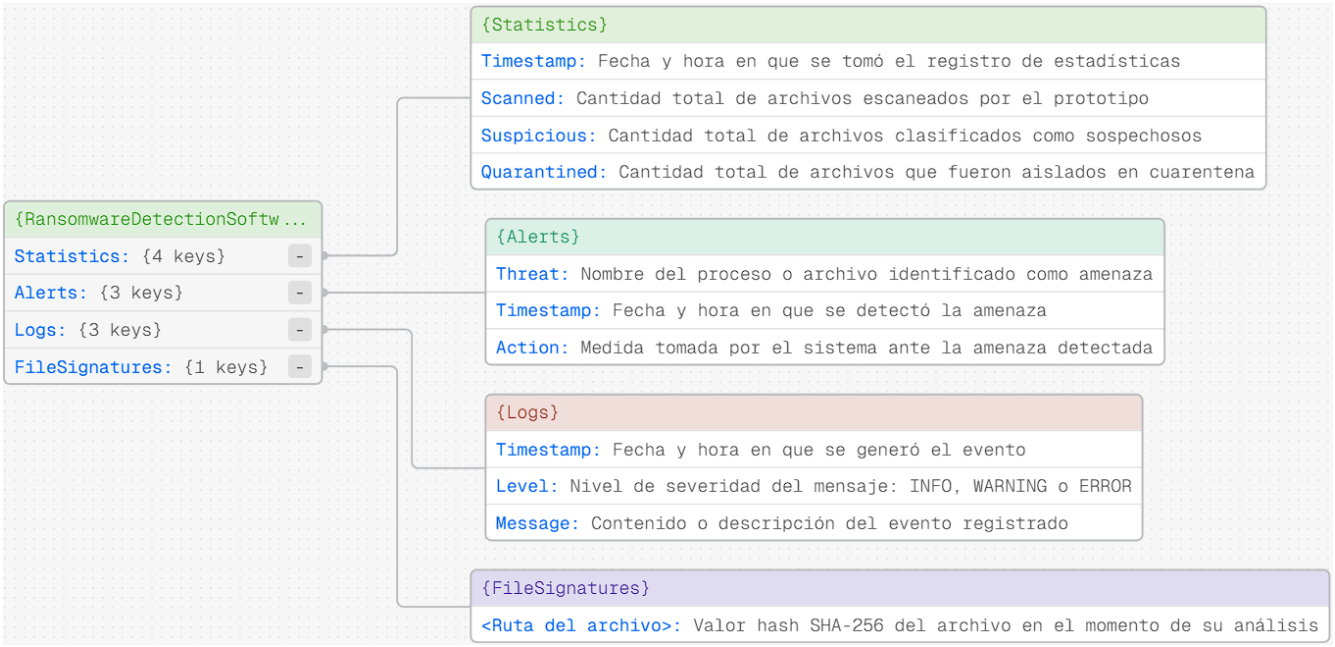
Los elementos que conforman este modelo incluyen:

- **Timestamp**: Momento en que se capturó la información estadística.

- **Scanned**: Total de archivos procesados durante el escaneo.
- **Suspicious**: Número de elementos flagged como potencialmente peligrosos.
- **Quarantined**: Cantidad de archivos aislados por medidas de seguridad.

5.2.5. Representación Gráfica del Modelo de Datos

Figura 5.2: Arquitectura del modelo de datos implementado en la aplicación. **Fuente: Elaboración propia.**



5.3. Diseño de la Interfaz del Prototipo

En este apartado se mostrará la evolución del diseño desde los wireframes iniciales hasta el mockup final implementado en el prototipo.

5.3.1. Componentes Principales del Prototipo

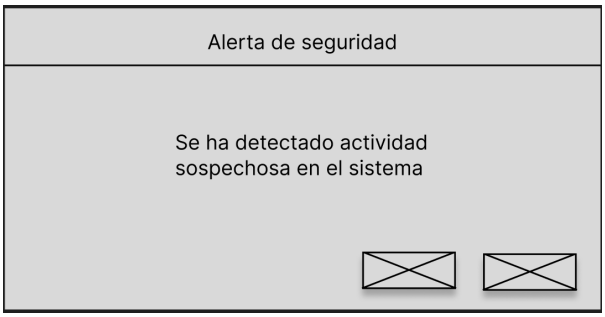
El prototipo final integra cinco componentes fundamentales que evolucionaron desde los wireframes básicos hacia implementaciones funcionales:

Componente de Alertas

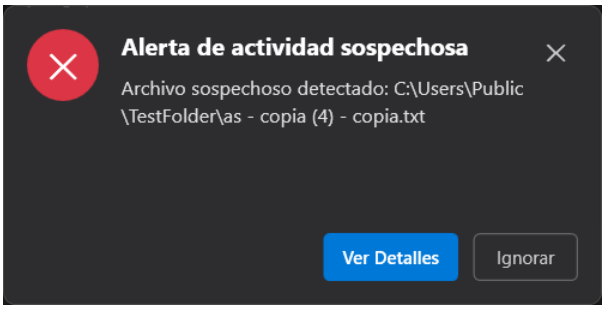
Como se observa en la Figura 5.3, el componente de alertas tiene como objetivo principal notificar al usuario sobre eventos críticos en el sistema. Contiene los siguientes elementos:

- **Icono de Alerta:** Indica la presencia de una alerta.
- **Título:** Describe brevemente el tipo de alerta.
- **Descripción:** Proporciona detalles adicionales sobre la alerta.
- **Acciones:** Botones para tomar acciones específicas, como "Ver más" o "Cerrar".

Figura 5.3: Comparativa entre wireframe y mockup del componente de alertas. **Fuente: Elaboración propia.**



Wireframe del componente de alertas



Mockup del componente de alertas

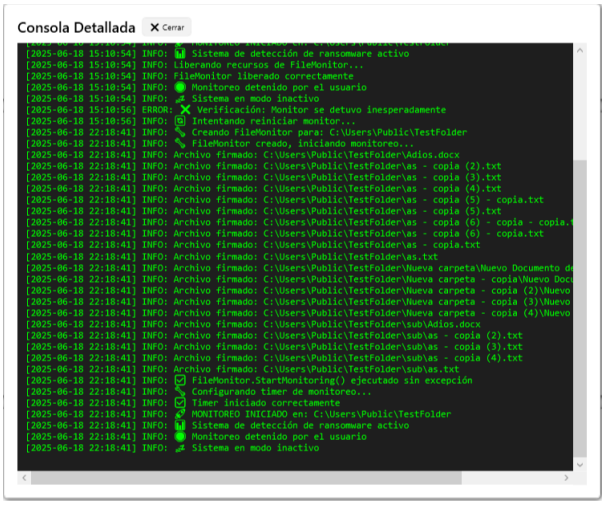
Componente de Consola

Como se observa en la Figura 5.4, el componente de consola tiene como objetivo mostrar los Logs del sistema en tiempo real, para que sean analizados en el entorno de pruebas. Tiene los siguientes elementos:

- **Título:** Describe el componente de consola.
- **Descripción:** Proporciona los Logs a detalle.

Figura 5.4: Comparativa entre wireframe y mockup del componente de consola. **Fuente: Elaboración propia.**

Wireframe del componente de consola



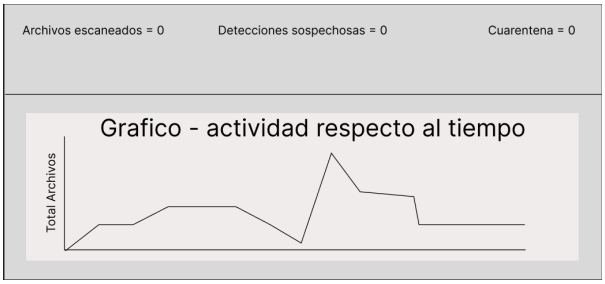
Mockup del componente de consola

Componente de Estadísticas

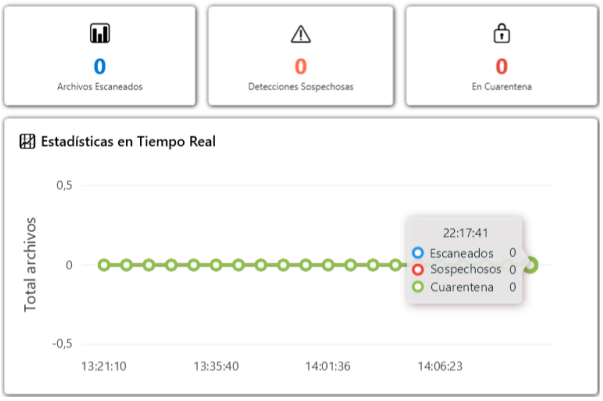
Como se observa en la Figura 5.5, el componente de estadísticas tiene como objetivo mostrar de manera gráfica al usuario los datos recolectados por el monitor de archivos, para visualizar lo que está pasando con el sistema operativo a tiempo real.

- **Paneles de información:** Describen los datos recolectados por el monitor de archivos. Se dividen en: Archivos analizados, Detecciones sospechosas y En Cuarentena.
- **Gráfica:** Muestra la cantidad de archivos analizados a lo largo del tiempo, además de las acciones tomadas por cada uno de ellos.

Figura 5.5: Comparativa entre wireframe y mockup del componente de estadísticas. **Fuente: Elaboración propia.**



Wireframe del componente de estadísticas



Mockup del componente de estadísticas

Componentes de Acciones

Como se observa en la Figura 5.6 y en la Figura 5.7, los componentes de acciones tine el acceso a a distintas herramientas que facilitan al usuario la manipulación de la salida de información que genera el prototipo. Estas acciones se dividen en dos componentes:

- **Inicio de la app:** Aquí se puede encender el prototipo por medio de un botón.
- **Archivos del monitor:**

  Cuarentena: Le permite al usuario dirigirse a la carpeta de cuarentena del prototipo.

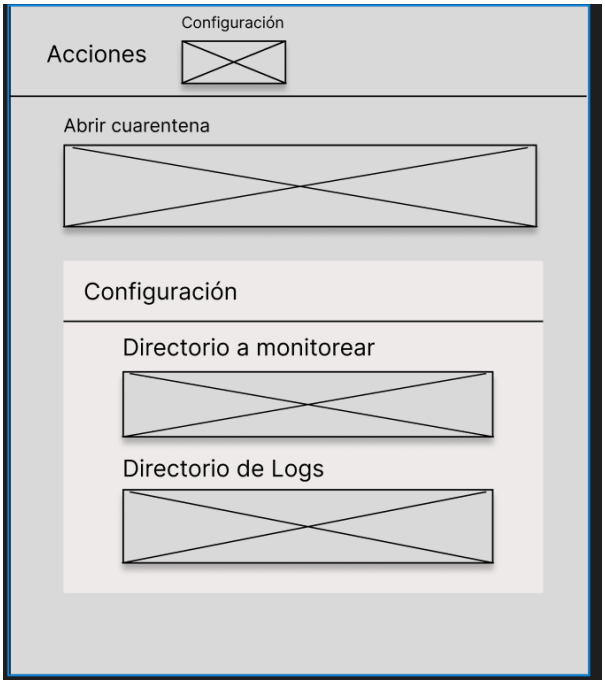
  Configuración: Le permite al usuario seleccionar la carpeta a analizar, además de acceso rápido a los Logs del prototipo.

Figura 5.6: Comparativa entre wireframe y mockup del componente de acciones. **Fuente: Elaboración propia.**

Wireframe del componente de acciones

Mockup del componente de acciones

Figura 5.7: Comparativa entre wireframe y mockup del componente de acciones segunda parte. **Fuente: Elaboración propia.**



Wireframe del componente de acciones segunda parte

Mockup del componente de acciones segunda parte

5.3.2. Ventana Principal del Prototipo

En la Figura 5.8 se muestra el diseño general del prototipo, uniendo todos los componentes anteriormente mencionados en una sola pantalla.

Figura 5.8: Wireframe de la ventana principal. **Fuente: Elaboración propia.**

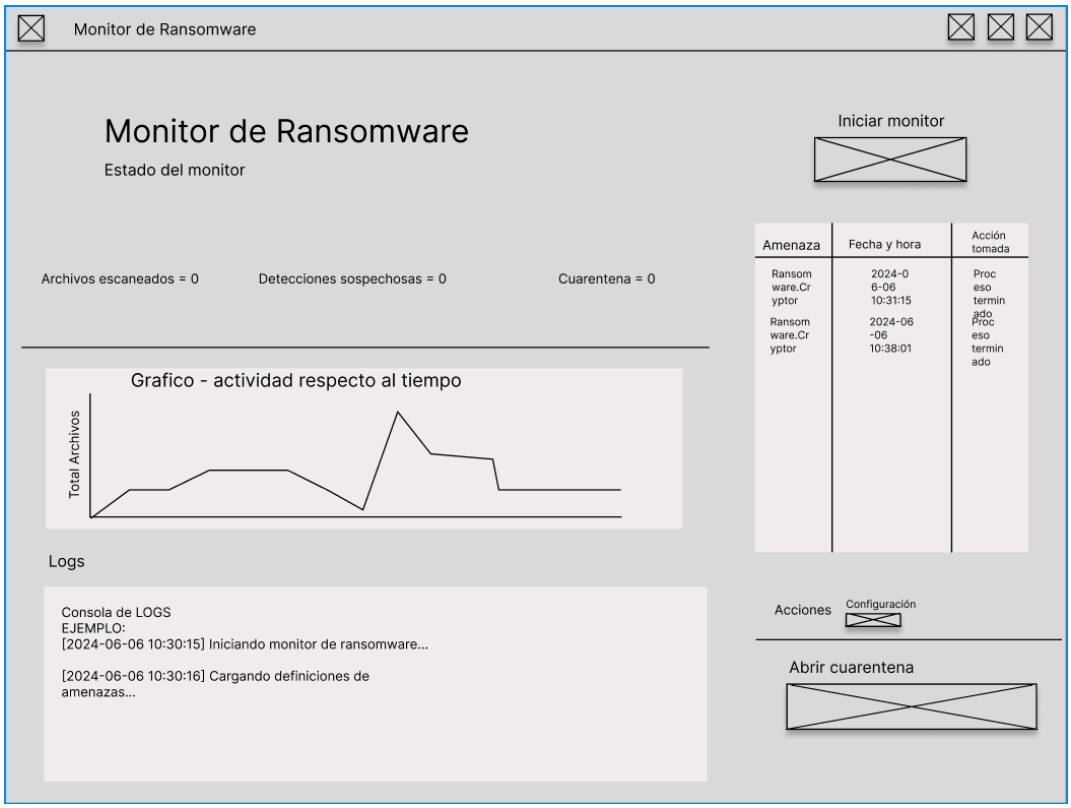
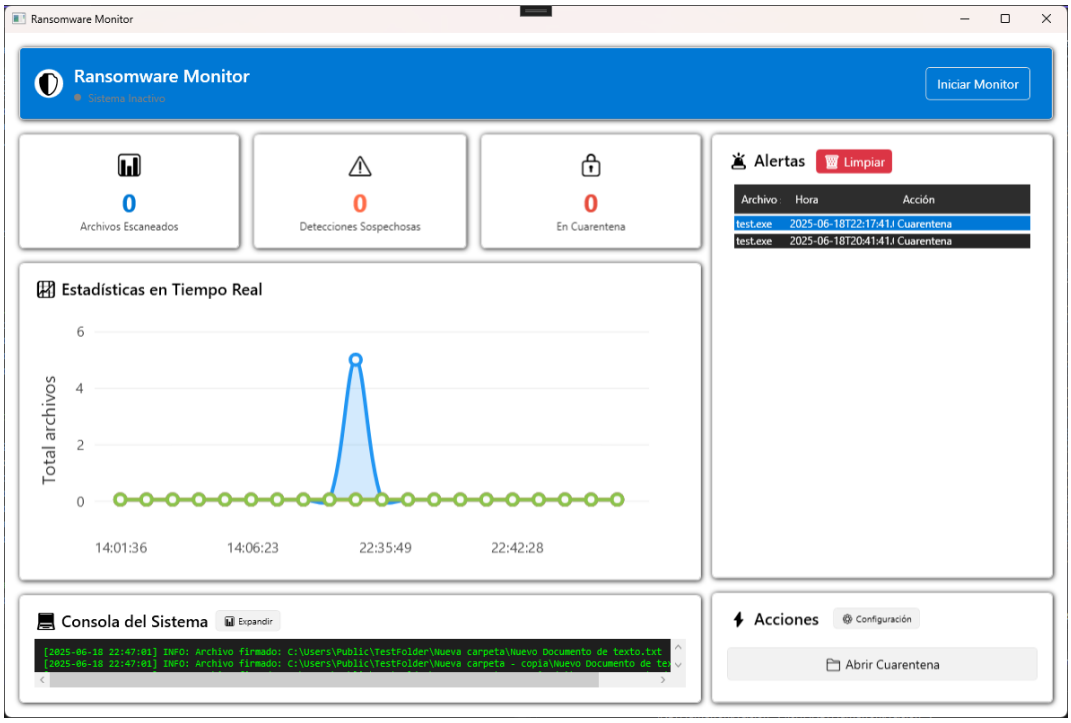


Figura 5.9: Mockup de la ventana principal. **Fuente: Elaboración propia.**



Capítulo 6

Implementación del Prototipo

6.1. Metolodogía de desarrollo de software

Como se mencionó anteriormente en la sección 1.8, se utilizó la metodología de desarrollo de software Kanban para el desarrollo del prototipo. Para la gestión del tablero Kanban se seleccionó la herramienta **FocalBoard**, una aplicación de código abierto que ofrece las funcionalidades necesarias de manera gratuita.

La implementación de Kanban se estructuró mediante un tablero con cinco columnas claramente definidas que se muestra en la Figura 6.1

1. **Backlog**: Contiene la totalidad de las actividades por realizar.
2. **Por hacer**: Contiene las tareas que están listas para realizarse, según el cronograma.
3. **En curso**: Contiene las actividades que se encuentran en desarrollo.
4. **Bloqueado**: Contiene las tareas que se encuentran bloqueadas por alguna razón.
5. **Finalizado**: Contiene las tareas finalizadas.

Figura 6.1: Tablero Kanban. **Fuente: Elaboración propia**

Historias de Usuario

Las historias de usuario son explicaciones generales e informales de una función de software escritas desde la perspectiva del usuario final. Su propósito es articular cómo proporcionará una función de software valor al cliente [5].

Para realizar HU (historias de usuario) se utiliza un lenguaje no técnico para tener contexto dentro del desarrollo, permitiendo una facil comprensión de lo que se esta construyendo [5].

Estructura de las Historias de Usuario

Para el desarrollo de este proyecto se utilizará la estructura estándar recomendada por Atlassian [5]:

"Como [perfil], [quiero] [para]."

Esta estructura se desglosa de la siguiente manera:

- **"Como [perfil]"**: Define específicamente para quién se desarrolla la funcionalidad. No se trata únicamente de identificar un rol o cargo, sino de comprender el perfil completo de la persona: cómo trabaja, piensa y se comporta [5].
- **"Quiero"**: Describe la intención real del usuario, enfocándose en qué está intentando lograr en lugar de las funciones técnicas que utilizará [5].
- **"Para"**: Explica cómo la necesidad inmediata del usuario se conecta con el beneficio general que se busca obtener [5].

A continuación, se presentan las historias de usuario definidas para el desarrollo del prototipo de sistema de detección temprana de software malintencionado por medio de comportamientos anómalos en Windows 10, a partir de los resultados esperados, definidos en 1.1.

Tabla 6.1: Historias de Usuario - Backlog. **Fuente: Elaboración propia.**

Identificador	Descripción	Tiempo definido
HU-001	Como investigador, quiero estudiar las metodologías de detección de ransomware existentes para seleccionar las más efectivas.	5-6
HU-002	Como investigador, quiero analizar el funcionamiento del ransomware para identificar patrones de comportamiento detectables.	6-7
HU-003	Como desarrollador, quiero analizar herramientas y tecnologías disponibles para definir la arquitectura del sistema.	8

El informe completo de las historias de usuario que se realizaron para este proyecto se encuentran en los anexos (9).

6.2. Configuración de los Entornos de Desarrollo y Producción

6.2.1. Entorno de Desarrollo

El desarrollo del prototipo del sistema de detección se implementó con técnicas, herramientas y tecnologías seleccionadas para garantizar una compatibilidad nativa con Windows 10, además de poder acceder a las APIs del sistema operativo.

Plataforma de Desarrollo

La configuración del entorno de desarrollo se realizó teniendo en cuenta que el prototipo se ejecutará en Windows 10, por lo que se requiere la mejor compatibilidad posible. En la Tabla 6.2, se presentan los componentes principales del entorno de desarrollo:

Tabla 6.2: Componentes principales del entorno de desarrollo. **Fuente: Elaboración propia.**

Componente	**Versión/Especificación**	**Justificación de selección**
Framework	.NET 8.0	Es el marco de trabajo con mejor afinidad para realizar aplicaciones en Microsoft Windows [24]
Lenguaje de programación	C#	Es el lenguaje de programación que utiliza .NET para el desarrollo de aplicaciones de escritorio.

Dependencias y Componentes del Sistema

El sistema requiere de diversas librerías específicas para realizar el análisis a tiempo real, la detección de anomalías y respuestas ante software malintencionado. En la Tabla 6.3 se muestran las librerías y dependencias que se utilizaron en el desarrollo del prototipo.

Tabla 6.3: Librerías y dependencias del sistema. **Fuente: Elaboración propia.**

Librería	**Utilidad**	**Funcionalidades implementadas**
System Security.Cryptography	Operaciones criptográficas	Generación de hashes SHA-256 para integridad de archivos, implementación de cifrado AES-128 para cuarentena
System IO.FileSystem	Monitoreo del sistema de archivos	Detección de cambios en tiempo real, seguimiento de operaciones de archivo (creación, modificación, eliminación, renombrado)
System.Text.Json	Serialización de datos	Persistencia de los modelos de datos utilizadps por el sistema en formato JSON
LiveChartsCore SkiaSharpView.WPF	Visualización de datos	Generación de gráficos en tiempo real para estadísticas de monitoreo

Herramientas Específicas de Análisis

Para el manejo de los procesos del sistema operativo, se deben utilizar herramientas externas auxiliares para obtener información detallada sobre estos procesos y poder manipularlos fácilmente. En la Tabla 6.4 se muestran las herramientas utilizadas y su función en el sistema.

Tabla 6.4: Herramientas auxiliares del sistema. **Fuente: Elaboración propia.**

Herramienta	**Función**	**Implementación en el sistema**
Handle.exe (Sysinternals)	Identificación de procesos	Determina qué procesos mantienen archivos abiertos durante modificaciones sospechosas

6.2.2. Arquitectura Modular del Proyecto

El sistema fue diseñado siguiendo principios de arquitectura limpia, separando responsabilidades en módulos independientes que facilitan el mantenimiento y escalabilidad futura del prototipo. Esto, justificado anteriormente en la sección de arquitectura del sistema, definida en la Figura 5.2.

Módulo Principal

En el módulo principal se encuentran los componentes que interactuan directamente con el sistema operativo, funciones esenciales para el funcionamiento del sistema y la lógica principal del modelo propuesto. En la Tabla 6.5 se muestran los componentes principales del módulo.

Tabla 6.5: Componentes del módulo RansomwareMonitor.Core. **Fuente: Elaboración propia.**

Componente	**Función principal**
FileMonitor	Implementa el modelo propuesto para detectar anomalías en el sistema de archivos a tiempo real
AnomalyDetector	Analiza patrones de comportamiento y aplica las técnicas de detección seleccionadas
PermissionManager	Gestiona permisos de administrador para que el prototipo pueda acceder sin restricciones al sistema operativo

Módulo de Servicios

En el módulo de servicios se encuentran los componentes que interactuan entre el módulo principal y la interfaz de usuario. En la Tabla 6.6 se muestran los servicios del prototipo.

Tabla 6.6: Servicios del sistema. **Fuente: Elaboración propia.**

Servicio	**Funcionalidad**
FileSignatureService	Implementa el algoritmo SHA-256, y la base de datos de firmas conocidas
LoggerService	Realiza el registro de los eventos del sistema a tiempo real
QuarantineService	Toma acciones sobre los archivos sospechosos, aislandolos y cifrandolos para evitar su futura ejecución
NotificationService	Notifica al usuario en tiempo real sobre las acciones o eventos más relevantes

Interfaz de Usuario

En el módulo de interfaz de usuario (UI) se encuentran los componentes que permiten que los usuarios puedan interactuar con el sistema y permite ver de manera gráfica lo que sucede a tiempo real. En la Tabla **??** se muestran los componentes de la interfaz de usuario.

Tabla 6.7: Componentes de la interfaz gráfica. **Fuente: Elaboración propia.**

Componente UI	**Propósito**	**Características de usabilidad**
MainWindow	Panel principal de control	Pestañas organizadas para logs, alertas y estadísticas
NotificationWindow	Alertas críticas	Ventanas emergentes para notificar eventos importantes

6.2.3. Configuración del Sistema de Monitoreo

Configuración de Parámetros de Funcionamiento

Para asegurar que el modelo propuesto funcione de la manera que se espera, se deben tener en cuenta ciertos parámetros de tolerancia para determinar si un proceso es malicioso o no. A continuación, se muestra la Tabla 6.8 con los parámetros de configuración del sistema:

Tabla 6.8: Configuración principal del sistema. **Fuente: Elaboración propia.**

Parámetro	**Valor predeterminado**	**Funcionalidad**
ChangeThreshold	5 cambios	Umbral de detección de actividad masiva sospechosa
IntervalMilliseconds	1000 ms	Frecuencia (en milisegundos) de evaluación de patrones
QuarantineFolder	C:\Quarantine	Ubicación de los archivos marcados como sospechosos que fueron aislados
ProcessTimeoutMs	5000 ms	Tiempo límite para que el modelo identifique un proceso sospechoso

6.2.4. Flujo Operativo del Sistema

Inicialización del prototipo

El prototipo tiene una secuencia de inicialización específica para su funcionamiento. Esta secuencia se puede observar en la Tabla 6.9:

Tabla 6.9: Fases de inicialización. **Fuente: Elaboración propia.**

Fase	**Actividades realizadas**	**Validaciones aplicadas**
Verificación de permisos	Comprobación de privilegios administrativos	Verifica que el prototipo tiene acceso a directorios del sistema, capacidad de terminación de procesos
Carga de configuración	Lectura de fileSignatures.json y parámetros	Valida la integridad de los archivos generados por el prototipo
Escaneo inicial	Creación de firmas SHA-256	Generación de firmas SHA-256 de archivos existentes en el directorio objetivo
Configuración de monitoreo	Inicialización de FileManager	El prototipo comienza a monitorear el directorio objetivo

Proceso de Detección

Una vez el prototipo ha sido inicializado, comienza a monitorear el directorio objetivo. El proceso de detección y análisis se puede observar en la Tabla 6.10:

Tabla 6.10: Algoritmo de detección en tiempo real. **Fuente: Elaboración propia.**

Etapa de detección	**Proceso ejecutado**	**Criterios de evaluación**	**Acciones resultantes**
Captura de eventos	Detección de cambios en archivos	Filtros configurados, tipos de evento	Registro en cola de procesamiento
Análisis de integridad	Generación y comparación de SHA-256	Diferencias con firmas almacenadas	Marcado como archivo modificado
Identificación de proceso	Ejecución de Handle.exe	Correlación archivo-proceso	Identificación del proceso responsable
Evaluación de patrón	Conteo temporal de cambios	Umbral configurado y ventana temporal	Clasificación de actividad sospechosa
Análisis de comportamiento	Evaluación de patrones de ransomware	Velocidad de cambios, tipos de archivo	Determinación de nivel de amenaza

Sistema de Respuesta

El sistema de respuesta del prototipo se implementó para actuar de manera automática ante la detección de una amenaza. Cada nivel de amenaza tiene un conjunto de acciones específicas, como se muestra en la Tabla 6.11.

Tabla 6.11: Mecanismos de respuesta ante amenazas. **Fuente: Elaboración propia.**

Nivel de amenaza	**Acciones**	**Objetivo**
Sospechoso (Nivel 1)	Logging detallado, notificación no crítica	Realiza un registro detallado de la actividad sospechosa
Sobrepaso del umbral (Nivel 2)	Bloqueo completo, notificación critica	Realiza un registro de la actividad sospechosa, alerta al usuario y bloquea el proceso

6.3. Repositorio de código fuente

Para el manejo de versiones del código fuente se utilizó GitHub, una plataforma de desarrollo colaborativo que permite el control de versiones y la colaboración en proyectos de software. A continuación, se describen los aspectos de la configuración del repositorio:

Estructura del repositorio

El repositorio se organizó siguiendo las mejores prácticas de desarrollo .NET:

- Directorio raíz con archivo de solución (.sln).
- Carpetas separadas para cada proyecto de la solución.
- Archivo .gitignore configurado para proyectos .NET.
- README.md con instrucciones de instalación y uso.

Gestión de ramas

Se implementó una estrategia de branching simplificada:

- **main**: Rama principal con código estable.
- **feature/***: Ramas específicas para implementación de características individuales.

El enlace completo al repositorio del proyecto se encuentra disponible en los anexos (9), donde se puede acceder al código fuente completo y el historial de versiones del prototipo.

Capítulo 7
Pruebas Aplicadas al Prototipo

7.1. Pruebas del Prototipo

7.1.1. Entorno de Pruebas

Para evaluar la efectividad del sistema de detección de ransomware desarrollado, se configuró un entorno de pruebas controlado que simula las condiciones reales de un ataque. La infraestructura de pruebas consistió en un sistema anfitrión ejecutando Ubuntu 20.04 LTS sobre hardware con las siguientes especificaciones:

- **Procesador**: AMD FX-4300 (4 núcleos @ 3.8 GHz).
- **Memoria RAM**: 8 GB DDR3 @ 1866 MHz.
- **Almacenamiento**: Disco duro mecánico de 320 GB.
- **Sistema operativo anfitrión**: Ubuntu 20.04 LTS.

Dentro de este sistema, se configuró una máquina virtual que actuó como víctima potencial del ransomware, con las siguientes características:

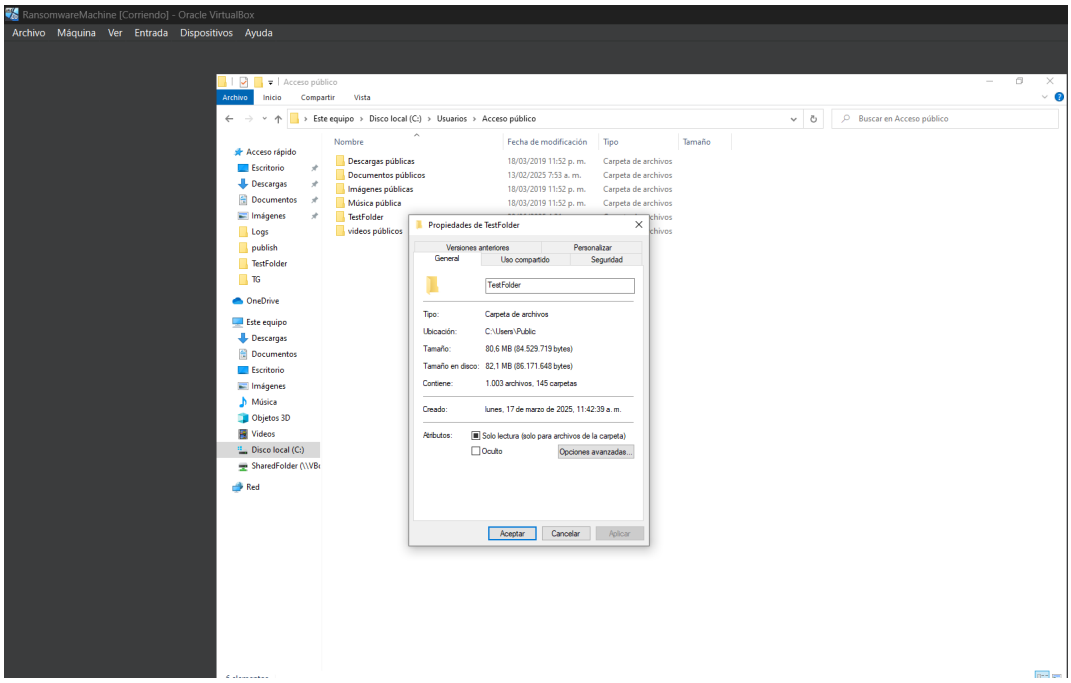
- **Sistema operativo**: Windows 10 versión 22H2.
- **Recursos asignados**: 6 GB de RAM, 64 GB de espacio en disco y 2 núcleos de CPU.
- **Software de virtualización**: VirtualBox 7.1.

Herramientas y Conjunto de Datos

Para simular el comportamiento de un ransomware real sin comprometer la seguridad del entorno, se utilizó *Ransim* (9), un simulador de ransomware diseñado específicamente para propósitos de investigación y pruebas de seguridad.

El conjunto de datos de prueba consistió en 1,000 archivos de diversos tipos y extensiones, como se observa en la Figura 7.1. Los archivos se obtuvieron del repositorio público de archivos de muestra disponible en el anexo (9). La elección de este conjunto de datos se justifica en la necesidad de replicar un entorno realista, donde los archivos críticos y sensibles podrían ser afectados por un ataque. Esta colección incluye documentos de oficina, imágenes, archivos de texto, PDFs y otros formatos comúnmente atacados por ransomware, proporcionando así un escenario realista para las pruebas.

Figura 7.1: Distribución de archivos de prueba por tipo y extensión. **Fuente: Elaboración propia.**



7.1.2. Evaluación del Rendimiento

Para medir la efectividad del prototipo, se establecieron tres métricas fundamentales:

1. **Tiempo de inicialización**: Período requerido para que el sistema complete su arranque y genere las firmas SHA-256 de todos los archivos monitorizados.

2. **Tiempo de respuesta**: Intervalo transcurrido desde el inicio del ataque hasta la detección y mitigación del comportamiento malicioso.

3. **Tasa de protección**: Proporción de archivos que permanecieron intactos gracias a la intervención del sistema.

Resultados

Se realizaron cinco pruebas para obtener resultados significativos. Los datos obtenidos se presentan en la Tabla 7.1.

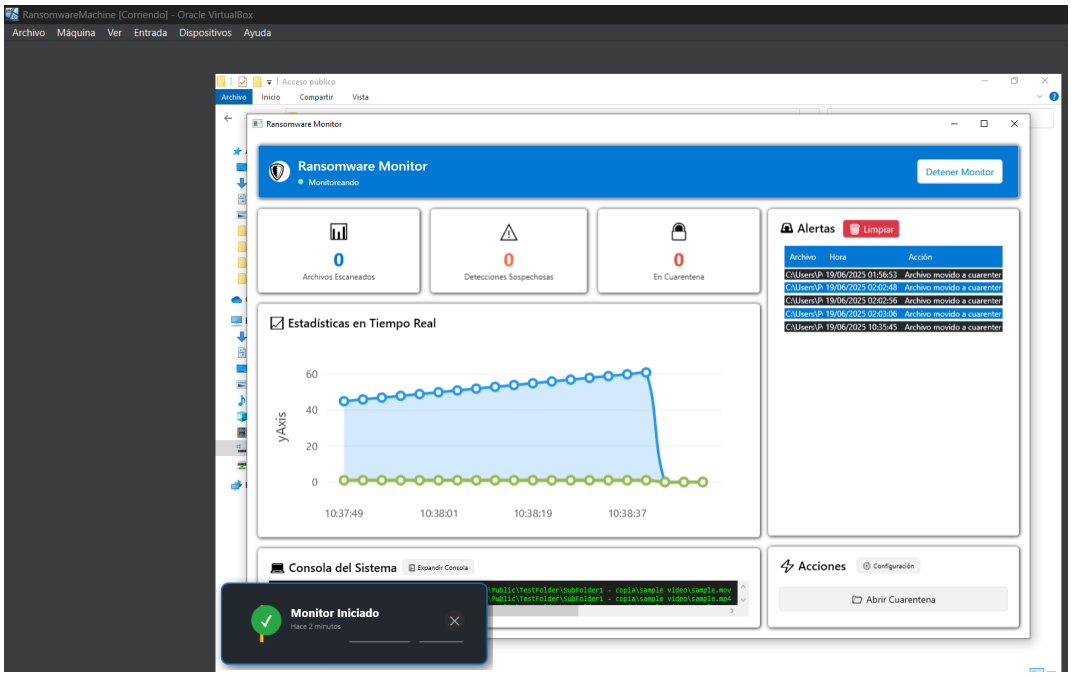
Tabla 7.1: Resultados de las pruebas de detección de ransomware

Prueba No.	Tiempo de inicialización (s)	Tiempo de respuesta (s)	Archivos afectados	Tasa de protección (%)
1	230	5	5/1000	99.5 %
2	207	7	4/1000	99.6 %
3	215	5	5/1000	99.5 %
4	209	6	4/1000	99.6 %
5	215	6	5/1000	99.5 %
Promedio	**215.2**	**5.8**	**4.6/1000**	**99.54 %**

Muestra visual de las pruebas

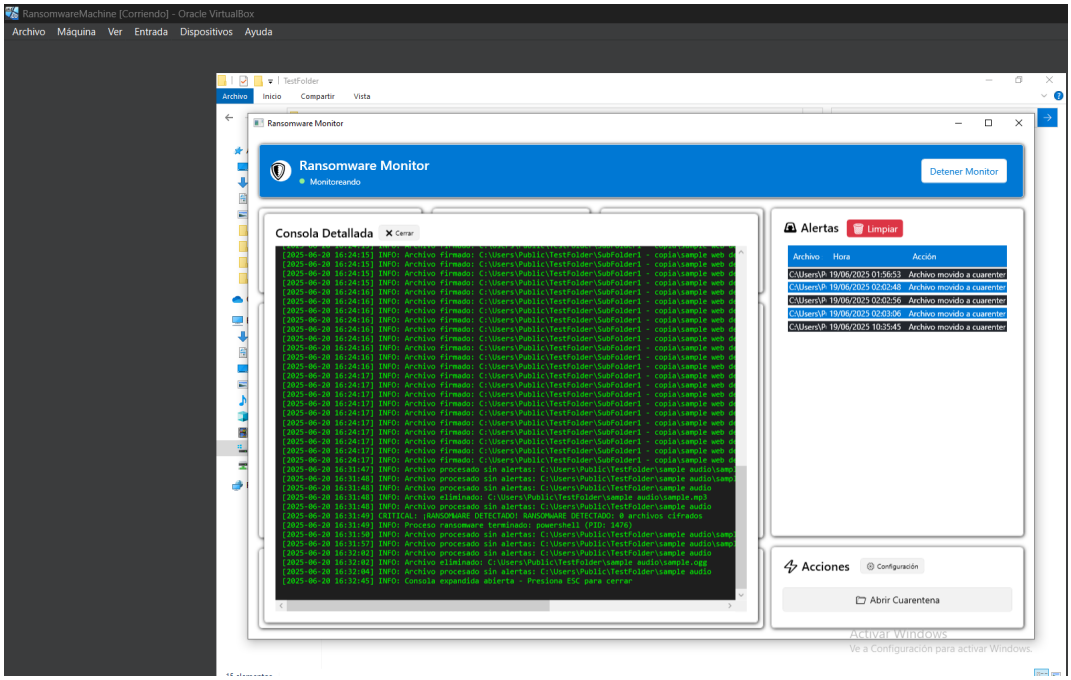
En la Figura 7.2 se muestra la pantalla de inicio del prototipo:

Figura 7.2: Pantalla de inicio del sistema de monitoreo. **Fuente: Elaboración propia.**



En la Figura 7.3 se muestra cómo la interfaz avisa cuando se detecta ransomware:

Figura 7.3: Pantalla de detección de ransomware. **Fuente: Elaboración propia.**



Análisis de Resultados

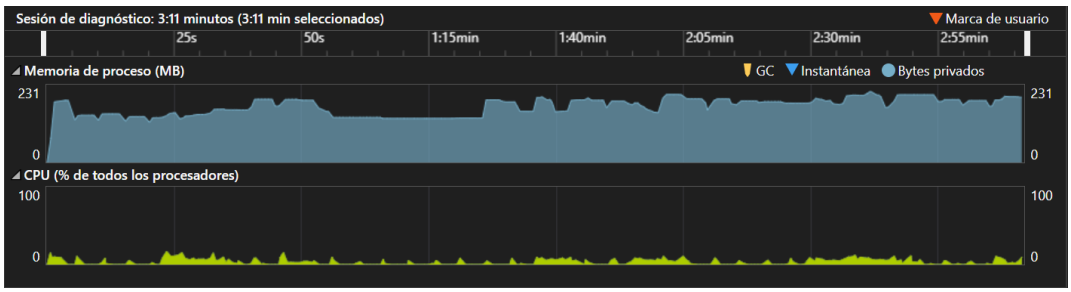
Los resultados obtenidos demuestran la alta efectividad del sistema propuesto:

- **Tiempo de inicialización promedio**: 3 minutos y 35 segundos (215.2 segundos). Aunque este tiempo puede parecer elevado, es importante considerar que la generación de firmas SHA-256 para 1,000 archivos es una operación intensiva que solo se realiza una vez al inicio del sistema.
- **Tiempo de respuesta promedio**: 5.8 segundos. Este resultado es particularmente significativo, ya que el sistema logra detectar y detener el ransomware en menos de 6 segundos desde el inicio del ataque, minimizando drásticamente el daño potencial.
- **Tasa de protección**: 99.54 %. El sistema logró proteger exitosamente más del 99 % de los archivos en todas las pruebas, con un promedio de solo 4.6 archivos afectados de un total de 1,000. Esta tasa de protección supera considerablemente a las soluciones tradicionales basadas en firmas.

Estos resultados demuestran la efectividad del enfoque basado en análisis de comportamientos anómalos para la detección temprana de ransomware, demostrando que es posible lograr tasas de protección superiores al 99 % con tiempos de respuesta inferiores a 6 segundos. Adicionalmente, el uso de los recursos del sistema fueron mínimos, con un uso máximo de 15 % del CPU y 231MB de

memoria RAM, como se puede observar en la Figura 7.4. Esto demuestra que el sistema es eficiente y no afecta significativamente el rendimiento del Sistema Operativo.

Figura 7.4: Uso de recursos del sistema. **Fuente: Elaboración propia.**



7.2. Encuestas y Justificación

7.2.1. Encuesta de usabilidad

Esta encuesta se formuló con el objetivo de evaluar la usabilidad y experiencia de usuario de la interfaz de seguridad desarrollada. La evaluación se centró en aspectos críticos como la claridad de la finalidad, apropiación del diseño visual, organización de elementos, efectividad de las notificaciones, facilidad de navegación y satisfacción general del usuario. Se aplicó a 32 participantes utilizando una metodología mixta que combinó escalas cualitativas y cuantitativas, así como preguntas abiertas para obtener retroalimentación específica sobre mejoras y funcionalidades adicionales.

7.2.2. Encuesta de compatibilidad

Esta encuesta se formuló con el objetivo de evaluar la compatibilidad técnica del prototipo desarrollado con el sistema operativo Windows 10, así como su capacidad para integrarse de manera nativa y eficiente en el entorno del usuario final. La evaluación se centró en aspectos críticos como la apariencia nativa de la aplicación, la capacidad de monitoreo en tiempo real sin comprometer el rendimiento del sistema, y la identificación de mejoras técnicas para optimizar la compatibilidad general.

7.2.3. Justificación de las encuestas

Las encuestas fueron diseñadas para obtener una evaluación de la interfaz de seguridad desarrollada, enfocándose en aspectos clave que impactan la usabilidad y la experiencia del usuario. Los soportes de estas encuestas se encuentran en el apartado de anexos.

7.3. Resultados y Análisis de las Encuestas

A continuación, se presentan los resultados obtenidos de las encuestas aplicadas, organizados por cada pregunta evaluada:

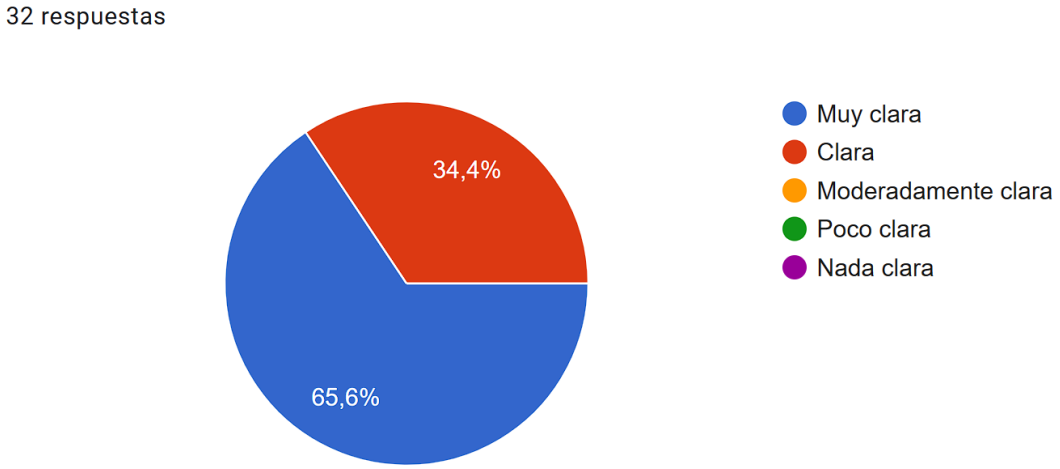
7.3.1. Encuesta de usabilidad

Claridad de la finalidad de la interfaz

Pregunta: Al ver la interfaz por primera vez, ¿qué tan clara le resulta su finalidad?
Escala de calificación: Cualitativa (Muy clara, Clara, Moderadamente clara, Poco clara, Nada clara)
Resultados:

- Muy clara: 21 respuestas (65,6 %)
- Clara: 11 respuestas (34,4 %)
- Moderadamente clara: 0 respuestas (0 %)
- Poco clara: 0 respuestas (0 %)
- Nada clara: 0 respuestas (0 %)

Análisis: El 100 % de los usuarios considera que la finalidad de la interfaz es clara o muy clara, con una mayoría significativa (65,6 %) que la encuentra muy clara. Este resultado indica una excelente comunicación visual del propósito de la aplicación.

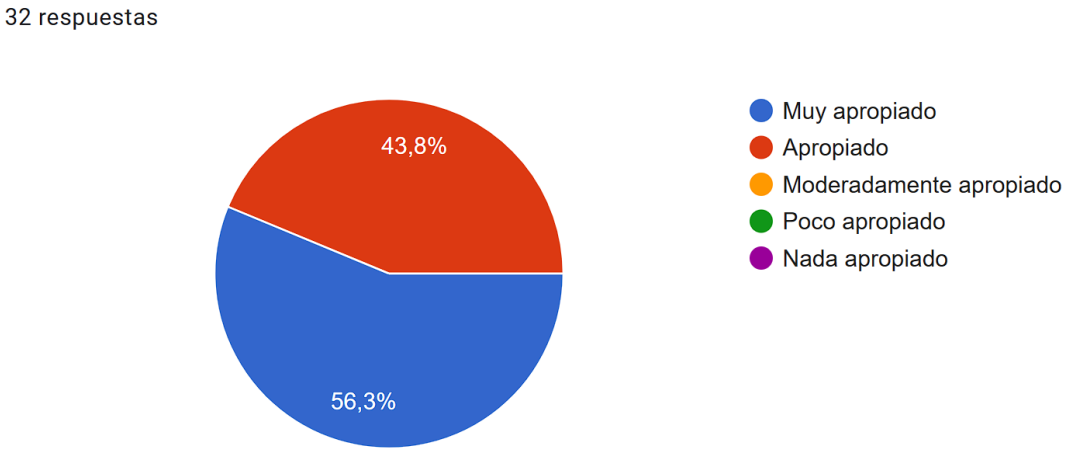


Apropiación del diseño visual

Pregunta: ¿Qué tan apropiado considera el diseño visual de la interfaz?
Escala de calificación: Cualitativa (Muy apropiado, Apropiado, Moderadamente apropiado, Poco apropiado, Nada apropiado)
Resultados:

- Muy apropiado: 18 respuestas (56,3 %)
- Apropiado: 14 respuestas (43,8 %)
- Moderadamente apropiado: 0 respuestas (0 %)
- Poco apropiado: 0 respuestas (0 %)
- Nada apropiado: 0 respuestas (0 %)

Análisis: Todos los usuarios (100 %) consideran el diseño visual apropiado o muy apropiado, mostrando una aceptación completa del diseño implementado.



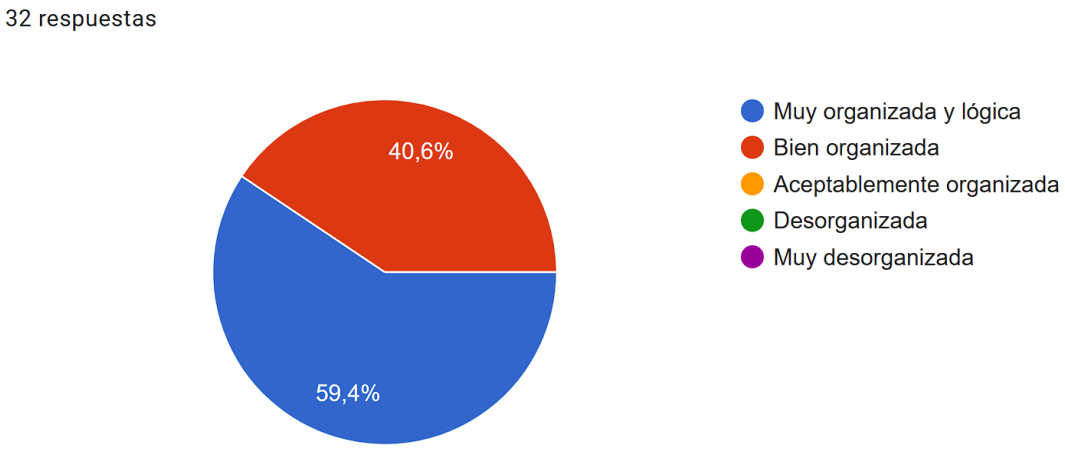
Organización de elementos en pantalla

Pregunta: La organización de los elementos en pantalla le parece
Escala de calificación: Cualitativa (Muy organizada y lógica, Bien organizada, Aceptablemente organizada, Poco organizada, Desorganizada)
Resultados:

- Muy organizada y lógica: 19 respuestas (59,4 %)
- Bien organizada: 13 respuestas (40,6 %)
- Aceptablemente organizada: 0 respuestas (0 %)
- Poco organizada: 0 respuestas (0 %)

- Desorganizada: 0 respuestas (0 %)

Análisis: La totalidad de usuarios (100 %) percibe la organización como buena o muy buena, con predominio de "muy organizada y lógica", lo que confirma la efectividad de la arquitectura de información implementada.

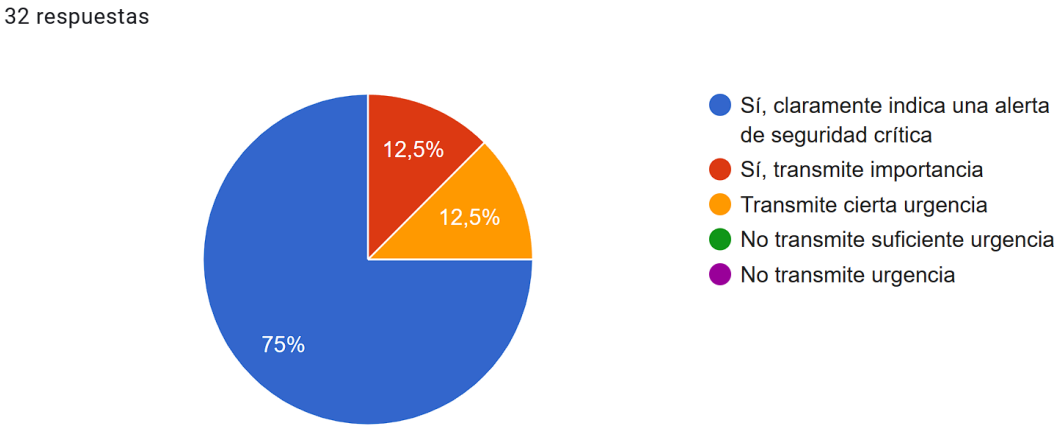


Efectividad de las notificaciones

Pregunta: ¿El diseño de las notificaciones le transmite urgencia apropiada?
Escala de calificación: Cualitativa (Sí claramente indica alerta crítica, Sí transmite importancia, Transmite cierta urgencia, No transmite suficiente urgencia, No transmite urgencia)
Resultados:

- Sí, claramente indica una alerta de seguridad crítica: 24 respuestas (75 %)
- Sí, transmite importancia: 4 respuestas (12,5 %)
- Transmite cierta urgencia: 4 respuestas (12,5 %)
- No transmite suficiente urgencia: 0 respuestas (0 %)
- No transmite urgencia: 0 respuestas (0 %)

Análisis: El 100 % de usuarios percibe algún nivel de urgencia, siendo la mayoría (74 %) quienes identifican claramente la criticidad de las alertas. Este resultado es crucial para una aplicación de seguridad.



Facilidad de navegación diaria

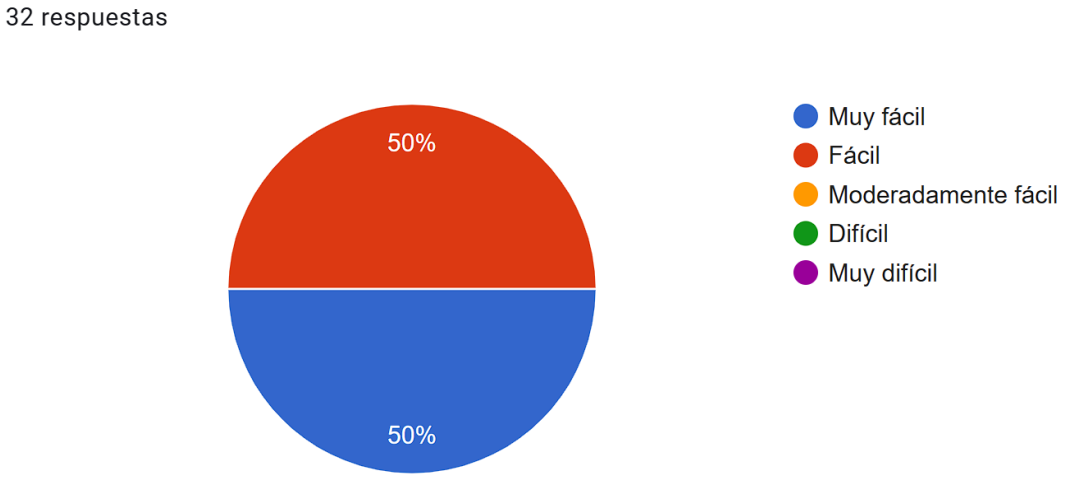
Pregunta: Si tuviera que usar esta herramienta diariamente, ¿qué tan fácil sería navegar por ella?

Escala de calificación: Cualitativa (Muy fácil, Fácil, Moderadamente fácil, Difícil, Muy difícil)

Resultados:

- Muy fácil: 16 respuestas (50 %)
- Fácil: 16 respuestas (50 %)
- Moderadamente fácil: 0 respuestas (0 %)
- Difícil: 0 respuestas (0 %)
- Muy difícil: 0 respuestas (0 %)

Análisis: Todos los usuarios (100 %) consideran la navegación fácil o muy fácil, indicando una excelente usabilidad para uso frecuente del diseño para usuarios regulares.



Elementos a cambiar o mejorar

Pregunta: ¿Qué elemento de la interfaz cambiaría o mejoraría?
Escala de calificación: Pregunta abierta
Principales respuestas:

- Sin cambios/Todo perfecto: 8 respuestas (26 %)
- Mejorar consola del sistema: 3 respuestas (10 %)
- Implementar modo oscuro: 3 respuestas (10 %)
- Mejorar tamaño de elementos: 2 respuestas (6 %)
- Actualizar diseño visual (eliminar cajas): 1 respuesta (3 %)
- Mejorar alertas recientes: 1 respuesta (3 %)
- Otros comentarios específicos: 13 respuestas (42 %)

Análisis: Un cuarto de los usuarios no sugiere cambios, mientras que las mejoras más solicitadas se centran en aspectos estéticos como modo oscuro y refinamiento de la consola del sistema.

Funcionalidades adicionales sugeridas

Pregunta: ¿Qué funcionalidad adicional sugeriría para la interfaz?
Escala de calificación: Pregunta abierta
Principales respuestas:

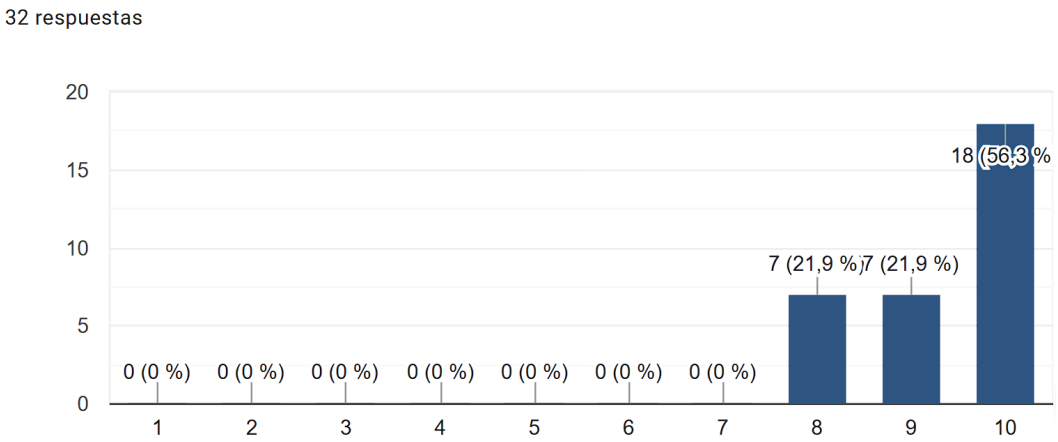
- Sin sugerencias/Está completa: 10 respuestas (32 %)
- Escaneos automáticos programados: 2 respuestas (6 %)
- Selección específica de archivos/carpetas: 2 respuestas (6 %)
- Modo oscuro: 2 respuestas (6 %)
- Información del sistema (CPU, red): 1 respuesta (3 %)
- Fecha del último escaneo: 1 respuesta (3 %)
- Restaurar archivos: 1 respuesta (3 %)
- Otras sugerencias: 12 respuestas (39 %)

Análisis: Un tercio considera la interfaz completa sin necesidad de funcionalidades adicionales. Las sugerencias se enfocan principalmente en automatización y personalización.

Calificación general de usabilidad

Pregunta: En una escala del 1 al 10, ¿cómo calificaría la usabilidad general de la interfaz?
Escala de calificación: Numérica (1-10, donde 10 es excelente)
Resultados:

- 10: 18 respuestas (56,3 %)
- 9: 7 respuestas (21,9 %)
- 8: 7 respuestas (21,9 %)
- 7 o menos: 0 respuestas (0 %)



Análisis estadístico:

- Promedio: 9.3/10

Conclusiones del análisis de usabilidad

Los resultados de la encuesta de usabilidad revelan una evaluación altamente positiva de la interfaz desarrollada. Las principales fortalezas identificadas incluyen:

1. **Claridad excepcional:** El 100 % de usuarios comprende la finalidad de la interfaz
2. **Diseño visual sólido:** Aceptación completa del diseño por parte de todos los evaluadores
3. **Organización efectiva:** Todos los usuarios perciben una buena organización de elementos
4. **Comunicación de urgencia:** El 87 % identifica claramente las alertas críticas
5. **Usabilidad superior:** Promedio de 9.3/10 en satisfacción general

Las áreas de mejora sugeridas por los usuarios se centran en:

1. **Personalización visual:** Implementación de modo oscuro como mejora prioritaria
2. **Refinamiento estético:** Actualización de la consola del sistema y elementos visuales
3. **Automatización:** Incorporación de escaneos programados automáticos
4. **Flexibilidad:** Capacidad de selección específica de elementos a escanear

Análisis General de Usabilidad

Los resultados de la encuesta sobre usabilidad ofrecen una mirada clara sobre qué tan bien se logró el objetivo de crear una interfaz intuitiva y efectiva. Un hallazgo muy positivo es que el 100 % de los usuarios comprendió de inmediato cuál era el propósito de la aplicación, lo cual confirma que tanto la estructura como los elementos visuales están bien diseñados para comunicar su función como software de seguridad. Esto es clave para generar confianza desde el primer momento.

En cuanto a la organización de la información en pantalla, el 59,4 % de los encuestados opinó que la interfaz está "muy organizada y lógica". Esto valida que las decisiones tomadas durante el diseño centrado en el usuario fueron acertadas. En una herramienta de seguridad, es esencial que los usuarios puedan encontrar rápidamente la información que necesitan para tomar decisiones importantes. La ausencia de comentarios negativos refuerza la efectividad de este enfoque.

Un punto especialmente relevante fue la percepción sobre las notificaciones de seguridad: el 75 % reconoció claramente las alertas como críticas. Esta capacidad para comunicar amenazas de forma clara y urgente es vital en un sistema de seguridad. Que ningún usuario haya considerado estas notificaciones como poco urgentes demuestra que el diseño visual cumple con los estándares de usabilidad para sistemas críticos.

La calificación promedio de 9.3/10 en usabilidad general, con un 56,3 % de usuarios otorgando la puntuación máxima, respalda de manera contundente la efectividad del diseño. Este resultado cobra aún más valor si se considera que en el ámbito de la seguridad, los usuarios suelen tener expectativas más altas. Por último, los comentarios abiertos reflejan que las sugerencias se enfocaron en aspectos estéticos —como la implementación de un modo oscuro— y no en fallas funcionales, lo cual indica que la base del diseño es sólida y está lista para futuras mejoras y ajustes finos.

7.3.2. Encuesta de compatibilidad

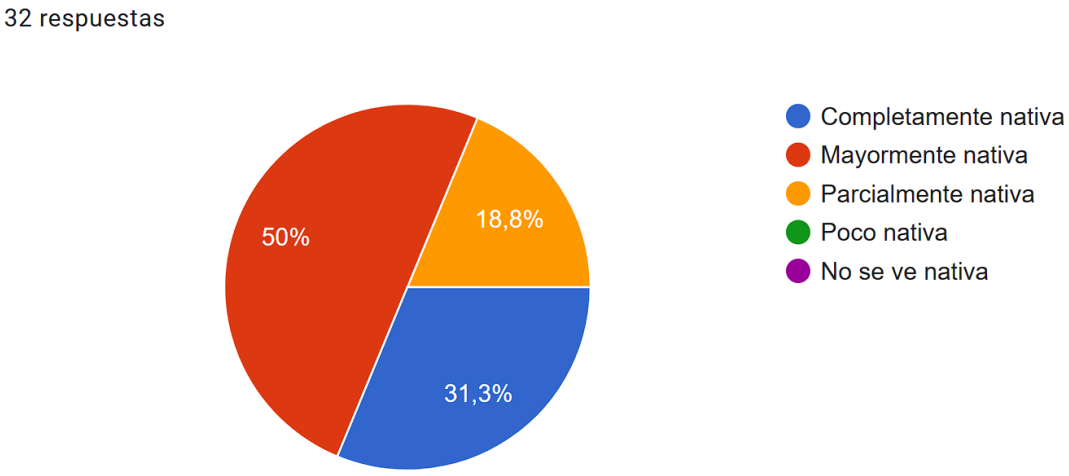
Apariencia nativa en Windows 10

Pregunta: ¿Considera que la aplicación se vería nativa en Windows 10?
Escala de calificación: Cualitativa (Completamente nativa, Mayormente nativa, Parcialmente nativa, Poco nativa, No se ve nativa)
Resultados:

- Completamente nativa: 10 respuestas (31,3 %)
- Mayormente nativa: 16 respuestas (50 %)

- Parcialmente nativa: 6 respuestas (18,8 %)
- Poco nativa: 0 respuestas (0 %)
- No se ve nativa: 0 respuestas (0 %)

Análisis: El 84 % de los usuarios considera que la aplicación tiene una apariencia nativa o mayormente nativa en Windows 10, con un 31,3 % que la percibe como completamente nativa. Solo el 18,8 % la considera parcialmente nativa, lo que indica una buena integración visual con el sistema operativo objetivo.



Capacidad de monitoreo en tiempo real

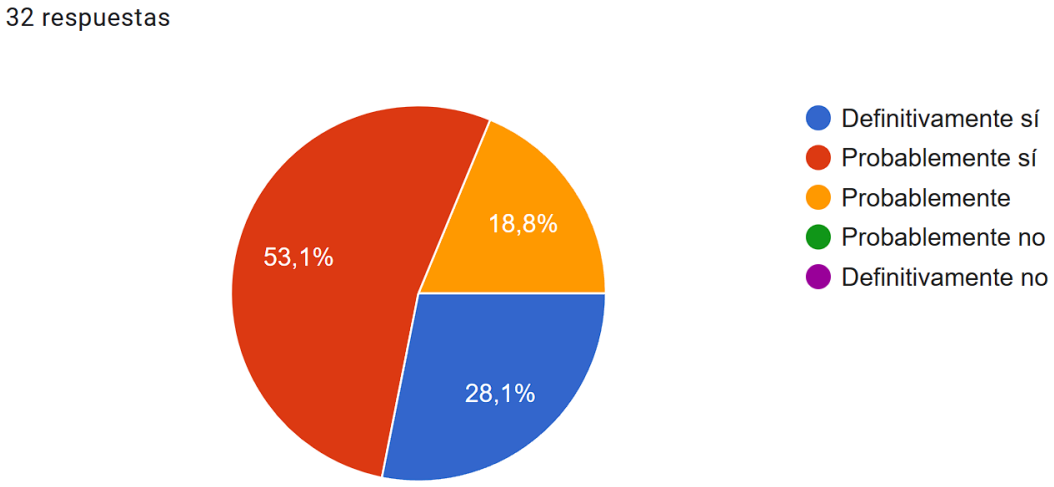
Pregunta: Basándose en la estructura mostrada, ¿cree que la aplicación podría manejar monitoreo en tiempo real sin afectar el rendimiento del sistema?

Escala de calificación: Cualitativa (Definitivamente sí, Probablemente sí, Probablemente, Probablemente no, Definitivamente no)

Resultados:

- Definitivamente sí: 9 respuestas (28.1 %)
- Probablemente sí: 17 respuestas (53.1 %)
- Probablemente: 6 respuestas (18.8 %)
- Probablemente no: 0 respuestas (0 %)
- Definitivamente no: 0 respuestas (0 %)

Análisis: El 84 % de los evaluadores considera que la aplicación probablemente o definitivamente puede manejar monitoreo en tiempo real sin afectar el rendimiento del sistema. Un 28,1 % tiene plena confianza en esta capacidad, mientras que un 53,1 % la considera probable, sugiriendo una arquitectura técnica sólida.



Mejoras técnicas sugeridas

Pregunta: ¿Qué mejoras técnicas sugeriría para optimizar la compatibilidad?
Escala de calificación: Pregunta abierta
Principales respuestas categorizadas:

- Sin sugerencias/N/A: 22 respuestas (71 %)
- Migración a Tauri: 2 respuestas (6 %)
- Consideración de ambientes informáticos: 1 respuesta (3 %)
- Herramientas multiplataforma: 1 respuesta (3 %)
- Arquitectura híbrida web-backend: 1 respuesta (3 %)
- Otras sugerencias técnicas: 4 respuestas (13 %)

Análisis: La mayoría (71 %) de los usuarios no identifica mejoras técnicas necesarias, sugiriendo satisfacción con la compatibilidad actual. Las sugerencias específicas se centran en tecnologías modernas como Tauri para mejorar el rendimiento y la multiplataforma, así como consideraciones de arquitectura híbrida.

Sugerencias técnicas destacadas:

- "Utilizar herramientas multiplataforma para garantizar un comportamiento consistente en distintos sistemas operativos"
- "Tener en cuenta los ambientes informáticos a los que va a ser instalado y los servicios que podrían generar alguna interferencia"
- "Tecnología web con lógica de negocio en backend para aplicaciones híbridas ligeras"

Calificación general de compatibilidad

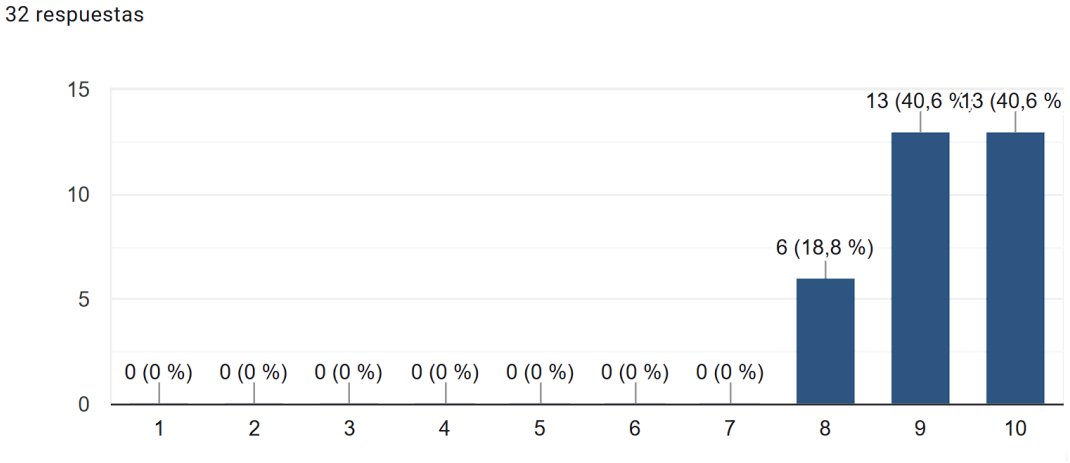
Pregunta: En una escala del 1 al 10, ¿cómo calificaría la compatibilidad general del prototipo?
Escala de calificación: Numérica (1-10, donde 10 es excelente)
Resultados:

- 10: 13 respuestas (40.6 %)
- 9: 13 respuestas (40.6 %)
- 8: 6 respuestas (18.8 %)
- 7 o menos: 0 respuestas (0 %)

Análisis estadístico:

- Promedio: 9.2/10

Análisis: La calificación promedio de 9.2/10 indica una excelente percepción de compatibilidad. El 81.2 % de los usuarios otorga calificaciones de 9 o 10, demostrando alta confianza en la compatibilidad técnica del prototipo.



Correlación entre variables de compatibilidad

Análisis cruzado:

- Los usuarios que perciben la aplicación como "Completamente nativa" tienden a calificar la compatibilidad general con 10
- Existe correlación positiva entre la confianza en el monitoreo en tiempo real y las calificaciones altas de compatibilidad
- Los usuarios que sugieren mejoras técnicas específicas mantienen calificaciones altas (8-10), indicando constructividad en lugar de insatisfacción

Conclusiones del análisis de compatibilidad

Los resultados de la encuesta de compatibilidad demuestran una evaluación altamente favorable del prototipo en términos de integración técnica con Windows 10. Las principales fortalezas identificadas incluyen:

1. **Integración visual sólida:** El 81,3 % percibe la aplicación como nativa o mayormente nativa en Windows 10
2. **Confianza en el rendimiento:** El 81,2 % considera viable el monitoreo en tiempo real sin impacto negativo
3. **Compatibilidad técnica superior:** Promedio de 9.2/10 en evaluación general
4. **Satisfacción técnica:** El 71 % no identifica mejoras necesarias en compatibilidad

Las recomendaciones técnicas identificadas se enfocan en:

1. **Optimización de tecnologías:** Consideración de frameworks modernos como Tauri para mejorar rendimiento
2. **Arquitectura híbrida:** Implementación de lógica de negocio en backend para aplicaciones más ligeras
3. **Compatibilidad multiplataforma:** Uso de herramientas que garanticen comportamiento consistente
4. **Consideraciones ambientales:** Evaluación de interferencias con otros servicios del sistema

Análisis General de Compatibilidad

La evaluación de compatibilidad permitió obtener información clave sobre qué tan viable es el prototipo para usarse en entornos reales. Un dato importante es que el 81,3 % de los participantes percibió la interfaz como nativa de Windows 10. Este aspecto es crucial, ya que muchos usuarios esperan que las aplicaciones se integren de forma fluida con el sistema operativo que utilizan a diario. Este resultado sugiere que las decisiones técnicas tomadas durante el desarrollo fueron acertadas para el entorno objetivo. Sin embargo, el 18,8 % que consideró la apariencia "parcialmente nativa" indica que aún hay oportunidades de mejora en la integración visual.

Otro aspecto fundamental fue la percepción del rendimiento del sistema durante el monitoreo en tiempo real. Aquí también se obtuvo un 81,2 % de respuestas positivas, lo que valida que el prototipo puede operar en segundo plano sin afectar la experiencia del usuario. Esta percepción refuerza la solidez de la arquitectura técnica adoptada y sugiere que el enfoque seguido es adecuado para un entorno de producción.

Aunque pocos usuarios hicieron sugerencias técnicas, estas aportaciones resultaron valiosas. Por ejemplo, algunos recomendaron migrar de Electron a Tauri, lo cual demuestra que el prototipo captó la atención de personas con conocimientos técnicos y que existe un camino claro para

optimizar aún más el rendimiento. También se mencionaron ideas sobre arquitecturas híbridas y soporte multiplataforma, lo que abre la puerta a futuras versiones que no se limiten exclusivamente a Windows 10.

Finalmente, la calificación promedio de 9.2/10 en compatibilidad general, con un 75 % de respuestas entre 9 y 10, muestra que el prototipo cumple con las expectativas técnicas de un software profesional de seguridad. Es especialmente destacable que no se recibieron calificaciones por debajo de 8, lo que indica que incluso los aspectos menos pulidos alcanzan un nivel aceptable de calidad técnica. Esto proporciona una base sólida para futuras mejoras y despliegues en escenarios reales.

7.4. Divulgación en Evento de Apropiación Social del Conocimiento

En el marco del **IV Encuentro Regional de Tesistas** organizado por la **Universidad del Valle**, se llevó a cabo la divulgación de los resultados del presente trabajo de grado, contribuyendo a la apropiación social del conocimiento en el área de ciberseguridad y detección de software malicioso.

7.4.1. Información del Evento

- **Título del evento:** IV Encuentro Regional de Tesistas
- **Fecha y lugar:** 2 de mayo de 2025, Universidad del Valle Seccional Buga

7.4.2. Descripción de la Participación

En el marco del IV Encuentro Regional de Tesistas se realizó la ponencia del trabajo de grado titulado “Prototipo de sistema de detección temprana de software malintencionado por medio de comportamientos anómalos en Windows 10”. Durante la presentación se compartieron con los asistentes los conceptos fundamentales sobre qué es y cómo funciona un *ransomware*, así como se expuso el modelo propuesto para mitigar los efectos de este tipo de amenazas en el sistema operativo Windows 10.

La ponencia permitió socializar los hallazgos de la investigación y generar un espacio de discusión académica sobre las problemáticas actuales en ciberseguridad, contribuyendo al intercambio de conocimientos entre la comunidad académica regional.

7.4.3. Evaluación y Reconocimiento

El trabajo presentado obtuvo una calificación de **98 puntos** por parte del evaluador, lo que refleja la calidad y relevancia de la investigación desarrollada.

7.4.4. Evidencias

Las evidencias de la participación en el evento, incluyendo fotografías y certificado de participación, se encuentran disponibles en los anexos (9). De igual forma, el trabajo presentado puede ser consultado en en los anexos (9)

Capítulo 8
Conclusiones y Trabajos Futuros

8.1. Conclusiones

La investigación realizada en este trabajo ha demostrado que es factible diseñar un prototipo capaz de detectar software malicioso mediante el análisis de patrones de comportamientos anómalos. Este logra identificar específicamente a los crypto-ransomware, los cuales se caracterizan por el cifrado masivo de archivos y patrones sospechosos de acceso de lectura y escritura. El prototipo alcanza una tasa de protección de aproximadamente 99 % frente a escenarios de cifrado masivo, con un tiempo de respuesta promedio de 5.8 segundos desde el inicio del ataque hasta su mitigación. Este desempeño demuestra que la detección basada en anomalías de entropía, frecuencia de operaciones y patrones de acceso es viable en tiempo real en Windows 10. Además, opera de manera completamente autónoma, monitoreando continuamente el sistema de archivos sin requerir intervención constante del usuario.

Desde el punto de vista técnico, esta investigación aporta varios elementos significativos al campo de la ciberseguridad, logrando integrar con éxico conceptos teóricos que combinan el enfoque de Kharraz "malice score" de REDemption [16] con el conteo de llamadas al sistema de Castañaga [6], lo que proporciona una detección robusta respecto a los métodos tradicionales y demuestra la viabilidad práctica en un entorno real de trabajo. Sin embargo, es importante reconocer ciertas limitaciones que se identificaron durante el desarrollo y las pruebas, como el proceso de generación de firmas SHA-256, el cual requiere un promedio de 3.5 minutos para completar su ejecución en un conjunto de datos de 1000 muestras, representando así, un área de oportunidad para optimización en futuras versiones. A pesar de que se implementaron mecanismos específicos para reducir los falsos positivos, existe la posibilidad de que ciertos procesos legítimos con comportamientos similares al ransomware puedan generar alertas incorrectas.

Esta aproximación establece bases sólidas para futuras investigaciones en el área, demostrando de manera práctica que la combinación de técnicas de análisis de comportamientos, técnicas de detección de anomalías y sistemas de respuesta automatizada puede proporcionar una defensa efectiva contra el ransomware y la pérdida de datos que este puede causar.

8.2. Trabajos Futuros

En este trabajo se desarrolló un prototipo de sistema para la detección temprana de ransomware mediante análisis de comportamientos anómalos, monitoreando de sistema de archivos a tiempo real; esto contruye una base sólida para construir sistemas de detección más especializados. En la revisión de la literatura, se pueden observar diferentes direcciones de investigación que podrían mejorar significativamente las capacidades del sistema de detección. A continuación se presentan algunas de las líneas de trabajos futuros con más afinidad para el prototipo desarrollado.

8.2.1. Mejoras en la Generación de Firmas SHA-256

El tiempo de inicialización del sistema, que actualmente promedia 3.5 minutos para generar las firmas de 1,000 archivos, representa una oportunidad significativa de optimización. Las mejoras futuras podrían incluir:

- **Procesamiento paralelo**: Implementar la generación de firmas utilizando múltiples hilos de ejecución, aprovechando los procesadores multinúcleo modernos para reducir el tiempo de inicialización hasta en un 70 %.
- **Optimización de E/S**: Implementar técnicas de lectura asíncrona y buffering optimizado para minimizar los tiempos de acceso a disco.
- **Caché persistente**: Mantener una base de datos local de firmas con verificación de integridad, permitiendo inicios rápidos del sistema mientras se actualizan solo los cambios en segundo plano.

8.2.2. Integración de Técnicas de ML (Machine Learning)

Detección Basada en API mediante Aprendizaje Automático

En la investigación realizada por Almousa et al. [2] se proponen modelos de detección basados en llamadas a la API nativa de Windows, lo que podría complementar el enfoque actual del prototipo, analizando patrones de llamadas al sistema y así extendiendo la capacidad de detección más allá del monitoreo de archivos.

Mejoras posibles al sistema:

- Entrenamiento de modelos de clasificación supervisada para identificar secuencias anómalas de llamadas a la API del sistema.
- Integrar la clasificación del modelo con el modelo de detección actual, para determinar de manera más eficiente comportamientos anómalos.

Análisis de Headers PE (Portable Executable) con Redes Neuronales

En las investigaciones realizadas por Manavi & Hamzeh [20, 21] realizan un modelado con redes neuronales convolucionales (CNN) y una red de memoria a corto-largo plazo (LSTM) para analizar headers de archivos ejecutables portables (PE). Este modelo demuestra un gran porcentaje de efectividad a la hora de detectar ransomware.

Mejoras posibles al sistema:

- Implementar el modelo al análisis estático de archivos actualmente implementado en el prototipo, para determinar si archivos portables ejecutables son ransomware.

Detección mediante Random Forest

La investigación realizada por Khammas [15] hace uso de un algoritmo llamado Random Forest, el cuál tiene la capacidad de clasificar y predecir el comportamiento de ransomware. El modelo reporta una precisión del 97.74 %, según las pruebas realizadas por el autor.

Mejora propuesta:

- Reemplazo del sistema de puntuación heurística por un ensemble de Random Forest.
- Entrenamiento con características como: entropía de archivos, frecuencia de I/O, patrones de acceso, metadatos temporales.
- Implementación de aprendizaje continuo para adaptación a nuevas variantes.

8.2.3. Enfoques de Deep Learning y Modelos Avanzados

Modelo CRED con Deep Learning Integrado

Alqahtani et al. [3] proponen el modelo Crypto-Ransomware Early Detection (CRED) que combina deep learning con modelos de espacio vectorial. Esta aproximación podría revolucionar el **AnomalyDetector** actual.

Arquitectura propuesta:

- Implementación de un **DeepLearningEngine** basado en redes neuronales profundas.
- Representación vectorial de comportamientos de archivo usando Word2Vec o Doc2Vec.
- Detección en tiempo real mediante análisis de ventanas deslizantes de actividad.
- Umbral dinámico basado en confianza del modelo.

Análisis de Memory Dumps con ML

Hossain & Islam [11] desarrollan técnicas avanzadas para detección de malware ofuscado en volcados de memoria. Esta técnica podría añadir una capa adicional de detección.

Trabajo futuro:

- Desarrollo de un **MemoryAnalysisModule** que capture y analice volcados de memoria de procesos sospechosos.
- Implementación de algoritmos de ML para detección de ransomware ofuscado en memoria.
- Correlación entre análisis de memoria y comportamiento de archivos.

8.2.4. Modelos Probabilísticos y Detección Temprana

Detección Pre-Cifrado Adaptativa

Urooj et al. [35] proponen un modelo adaptativo de detección pre-cifrado que podría integrarse perfectamente con la arquitectura actual del prototipo.

Implementación sugerida:

- Desarrollo de un **PreEncryptionDetector** que identifique comportamientos previos al cifrado.
- Análisis probabilístico de patrones de acceso a archivos.
- Modelo adaptativo que ajuste umbrales según el comportamiento histórico del usuario.
- Integración con el **QuarantineService** para acción preventiva.

8.2.5. Técnicas de Análisis Dinámico Avanzado

Framework de Análisis Dinámico Automatizado

Sgandurra et al. [31] describen las ventajas y limitaciones del análisis dinámico automatizado de ransomware, sugiriendo un enfoque híbrido que podría mejorar significativamente el prototipo.

Extensión propuesta:

- Desarrollo de un **DynamicAnalysisEngine** que complemente el monitoreo estático actual.
- Implementación de sandboxing ligero para análisis de comportamiento.
- Correlación entre análisis estático (FileManager) y dinámico.
- Generación de firmas comportamentales para detección rápida.

Monitoreo de Instrucciones de Cifrado

Enomoto et al. [8] proponen la detección temprana mediante monitoreo de instrucciones de cifrado optimizadas para CPU, una técnica altamente relevante para el prototipo desarrollado.

Trabajo futuro:

- Integración de un **CryptoInstructionMonitor** que detecte uso intensivo de instrucciones AES-NI.
- Análisis de patrones de uso de instrucciones criptográficas.
- Correlación con actividad de archivos para reducir falsos positivos.
- Implementación de contadores de rendimiento específicos.

8.2.6. Arquitecturas de Detección Sistemática

Framework de Alta Supervivencia

Ahmadian & Shahriari [1] presentan 2entFOX, un framework para detección de ransomware de alta supervivencia que podría inspirar mejoras arquitecturales.

Mejoras arquitecturales sugeridas:

- Implementación de redundancia en el **FileManager** para resistir ataques dirigidos.
- Desarrollo de múltiples canales de comunicación entre módulos.
- Implementación de auto-protección del sistema de detección.
- Backup distribuido de firmas y configuraciones críticas.

Detección Sistemática Multi-Técnica

Lee et al. [17] proponen técnicas de detección sistemática que combinan múltiples enfoques, lo que podría mejorar la robustez general del sistema.

Integración propuesta:

- Desarrollo de un **MultiTechniqueOrchestrator** que coordine diferentes métodos de detección.
- Implementación de votación ponderada entre técnicas estáticas, dinámicas y comportamentales.
- Sistema de confianza adaptativo basado en precisión histórica de cada técnica.

Framework Estático Mejorado

Medhat et al. [22] proponen un framework estático que podría complementar las capacidades actuales del **FileManager**.

Extensión propuesta:

- Implementación de análisis estático de archivos sospechosos.
- Extracción automática de características estáticas relevantes.
- Correlación entre características estáticas y comportamiento dinámico.
- Base de datos de firmas estáticas para detección rápida.

Bibliografía

[1] Ahmadian, M. M., & Shahriari, H. R. (2016). 2entFOX: A framework for high survivable ransomwares detection. *2016 13th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*, 79-84. https://doi.org/10.1109/ISCISC.2016.7736455

[2] Almousa, M., Basavaraju, S., & Anwar, M. (2021). API-Based Ransomware Detection Using Machine Learning-Based Threat Detection Models. *2021 18th International Conference on Privacy, Security and Trust (PST)*, 1-7. https://doi.org/10.1109/PST52912.2021.9647816

[3] Alqahtani, A., Gazzan, M., & Sheldon, F. T. (2020). A proposed Crypto-Ransomware Early Detection(CRED) Model using an Integrated Deep Learning and Vector Space Model Approach. *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, 0275-0279. https://doi.org/10.1109/CCWC47524.2020.9031182

[4] Alzahrani, A., Alshehri, A., Alharthi, R., Alshahrani, H., & Fu, H. (2017). An Overview of Ransomware in the Windows Platform. *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*, 612-617. https://doi.org/10.1109/CSCI.2017.106

[5] Atlassian. (2024). *Historias de usuario — Ejemplos y plantilla* [Guía sobre historias de usuario en desarrollo ágil y metodología Kanban]. Consultado el 2 de junio de 2024, desde https://www.atlassian.com/es/agile/project-management/user-stories

[6] Castañaga, I., Gibellini, F., Frias, P., Ruhl, L., Ciceri, L., Parisi, G., Zea, M., Bertola, F., & Olmedo, P. (2019). Estrategias de detección de ransomware de cifrado. *ASSE, Simposio Argentino de Ingeniería de Software (JAIIO)*.

[7] Diazgranados, H. (2021, abril). *22 % de usuarios de PC todavía utiliza el sistema operativo Windows 7*. Consultado el 1 de enero de 2024, desde https://latam.kaspersky.com/blog/22-de-usuarios-de-pc-todavia-utiliza-el-sistema-operativo-windows-7/21774/

[8] Enomoto, S., Kuzuno, H., Yamada, H., Shiraishi, Y., & Morii, M. (2024). Early mitigation of CPU-optimized ransomware using monitoring encryption instructions. *International Journal of Information Security*, *23*(5), 3393-3413. https://doi.org/10.1007/s10207-024-00892-2

[9] Hamdan, M. (2024, mayo). *Ransomware Detection with Advanced Elastic Search Queries*. Consultado el 1 de enero de 2024, desde https://motasem-notes.net/ransomware-detection-with-advanced-elastic-search-queries-tryhackme-advanced-elk/

[10] Hilabi, R., & Abu-Khadrah, A. (2024). Windows operating system malware detection using machine learning. *Bulletin of Electrical Engineering and Informatics*, *13*(5), 3401-3410. https://doi.org/10.11591/eei.v13i5.8018

[11] Hossain, M. A., & Islam, M. S. (2024). Enhanced detection of obfuscated malware in memory dumps: a machine learning approach for advanced cybersecurity. *Cybersecurity*, *7*(1), 16. https://doi.org/10.1186/s42400-024-00205-z

[12] IBM. (2024, abril). *IBM Security Guardium Data Protection 12.x*. Consultado el 1 de enero de 2024, desde https://www.ibm.com/docs/es/gdp/12.x?topic=policies-security-anomalies

[13] IBM. (2025). *Cost of a Data Breach Report 2025*. Consultado el 31 de julio de 2025, desde https://www.ibm.com/reports/data-breach

[**14**] KARA, I., & AYDOS, M. (2020). Cyber Fraud: Detection and Analysis of the Crypto-Ransomware. *2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 0764-0769. https://doi.org/10.1109/UEMCON51285.2020.9298128

[**15**] Khammas, B. M. (2020). Ransomware Detection using Random Forest Technique. *ICT Express*, *6*(4), 325-331. https://doi.org/10.1016/j.icte.2020.11.001

[**16**] Kharraz, A., & Kirda, E. (2017). Redemption: Real-Time Protection Against Ransomware at End-Hosts. En *Proceedings of the 20th International Symposium on Research in Attacks, Intrusions and Defenses* (pp. 98-119). https://doi.org/10.1007/978-3-319-66332-6_5

[**17**] Lee, S.-J., Shim, H.-Y., Lee, Y.-R., Park, T.-R., Park, S.-H., & Lee, I.-G. (2022). Study on Systematic Ransomware Detection Techniques. *2022 24th International Conference on Advanced Communication Technology (ICACT)*, 297-301. https://doi.org/10.23919/ICACT53585.2022.9728909

[**18**] Lugo, D. Á. (2024, mayo). *Las consecuencias de la falta de capacitación en ciberseguridad*. Forbes República Dominicana. Consultado el 1 de enero de 2024, desde https://forbes.do/red-forbes/2024-05-07/las-consecuencias-de-la-falta-de-capacitacion-en-ciberseguridad

[**19**] Malik, N. A., Delshadi, A. M., Ibrar, M., Hamid, K., Aamir, M., Ahmed, F., & Ahmad, G. (2024). Behavior and Characteristics of Ransomware - A Survey. *2024 2nd International Conference on Cyber Resilience (ICCR)*, 01-05. https://doi.org/10.1109/ICCR61006.2024.10532983

[**20**] Manavi, F., & Hamzeh, A. (2020). A New Method for Ransomware Detection Based on PE Header Using Convolutional Neural Networks. *2020 17th International ISC Conference on Information Security and Cryptology (ISCISC)*, 82-87. https://doi.org/10.1109/ISCISC51277.2020.9261903

[**21**] Manavi, F., & Hamzeh, A. (2021). Static Detection of Ransomware Using LSTM Network and PE Header. *2021 26th International Computer Conference, Computer Society of Iran (CSICC)*, 1-5. https://doi.org/10.1109/CSICC52343.2021.9420580

[**22**] Medhat, M., Gaber, S., & Abdelbaki, N. (2018). A New Static-Based Framework for Ransomware Detection. *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/-DataCom/CyberSciTech)*, 710-715. https://doi.org/10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00124

[**23**] Microsoft. (s.f.[a]). *Microsoft Defender ATP*. Consultado el 1 de enero de 2024, desde https://learn.microsoft.com/es-mx/connectors/wdatp/

[**24**] Microsoft. (s.f.[b]). *What is .NET?* Consultado el 1 de enero de 2024, desde https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet

[**25**] Microsoft. (s.f.[c]). *Windows 10 release information*. Consultado el 1 de enero de 2024, desde https://learn.microsoft.com/en-us/windows/release-health/release-information

[**26**] MINISTERIO DE CIENCIA, TECNOLOGÍA E INNOVACIÓN. (2021, noviembre). *NIVELES DE MADUREZ TECNOLÓGICA (TRL) Y DE MANUFACTURA (MRL)* (inf. téc.). MINISTERIO DE CIENCIA, TECNOLOGÍA E INNOVACIÓN. Bogotá. https://

minciencias.gov.co/sites/default/files/upload/convocatoria/anexo_9_-_niveles_de_madurez_tecnologica_y_de_manufactura-_trl_y_mrl.pdf

[**27**] Mukesh, S. D. (2018). An Analysis Technique to Detect Ransomware Threat. *2018 International Conference on Computer Communication and Informatics (ICCCI)*, 1-5. https://doi.org/10.1109/ICCCI.2018.8441502

[**28**] Palo Alto Networks. (s.f.). *Ransomware Protection - Cortex*. Consultado el 1 de enero de 2024, desde https://www.paloaltonetworks.com/cortex/ransomware-protection

[**29**] Panda Security. (s.f.). *¿Por qué Adaptive Defense 360?* Consultado el 1 de enero de 2024, desde https://www.pandasecurity.com/es/mediacenter/por-que-adaptive-defense-360/

[**30**] Pérez, P. G. (2020, noviembre). *OSSEC: El mundo del IDS (Intrusion Detection System) y del HIDS (Host IDS)*. Consultado el 1 de enero de 2024, desde https://www.elladodelmal.com/2020/11/ossec-el-mundo-del-ids-intrusion.html

[**31**] Sgandurra, D., Muñoz-González, L., Mohsen, R., & Lupu, E. C. (2016). Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection. https://doi.org/https://doi.org/10.48550/arXiv.1609.03020

[**32**] Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, *27*(3), 379-423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

[**33**] Sophos. (2024, abril). *The State of Ransomware 2024* (inf. téc.). Sophos. https://assets.sophos.com/X24WTUEQ/at/9brgj5n44hqvgsp5f5bqcps/sophos-state-of-ransomware-2024-wp.pdf

[**34**] Statcounter GlobalStats. (2024). *Desktop Windows Version Market Share Worldwide Mar 2024 - Mar 2025*. Consultado el 1 de enero de 2024, desde https://gs.statcounter.com/windows-version-market-share/desktop/worldwide/%5C#monthly-202403-202503

[**35**] Urooj, U., Maarof, M. A. B., & Al-rimy, B. A. S. (2021). A proposed Adaptive Pre-Encryption Crypto-Ransomware Early Detection Model. *2021 3rd International Cyber Resilience Conference (CRC)*, 1-6. https://doi.org/10.1109/CRC50527.2021.9392548

[**36**] Zanoramy, W., Abdollah, M. F., Abdollah, O., & S.M.M, S. W. M. (2024). Ransomware Early Detection using Machine Learning Approach and Pre-Encryption Boundary Identification. *Journal of Advanced Research in Applied Sciences and Engineering Technology*, *47*(2), 121-137. https://doi.org/10.37934/araset.47.2.121137

Capítulo 9

Anexos

Los anexos que soportan esta investigación se pueden encontrar en los siguientes repositorios y enlaces:

Repositorio del Proyecto

Repositorio GitHub: RansomwareDetectionSoftware
`https://github.com/ClusterMax/RansomwareDetectionSoftware`

Historias de Usuario

Historias de Usuario
`https://drive.google.com/`

Encuestas de Usabilidad y Compatibilidad

Resultados de las encuestas
`https://drive.google.com/`

Participación en el IV Encuentro Regional de Tesistas

Certificado de participación
`hhttps://drive.google.com/`

Presentación del IV Encuentro Regional de Tesistas

Presentación del prototipo
`https://invessoft.app/`

Herramienta de Simulación de Ransomware

Repositorio GitHub: RanSim
`https://github.com/lawndoc/RanSim`

Archivos para Pruebas (Dummy Files)

Repositorio GitHub: sample-files
https://github.com/