UNIVERSIDAD DE COSTA RICA
SISTEMA DE ESTUDIOS DE POSGRADO

IMPLEMENTACIÓN DE UN SISTEMA DE DETECCIÓN DE URLS MALICIOSAS EN TIEMPO REAL

Trabajo Final de Investigación Aplicada sometido a la consideración de la Comisión del Programa de Estudios de Posgrado en Computación e Informática para optar al grado y título de Maestría Profesional en Computación e Informática

DIEGO OROZCO FONSECA

Ciudad Universitaria Rodrigo Facio, Costa Rica
2024

Agradecimientos

Quiero expresar mi agradecimiento a todas las personas que han sido fundamentales en la realización de este trabajo.

A mis padres, quienes con su apoyo incondicional y amor fueron las bases sobre las que he construido todos mis logros. Su ejemplo y enseñanzas han sido el motor que me ha impulsado a seguir adelante y alcanzar cada una de mis metas.

A mi esposa, por estar a mi lado en cada momento de este proceso. Su apoyo, comprensión y ánimo me han dado la fuerza necesaria para superar los desafíos y continuar avanzando con determinación.

A mi profesor guía, por su paciencia, dedicación y valiosas recomendaciones. Su orientación ha sido de gran valor en cada etapa de este trabajo, ayudándome a perfeccionar mi investigación.

"Este Trabajo Final de Investigación Aplicada es presentada a la Comisión del Programa de Estudios de Posgrado en Computación e Informática de la Universidad de Costa Rica, como requisito parcial para optar por el grado y título de Maestría Profesional en Computación e Informática."

ELENA GABRIELA BARRANTES SLIESARIEVA (FIRMA)
PERSONA FISICA, CPF-01-0866-0165.
Fecha declarada: 20/11/2024 04:03:50 PM
Esta es una representación gráfica únicamente, verifique la validez de la firma.

Dra. Gabriela Barrantes Sliesarieva
Representante de la Decana
Sistema de Estudios de Posgrado

ADRIAN LARA PETITDEMANGE (FIRMA)
PERSONA FISICA, CPF-01-1225-0679.
Fecha declarada: 20/11/2024 02:07:31 PM
Esta es una representación gráfica únicamente, verifique la validez de la firma.

Dr. Adrián Lara Petitdemange
Profesor Guía

LUIS JOSE QUESADA QUIROS (FIRMA)
PERSONA FISICA, CPF-02-0536-0228.
Fecha declarada: 20/11/2024 02:11:50 PM
Esta es una representación gráfica únicamente, verifique la validez de la firma.

Dr. Luis Quesada Quirós
Lector

KRYSCIA RAMIREZ BENAVIDES (FIRMA)
PERSONA FISICA, CPF-01-1033-0144.
Fecha declarada: 20/11/2024 02:18:41 PM
Lugar: San José Contacto: kryscia.ramirez@ucr.ac.cr

Dra. Kryscia Ramirez Benavides
Lectora

GUSTAVO LOPEZ HERRERA (FIRMA)
Firmado digitalmente por GUSTAVO LOPEZ HERRERA (FIRMA)
Fecha: 2024.11.20 16:23:55 -06'00'

Dr. Gustavo López Herrera
Director
Programa de Posgrado en Computación e Informática

DIEGO OROZCO FONSECA (FIRMA)
Digitally signed by DIEGO OROZCO FONSECA (FIRMA)
Date: 2024.11.20 13:46:03 -06'00'

Diego Orozco Fonseca
Sustentante

Tabla de contenido

Resumen

En el contexto actual de crecientes amenazas cibernéticas, la detección efectiva y rápida de URLs maliciosas es crucial para proteger a los usuarios y a las organizaciones de ataques como phishing, distribución de *malware* y robo de información confidencial.

Para abordar esta problemática, se desarrolló un prototipo de proxy que integra dos técnicas de detección principales: la clasificación basada en listas negras y el aprendizaje automático supervisado (*Machine Learning*). Este proxy fue probado en dos entornos distintos: uno local, que representa la infraestructura interna de una organización, y otro en la nube, utilizando *Amazon Web Services* (*AWS*) para representar un entorno virtualizado. El objetivo era comparar el rendimiento y la latencia de estas técnicas en ambos entornos bajo diferentes cargas de solicitudes.

En la fase de implementación, se configuró un proxy personalizado que redirige todas las solicitudes de URL hacia el mecanismo de detección correspondiente. Para evaluar el impacto en la latencia, se evaluó cómo la ubicación (local o en la nube) y la técnica de detección (listas negras o aprendizaje automático) influían en el tiempo de respuesta del sistema. Además. se centró en la capacidad de carga del sistema, incrementando progresivamente el número de solicitudes simultáneas hasta identificar el punto en que el proxy comenzaba a perder conexiones.

Los hallazgos de esta investigación subrayan la importancia de seleccionar la infraestructura adecuada para la implementación de sistemas de detección de URLs maliciosas en tiempo real. La infraestructura en la nube demostró ser superior en términos de reducir la latencia y manejar cargas de trabajo intensivas, lo cual es esencial para aplicaciones críticas donde el tiempo de respuesta es fundamental.

Lista de figuras

Lista de tablas

Lista de abreviaturas

ML	Machine Learning
AAS	Aprendizaje Automático Supervisado
DNS	Domain Name System
BL	Black List

CAPÍTULO 1: INTRODUCCIÓN

En la actualidad, el aumento exponencial de amenazas cibernéticas representa un desafío significativo para la seguridad de los sistemas informáticos y la protección de los usuarios en línea [1]. Entre estas amenazas, las URLs maliciosas se han convertido en una preocupación constante, ya que los atacantes las utilizan como herramientas para llevar a cabo actividades delictivas, como el *phishing*, la distribución de *malware* y el robo de información confidencial [2][13].

El Sistema de Nombres de Dominio (DNS) es responsable de traducir los nombres de dominio legibles por humanos en direcciones IP que las computadoras puedan entender [4]. Esta función es esencial para el correcto funcionamiento de servicios en línea, como el acceso a sitios web, el envío de correos electrónicos y otras actividades de comunicación.

Para garantizar la seguridad en línea, es fundamental contar con mecanismos efectivos de detección de URLs maliciosas en los sistemas de nombres de dominio [6][8]. La detección temprana y precisa de estas URLs puede prevenir el acceso a contenido malicioso y proteger a los usuarios de posibles ataques.

En la mayoría de las redes tradicionales, las URLs no se clasifican y filtran, lo que aumenta la posibilidad de ataques de *phishing*, *botnets* y otras amenazas maliciosas [1]. Para abordar este problema, se han desarrollado numerosos clasificadores de URLs basados en aprendizaje automático supervisado (AAS). Sin embargo, hay pocos estudios sobre la viabilidad de introducir la clasificación de URLs en tiempo real en un DNS debido al posible impacto en la latencia [1].

Sin embargo, la implementación de técnicas de detección en tiempo real en el DNS plantea desafíos adicionales, como el impacto en la latencia del sistema [14]. La latencia, que

se refiere al tiempo que tarda un sistema en responder a una solicitud, es un factor crítico en la experiencia del usuario y el rendimiento de las aplicaciones en línea. Cualquier modificación o introducción de procesos adicionales en el DNS puede afectar la latencia y, por lo tanto, la velocidad de respuesta del sistema.

En este trabajo, se configuró un proxy como entidad intermediaria que actúa como un servidor intermedio entre los usuarios y el servidor final al que están tratando de acceder. Su función principal es recibir y redirigir solicitudes entre estos dos extremos, actuando como un intermediario para facilitar la comunicación. El proxy se configuró específicamente para manejar solicitudes relacionadas con la detección de URLs maliciosas en tiempo real.

El objetivo de este trabajo fue investigar la factibilidad de la detección de URLs en tiempo real y cómo diferentes aspectos pueden afectar la latencia. Para lograr esto, se implementó un proxy capaz de recibir y redirigir solicitudes en un entorno de producción real, tanto en una ubicación local como en la nube [7][9][11], utilizando tanto la clasificación basada en listas negras como modelos de aprendizaje automático supervisado [5][15]. Además, se realizaron pruebas bajo diferentes cargas de solicitudes para determinar la capacidad y resistencia del proxy. Es importante destacar que, aunque existen enfoques asincrónicos relacionados con la detección de URLs en la literatura, tanto utilizando ML como BL, esta investigación representó un paso adelante al poner a prueba estos métodos en un escenario de tiempo real.

Los resultados obtenidos permitieron comprender el impacto de cada enfoque en la latencia y proporcionaron información valiosa para el diseño y la implementación de sistemas de detección de URLs maliciosas en el DNS.

1.1 Preguntas de investigación

Las preguntas de investigación son las siguientes:

1. ¿La ubicación del proxy tiene un impacto significativo en la latencia del sistema de detección de URLs maliciosas en tiempo real, considerando tanto una ubicación local como en la nube?
2. ¿La elección de la técnica de detección (clasificación basada en listas negras o modelos de aprendizaje automático supervisado) afecta la latencia del sistema en una configuración en tiempo real?
3. ¿Cuál es la capacidad de carga efectiva del proxy implementado en el sistema de detección de URLs maliciosas en tiempo real?

1.2 Objetivos

El objetivo de esta investigación es evaluar cómo la ubicación, el tipo de técnica usada y la carga recibida pueden afectar el tiempo de respuesta a la hora de integrar un clasificador de URLs en tiempo real en un sistema de DNS.

Para alcanzar este objetivo general, se plantean los siguientes objetivos específicos:

1. Identificar a través de una revisión literaria las técnicas comunes utilizadas en la detección de URLs maliciosas.
2. Desarrollar un proxy que sea capaz de ejecutarse en diversas ubicaciones y permita la integración de distintas técnicas de clasificación de URLs.
3. Evaluar el impacto en la latencia del sistema de detección de URLs maliciosas considerando la ubicación, la técnica de detección y la cantidad de solicitudes simultáneas.

1.3 Justificación

Esta investigación desempeña un papel importante al abordar vacíos significativos en el ámbito de la detección de URLs maliciosas. Su relevancia se destaca por varias razones.

La implementación de un sistema de detección de URLs maliciosas en tiempo real en un entorno de producción constituye un avance innovador. Actualmente, existe una carencia en la literatura y en las soluciones prácticas que aborden la detección de URLs maliciosas en tiempo real, especialmente considerando una implementación en un entorno de producción real. Este trabajo propone llenar este vacío, ofreciendo una contribución tangible al campo de la ciberseguridad.

El producto resultante de este trabajo no solo se traduce en un avance tecnológico, sino que también beneficia directamente a organizaciones e individuos que dependen de la seguridad en línea. Proporciona una herramienta práctica para la detección proactiva de amenazas en tiempo real, mejorando la capacidad de respuesta y protegiendo a los usuarios contra ataques maliciosos. Cabe destacar que el enfoque de esta investigación busca específicamente minimizar la latencia, permitiendo respuestas más rápidas ante posibles amenazas.

En el ámbito de la ciberseguridad, la detección de URLs maliciosas es una medida para proteger a los usuarios de posibles amenazas. Por dicho motivo, esta investigación aborda la necesidad de proteger la infraestructura digital frente a la creciente sofisticación de los ataques cibernéticos. La detección de URLs maliciosas en tiempo real no solo mejora la seguridad de los sistemas informáticos, sino que también incrementa la confianza de los usuarios en los servicios en línea.

La contribución principal de este trabajo radica en el desarrollo y evaluación de un proxy configurado para manejar solicitudes en tiempo real, utilizando tanto listas negras como técnicas de aprendizaje automático supervisado. Este enfoque experimental permite una comprensión del impacto de diferentes técnicas y configuraciones en la latencia del sistema, proporcionando datos valiosos que pueden ser utilizados para optimizar y mejorar las soluciones de seguridad existentes.

Además, este estudio no solo se centra en los aspectos técnicos de la detección de URLs maliciosas, sino que también considera la aplicabilidad práctica de las soluciones propuestas. La inclusión de pruebas bajo diferentes cargas de solicitudes asegura que el sistema desarrollado pueda adaptarse a diversas condiciones operativas, ofreciendo una solución robusta y escalable que puede ser implementada en entornos de producción reales.

Esta investigación realiza aportes al campo de la seguridad informática al desarrollar un sistema para la detección de amenazas en tiempo real. Su contribución principal es establecer un marco experimental sólido que permitirá evaluar diversas técnicas de detección de URLs maliciosas bajo diversas condiciones. Este enfoque no solo aborda desafíos técnicos, sino que también proporciona soluciones prácticas, beneficiando directamente a la comunidad de la seguridad.

1.4 Estructura

Esta sección detalla la estructura propuesta para el documento.

El Capítulo 2 es sobre el marco conceptual. Proporciona un marco conceptual integral sobre la detección de URLs maliciosas. Se exploran los conceptos clave relacionados con la detección de amenazas en línea, como el *phishing*, el *malware* y otros ataques basados en URLs maliciosas. Además, se presentan las técnicas y enfoques utilizados en la detección de

URLs maliciosas, incluyendo la clasificación basada en lista negra y la clasificación basada en aprendizaje automático supervisado.

El siguiente es Capítulo 3, el cual trata sobre los Antecedentes y Estado del Arte. En este capítulo, se revisan los antecedentes y el estado del arte de la detección de URLs maliciosas. Se examinan estudios y trabajos previos relacionados con el tema, incluyendo investigaciones académicas, informes técnicos y artículos científicos. Se analizan los enfoques y métodos utilizados en investigaciones anteriores, así como los conjuntos de datos y técnicas de evaluación utilizados para medir el rendimiento de los sistemas de detección.

El Capítulo 4 es sobre la metodología. En este capítulo, se presenta la metodología utilizada en este estudio. Se describen en detalle los conjuntos de datos utilizados, incluyendo su origen y características. Además, se explican los enfoques de detección de URLs maliciosas implementados, así como las técnicas de preprocesamiento de datos y extracción de características utilizadas. También se describe el diseño experimental y los procedimientos de evaluación utilizados para comparar y medir el rendimiento de los diferentes enfoques de detección.

Posteriormente, en el Capítulo 5 se mencionan los resultados. En este capítulo se presentan los resultados obtenidos de los experimentos realizados. Se incluye un análisis exhaustivo del impacto en la latencia del sistema de detección de URLs maliciosas, considerando la ubicación (entorno local y en la nube), las técnicas de detección (listas negras y aprendizaje automático), y la capacidad de carga del sistema. Además, se discuten las diferencias observadas en la latencia y tasa de fallos entre los distintos enfoques y entornos. Los resultados se complementan con un análisis estadístico detallado, utilizando un modelo mixto de efectos para validar los hallazgos. Estos resultados proporcionan una comprensión

profunda del rendimiento y la fiabilidad de los métodos de detección implementados en diferentes condiciones experimentales.

Finalmente, el Capítulo 6 es sobre las conclusiones. Este capítulo resume los hallazgos clave del estudio, discutiendo las implicaciones de los resultados y ofreciendo recomendaciones para futuras investigaciones y aplicaciones prácticas. Se reflexiona sobre la importancia de seleccionar el entorno y el método de detección adecuados según los requisitos de rendimiento y confiabilidad del sistema. Además, se destacan las principales contribuciones de la investigación y se sugieren posibles mejoras y líneas de investigación futuras.

Posteriormente en el Capítulo 7 se mencionan las referencias. En este capítulo se presenta una lista completa de todas las fuentes y referencias bibliográficas utilizadas a lo largo del documento. Esto incluye artículos académicos, informes técnicos, libros y otras fuentes relevantes que han respaldado la investigación y el desarrollo de este estudio.

CAPÍTULO 2: MARCO CONCEPTUAL

En este capítulo se presenta el marco conceptual que servirá como base teórica para la metodología propuesta en este trabajo. Se abordarán los conceptos clave relacionados con el Sistema de Nombres de Dominio (DNS), el controlador en el DNS, así como los enfoques de detección de URLs maliciosas, incluyendo la clasificación tradicional basada en lista negra y el aprendizaje automático supervisado (AAS).

2.1 Sistema de nombres de dominio (DNS)

El Sistema de Nombres de Dominio es una infraestructura de Internet que se encarga de traducir los nombres de dominio legibles para los humanos en direcciones IP numéricas utilizadas por las computadoras para comunicarse en la red [5].

El DNS actúa como una especie de "libro de direcciones" de Internet, donde los nombres de dominio, como "www.ejemplo.com", se traducen en direcciones IP numéricas, como "192.0.2.1". Para llevar a cabo esta traducción, el DNS opera en una jerarquía que incluye servidores raíz, servidores de dominio de nivel superior (TLD), servidores autorizados y servidores locales. Este sistema de capas permite una distribución eficiente y escalable de la carga de trabajo asociada con la resolución de nombres de dominio [19].

Para resolver un dominio, un cliente normalmente necesita consultar un servidor DNS recursivo local (RDNS). Un servidor recursivo descubre de forma iterativa qué servidor de nombres autorizado (ANS) es responsable de cada zona. El resultado típico de este proceso iterativo es la asignación entre el nombre de dominio solicitado y sus direcciones IP actuales.

2.2 URLs maliciosas

Cuando una URL se emplea con intenciones maliciosas, se entiende que ha sido diseñada deliberadamente para causar daño y representa un riesgo elevado de amenaza para el usuario

final. Por ejemplo, aquellos sitios con categorías de *malware* y *phishing* son comúnmente asociadas a tener un alto riesgo y por ende se bloquean en tecnologías de filtrado [6].

Los comportamientos más comunes son *phishing*, *botnet*, *spam* y *malware* [7]. Para cada categoría hay técnicas comunes que son usadas por los atacantes en la construcción de su URL que pueden ayudar a diferenciarlas de una URL legitima. Podemos listar las siguientes [8]:

- ***Phishing*:** cambios en el Dominio de nivel superior, por sus siglas en ingles TLD, similitud de suplantación de marca, ofuscación mediante URL largas.
- ***Spam*:** uso de palabras atrayentes, uso de identificadores únicos para evadir filtros de spam.
- ***Botnets*:** algoritmos de generación automática (DGA), uso de *Fast-Flux* y *Domain-Flux.*
- ***Malware*:** uso de redireccionamiento, incluye el nombre del archivo en sus URL de descarga, uso de IP para evitar resolución del dominio.

En el marco de esta investigación, se debe de comprender los diferentes factores que afectan el desempeño de un sistema de detección de URLs maliciosas en tiempo real. Entre estos factores, la latencia juega un papel central, ya que determina la rapidez con la que un sistema puede identificar y mitigar amenazas. A continuación, se explorará en detalle el concepto de latencia en la detección de URLs, para ofrecer una visión completa de su importancia y los desafíos asociados a su optimización.

2.3 Latencia en la detección de URLs

La latencia, es el tiempo total que tarda un paquete de datos en ser transmitido desde el origen hasta el destino en una red, es influenciada por factores como la frecuencia de

muestreo y el número de paquetes, y es crucial para el desempeño en tiempo real en aplicaciones [25] [26].

En el contexto de la detección de URLs maliciosas, se refiere al tiempo que tarda un sistema en procesar una solicitud y devolver una respuesta. Este tiempo de respuesta incluye el tiempo necesario para, realizar la predicción sobre si el tráfico es maligno o no, el tiempo de consulta al DNS en caso de que sea benigno y el tiempo de vuelta al cliente con la respuesta respectiva [27]. Una latencia baja es importante, ya que las decisiones rápidas y precisas son necesarias para proteger a los usuarios contra amenazas en línea.

La latencia puede verse afectada por varios factores, como el tamaño de la lista negra utilizada, la complejidad de los algoritmos de aprendizaje automático y la ubicación del servidor (local o en la nube).

A continuación, se presentará el concepto de controlador en el DNS, que actúa como intermediario entre los clientes y los servidores de nombres de dominio. Se discutirá su rol en la gestión de las consultas DNS y su capacidad para implementar técnicas de detección de URLs maliciosas.

2.4 Controlador en el DNS

Un controlador en el DNS se refiere a un componente o software responsable de gestionar y controlar el funcionamiento de un Sistema de Nombres de Dominio. Según la investigación realizada por Liu [9], un controlador en el DNS es responsable de recibir las consultas de resolución de nombres de dominio, procesarlas y proporcionar respuestas precisas a los clientes. Además, un controlador en el DNS también puede incorporar funcionalidades adicionales, como la detección de URLs maliciosas. Estas capacidades de detección pueden

integrarse en el controlador para analizar y clasificar las URLs en tiempo real, con el fin de proteger a los usuarios finales de posibles amenazas cibernéticas.

A continuación, se explican las dos técnicas de detección de URLs maliciosas más usadas.

2.5 Clasificación tradicional basada en lista negra

La clasificación tradicional basada en lista negra es un enfoque comúnmente utilizado en la detección de URLs maliciosas. Consiste en mantener una lista de URLs previamente identificadas como maliciosas y comparar las nuevas URLs con esta lista para determinar si son potencialmente peligrosas.

Este enfoque se basa en la premisa de que las URLs maliciosas tienden a repetirse en diferentes ataques y pueden estar asociadas con dominios o patrones específicos. Al utilizar una lista negra de URLs conocidas, se busca identificar y bloquear aquellas URLs que ya han sido identificadas como maliciosas en el pasado.

Según los estudios recientes, la clasificación tradicional basada en lista negra ha demostrado ser efectiva en la detección de URLs maliciosas. En un estudio realizado por Joshi [10], se evaluó la eficacia de diferentes enfoques de detección de URLs maliciosas y que el método de detección tradicional basado en listas negras sigue siendo una técnica clásica en la detección de URL maliciosas debido a su capacidad para identificar y bloquear URL con reputación conocida de ser maliciosas.

Algunas de las heurísticas clave utilizadas para detectar URL maliciosas en las listas negras son: longitud de la URL, dominios similares, palabras clave, redirecciones, dominios emergentes y ubicación geográfica del servidor [4].

2.6 Aprendizaje automático supervisado (AAS)

El aprendizaje automático supervisado es una rama de la inteligencia artificial que utiliza algoritmos para entrenar modelos a partir de datos etiquetados previamente [11]. En el contexto de la detección de URLs maliciosas, se utilizan algoritmos de AAS para construir clasificadores capaces de identificar automáticamente si una URL es maliciosa o no.

En el AAS, los modelos de aprendizaje automático se construyen mediante algoritmos que aprenden patrones y características de los datos de entrenamiento. Estos datos de entrenamiento consisten en ejemplos etiquetados, donde cada ejemplo se compone de una URL y una etiqueta que indica si la URL es maliciosa o no [12].

Los algoritmos más destacados en términos de clasificación son los bosques aleatorios (RF) y Máquinas de Soporte Vectorial (SVM). A continuación, se presenta una breve descripción de cada uno [13]:

- **Bosques aleatorios**: Los bosques aleatorios están compuestos por múltiples árboles de decisión, cada uno entrenado con conjuntos de datos diferentes. El modelo final se obtiene mediante la combinación o promedio de las decisiones tomadas por cada árbol de decisión individual.
- **Máquinas de Soporte Vectorial:** La idea principal detrás de las SVM es encontrar un hiperplano óptimo que maximice la separación entre las clases de datos en un espacio de características. Los SVM son eficaces en la clasificación de datos linealmente separables

2.7 Proxy

El servidor proxy es una herramienta multifuncional en la gestión del tráfico web, proporciona acceso eficiente, mejorando los tiempos de respuesta y elevando el nivel de

seguridad para los usuarios que utilizan la red. Aunque la dinámica de la web evoluciona, la relevancia del servidor proxy persiste al optimizar el rendimiento y resguardar la seguridad de la red [22].

En esencia, un proxy es un servidor intermedio entre un cliente y otros servidores. Al recibir una solicitud del cliente, procesa la solicitud y luego la transmite al servidor correspondiente. La respuesta del servidor retorna al proxy, que finalmente la envía al cliente. Este proceso posibilita diversas funcionalidades [23]:

1. **Mejora del Rendimiento:** El proxy tiene la capacidad de almacenar en caché las respuestas de los servidores, permitiendo proporcionar respuestas almacenadas en lugar de enviar nuevas solicitudes al servidor cuando el cliente solicita información ya obtenida.
2. **Privacidad:** La capacidad del proxy para ocultar la dirección IP del cliente al enviar solicitudes a los servidores contribuye a resguardar la privacidad del usuario.
3. **Seguridad:** Los proxies actúan como una barrera de seguridad, funcionando como firewall para bloquear accesos no autorizados a ciertos servidores o sitios web. Además, tienen la capacidad de filtrar el tráfico para identificar y bloquear amenazas de seguridad.

En el siguiente capítulo, se abordarán los antecedentes relacionados con la detección de URLs maliciosas y el impacto en la latencia de los sistemas de nombres de dominio.

CAPÍTULO 3: ANTECEDENTES

En esta sección, se presenta los antecedentes relevantes relacionados con la detección de URLs maliciosas y el impacto en la latencia de los sistemas de nombres de dominio. Se abordarán los avances previos en la clasificación de URLs, el uso de técnicas tradicionales

basadas en lista negra y los estudios sobre la influencia de la latencia en los sistemas de nombres de dominio.

3.1 Características de las URLs

Las características de las URL se dividen en dos categorías principales: estáticas y dinámicas. Las características estáticas son atributos que se pueden obtener sin interactuar directamente con la URL. Por otro lado, las características dinámicas requieren una interacción más profunda con la URL y, aunque pueden lograr una alta precisión en la detección de URL maliciosas, su recopilación a menudo consume tiempo y recursos, lo que las hace menos adecuadas para el análisis en tiempo real [20].

En el ámbito de la detección de dominios maliciosos, algunos métodos se basan por completo en el uso de características léxicas. Estos enfoques se centran en atributos como la proporción de caracteres significativos, puntuaciones de n-gramas, longitud de nombres de subdominio y el número de intercambios de caracteres alfanuméricos. Además, utilizan técnicas como TF-IDF (Frecuencia de Término-Inversa de Frecuencia en Documentos) para extraer términos relevantes y aplican métodos de reducción dimensional, como el análisis de componentes principales (PCA), para obtener vectores representativos [11].

En nuestro artículo, "Taxonomía de Técnicas de Detección de URL Maliciosas", propusimos una taxonomía de métodos de detección basados en las características descritas anteriormente [4]. En esta taxonomía, exploramos las diferentes formas en que estos métodos trabajaban con las características de las URL, facilitando la comprensión y comparación de las distintas técnicas de detección y sus enfoques en relación con las propiedades de las URL. Además, aborda la importancia y la complejidad de detectar URLs maliciosas, se

identificaron dos enfoques principales para la detección de URLs maliciosas: las técnicas basadas en reglas y las técnicas basadas en aprendizaje automático

Como parte del trabajo desarrollado en este TFIA, se proporcionó un mayor detalle sobre esta taxonomía en la sección de resultados, donde se presentaron y analizaron de manera exhaustiva los hallazgos y las conclusiones derivadas de este estudio.

3.2 Detección de URLs maliciosas a través de modelos de aprendizaje automático supervisado

La detección de URLs maliciosas ha sido un tema de gran importancia en el campo de la seguridad informática. La proliferación de amenazas en línea, como el *phishing*, los ataques de *botnets* y el *malware,* ha llevado al desarrollo de diversas técnicas y enfoques para combatir estas amenazas. Entre ellos, los modelos de aprendizaje automático supervisado (AAS) han demostrado ser efectivos en la detección de URLs maliciosas [13].

Las técnicas de detección basadas en el aprendizaje automático analizan características específicas de las URL y determinan si son maliciosas o benignas. Los clasificadores consideran características como la longitud de la URL, el dominio y subdominio, palabras clave y seguimiento de redirecciones, entre otros. Además, se exploran técnicas avanzadas como el aprendizaje profundo y métodos de ensamblaje como *Random Forests* y *AdaBoost* para mejorar la precisión en la detección [20][11][7].

La clasificación de URLs mediante AAS ha sido ampliamente estudiada en la literatura. Estos modelos utilizan conjuntos de datos etiquetados para entrenar algoritmos de aprendizaje automático y lograr la detección de URLs maliciosas con altos niveles de precisión y *recall* [14]. Diversos enfoques de aprendizaje automático supervisado han sido

aplicados en la detección de URLs maliciosas, como algoritmos basados en árboles de decisión, redes neuronales, máquinas de vectores de soporte, entre otros [8].

Sin embargo, a pesar de la efectividad de estos modelos en términos de precisión y *recall*, hay pocos estudios que evalúen el impacto en la latencia de la introducción de la clasificación de URLs en tiempo real en los sistemas de nombres de dominio [13].

3.3 Detección de URLs maliciosas a través de modelos basados en reglas

El uso de listas negras para detectar URL maliciosas a través de técnicas tradicionales se emplea comúnmente debido a su eficacia y simplicidad. Las listas negras contienen registros de URL maliciosas previamente identificadas. Según Joshi et al. [10], el método tradicional de detección basado en listas negras sigue siendo una técnica clásica en la detección de URL maliciosas debido a su capacidad para identificar y bloquear URL con reputaciones conocidas de ser maliciosas.

Investigaciones previas han examinado y evaluado técnicas basadas en reglas, como listas negras y listas blancas, en términos de su efectividad en la detección de URL maliciosas. Sin embargo, estos enfoques pueden tener limitaciones, especialmente en lo que respecta a la latencia que generan. El procesamiento basado en reglas puede requerir un tiempo significativo para consultar una lista de URL conocidas y tomar decisiones basadas en esa lista [22].

Estudios demuestran la eficacia de la lista negra como un enfoque de clasificación para identificar URLs maliciosas conocidas [10]. Sin embargo, también se señala que este enfoque puede ser limitado debido a la falta de actualización frecuente de las listas negras, lo que dificulta la detección de nuevas amenazas.

3.4 Latencia en la detección de URLs maliciosas

La latencia se refiere al tiempo que tarda un sistema en procesar una solicitud y devolver una respuesta. En el contexto de la detección de URLs maliciosas, la latencia juega un papel impactante, ya que es necesario tomar decisiones rápidas y precisas para proteger a los usuarios contra amenazas en línea.

Determinar el tipo de algoritmo más adecuado a utilizar en la propuesta requiere considerar varios factores, como la eficiencia computacional y el rendimiento en tiempo real. Algunos algoritmos de aprendizaje automático, como los basados en árboles de decisión, son conocidos por su capacidad de procesamiento rápido y su baja latencia [16]. Por otro lado, las redes neuronales pueden ofrecer una mayor precisión, pero pueden requerir más recursos computacionales y, por lo tanto, pueden generar una mayor latencia en los sistemas de nombres de dominio [7].

En cuanto al impacto en la latencia de los sistemas de nombres de dominio, se ha observado que la introducción de nuevos mecanismos de detección de URLs maliciosas puede afectar el tiempo de respuesta de los DNS [17]. El aumento de la latencia puede ser problemático en entornos donde se requiere una resolución rápida de nombres de dominio, como en aplicaciones críticas o en redes con alto tráfico. Por lo tanto, es importante evaluar y comparar el impacto en la latencia de los diferentes enfoques de detección de URLs maliciosas, incluyendo la no clasificación, la clasificación basada en lista negra y la incorporación de modelos de aprendizaje automático en controladores DNS en tiempo real.

Hamidouche ha realizado una contribución significativa al entender las implicaciones de latencia en dispositivos con recursos limitados. Su estudio se centra en implementar un modelo ligero para la detección de túneles DNS, optimizado para dispositivos con recursos

computacionales limitados, como los dispositivos IoT. Un hallazgo clave de su investigación es la importancia de mantener las latencias de procesamiento por debajo de 1 milisegundo para asegurar la detección y respuesta a amenazas de manera oportuna. Este estudio resalta la necesidad de mecanismos de detección efectivos que no sacrifiquen el rendimiento al integrarse en sistemas en tiempo real [27].

En el siguiente capítulo, se describirá la metodología propuesta.

CAPÍTULO 4: METODOLOGÍA

El presente estudio se basa en un enfoque mixto de investigación, combinando elementos de la investigación exploratoria y la investigación aplicada. La investigación exploratoria permite comprender el impacto de diferentes enfoques de detección de URLs maliciosas en tiempo real, mientras que la investigación aplicada busca aplicar estos conocimientos en un entorno controlado y evaluar su efectividad.

Este trabajo es la continuación del proyecto del curso PF-3885 Diseño de Experimentos. En dicho curso se diseñó e implementó un experimento que sirvió como base para evaluar la latencia en un sistema de detección de URLs maliciosas. Los principios y metodologías enseñados en el curso fueron fundamentales para el diseño y ejecución de los experimentos realizados en este estudio. En el anexo 3 se puede consultar la prueba piloto realizada en el curso mencionado, donde se

El objetivo general de esta investigación fue evaluar cómo la ubicación, el tipo de técnica usada y la carga recibida pueden afectar el tiempo de respuesta al integrar un clasificador de URLs en tiempo real en un sistema de DNS.

A continuación, se detalla la metodología que se siguió en este trabajo. Primero, se realizó una revisión de literatura, descrita en la sección 4.1. En la sección 4.2, se detallan los pasos para implementar el clasificador. Finalmente, concluimos con la metodología creada para la evaluación del modelo en la sección 4.3.

4.1 Revisión de literatura (Objetivo Específico 1)

Dado que ya existía una revisión sistemática de literatura previa, el enfoque de este trabajo no fue realizar una nueva revisión sistemática, sino actualizarla con los artículos más

recientes de los últimos años. Esta actualización permitió identificar las técnicas y metodologías más actuales en la detección de URLs maliciosas. Como resultado de esta actualización, se desarrolló una propuesta de taxonomía que clasifica las características de las URLs más utilizadas en diferentes tipos de aprendizaje automático. Los resultados de esta taxonomía se presentarán en detalle en la sección de resultados y se encuentra publicados en Orozco et al. [4] (Anexo 2).

4.1.1 Criterios de elegibilidad

En esta sección, se describen los criterios de elegibilidad y las fuentes de información utilizadas para llevar a cabo una revisión de literatura. El objetivo fue identificar las técnicas comunes utilizadas en la detección de URLs maliciosas. Se incluyeron estudios de años específico, usando varias técnicas de aprendizaje automático y otros criterios importantes explicados a continuación:

4.1.1.1 Año de publicación

Se incluyeron exclusivamente estudios publicados a partir del año 2022. Esta selección se basa en el rápido avance y la creciente popularidad de los algoritmos de aprendizaje profundo supervisado en los últimos años. Según investigaciones, la cantidad de artículos publicados en este campo en 2020 superó significativamente a los de 2019, destacando la evolución y relevancia actual del tema [29]. Consultar literatura anterior a 2022 podría resultar obsoleta en términos de metodologías, técnicas y avances recientes. Así, enfocarse en la literatura más reciente garantiza una revisión actualizada que refleja los últimos desarrollos en la detección de URLs maliciosas mediante modelos de aprendizaje automático.

4.1.1.2 Variedad de técnicas de Aprendizaje Automático

Se consideraron estudios que abordan una amplia gama de enfoques y técnicas de aprendizaje automático, como redes neuronales, árboles de decisión y regresión, entre otros. Este criterio asegura una revisión exhaustiva y completa, abarcando diversas técnicas y su aplicabilidad en el contexto del estudio. Evaluar una variedad de enfoques enriquece la comprensión del panorama general y permite identificar las ventajas y desventajas de cada técnica, fundamental para una revisión rigurosa.

4.1.1.3 Aplicabilidad a entornos de red

Se incluyeron estudios que aplican técnicas de aprendizaje automático en contextos específicos de redes, como el análisis de tráfico de red, la detección de amenazas en tiempo real y la detección de malware en comunicaciones de red. Para asegurar la relevancia de los resultados obtenidos, se filtraron cuidadosamente los estudios, seleccionando únicamente aquellos que se enfocan en la detección de URLs maliciosas dentro de estos contextos. Este criterio garantiza que los estudios seleccionados sean altamente relevantes y aplicables al objetivo de nuestra investigación, fortaleciendo la validez de los resultados obtenidos.

4.1.1.4 Idioma

Se seleccionaron únicamente estudios escritos en inglés, el idioma predominante en la literatura científica. Esta elección asegura el acceso a una mayor cantidad de evidencia relevante, ya que solo el 6% de los investigadores publican en español. Desde los años 70, el inglés ha prevalecido como el idioma principal de la ciencia, lo que justifica su selección para esta revisión [30].

4.1.2 Fuentes de información

En el presente estudio, se han seleccionado dos fuentes de información reconocidas: *Springer e IEEE Xplore Digital Library*. Estas fuentes fueron elegidas debido a su reputación y especialización en áreas relevantes para la investigación. Springer es conocido por su amplia colección de publicaciones científicas y técnicas, mientras que IEEE Xplore Digital Library es una referencia en ingeniería eléctrica, informática y disciplinas relacionadas.

La utilización de estas bases de datos asegura el acceso a investigaciones actualizadas y de alta calidad, fortaleciendo la validez y relevancia de los resultados obtenidos en este estudio sobre la detección de URLs maliciosas mediante modelos de aprendizaje automático supervisado.

4.2 Implementación del proxy (Objetivo Específico 2)

Para cumplir con el segundo objetivo específico de esta investigación, se integró en un controlador de proxy un detector de URLs maliciosas basado en aprendizaje automático y un clasificador basado en lista negra. Esta etapa del proceso permitió crear un entorno experimental controlado en el que se pudo evaluar y comparar el rendimiento de estas dos técnicas de detección en un escenario real.

4.2.1 Topología de red

En la Figura 1, se presenta la topología de la red utilizada para llevar a cabo este experimento. La red estaba compuesta por dos ubicaciones: una ubicación local y una ubicación en la nube. En ambas ubicaciones, se implementaron servidores de pruebas y se estableció un flujo controlado de solicitudes de DNS para llevar a cabo la evaluación de la latencia del sistema. Esta elección topológica proporcionó el marco necesario para evaluar la latencia del sistema, considerando diferentes ubicaciones y permitiendo evaluar cada

solicitud del cliente en las dos técnicas, garantizando así la robustez y aplicabilidad de los resultados obtenidos.

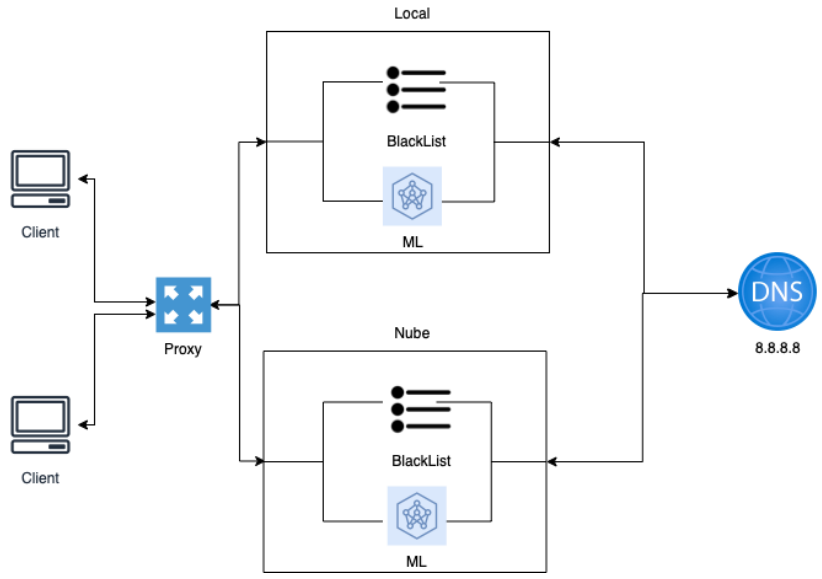

Figura 1: Topología del experimento

Para la configuración del entorno, se utilizaron tanto una máquina virtual (VM) en un entorno local como una instancia de servidor *en Amazon Web Services* (AWS). Ambas configuraciones tenían características similares para asegurar la comparabilidad de las pruebas. En ambas configuraciones (local y nube), se utilizó una VM con un procesador de 1 vCPU, 4 GB de RAM, y hasta 30 GB de almacenamiento EBS de uso general (SSD) y el sistema operativo instalado fue *Ubuntu Minimal* 20.04 LTS.

4.2.2 Desarrollo e implementación del proxy

Se crea un proxy personalizado para recibir todas las solicitudes de URL entrantes. Cada solicitud se enruta a través de este proxy hacia el mecanismo de detección adecuado, ya sea la lista negra o el clasificador de aprendizaje automático. En la configuración local, un servidor procesa las solicitudes desde una red interna, mientras que, en la configuración

en la nube, el proxy maneja las solicitudes en un entorno virtualizado proporcionado por AWS.

El sistema de proxy gestiona y verifica las solicitudes de URL según un procedimiento predeterminado. Inicialmente, los clientes envían consultas de URL al proxy. Cada solicitud se reenvía posteriormente a los mecanismos de detección. Para la detección basada en lista negra, la solicitud se compara con una lista de URLs maliciosas conocidas. Si la URL se encuentra en la lista negra, se marca como maliciosa. Para la detección basada en aprendizaje automático, se evalúa la URL utilizando un modelo entrenado [28] para reconocer patrones peligrosos basados en diferentes atributos de la URL.

4.2.3 Medición de la latencia

Todo el proceso, desde la recepción de la solicitud hasta la entrega de la respuesta, fue cronometrado. Este tiempo incluía el proceso de detección y la resolución DNS para proporcionar una medida precisa de la latencia. El experimento implicó el envío de solicitudes con diferentes cargas al mismo tiempo para evaluar el rendimiento del sistema bajo varios escenarios. Cada escenario de carga fue examinado en configuraciones locales y en la nube, y se calculó la latencia de cada configuración. El objetivo fue comparar el rendimiento del ambiente local y en la nube y evaluar cómo las diferentes cargas afectaban la capacidad de respuesta del sistema.

Para determinar la capacidad del proxy, se empezó a probar diferentes cargas de solicitudes, incrementando de manera progresiva el volumen de solicitudes en ambos entornos. El proceso se inició con un número reducido de solicitudes (10) y se aumentó gradualmente, con un aumento de 100, para evaluar la capacidad del proxy. El objetivo fue

determinar cuántas solicitudes simultáneas podía gestionar antes de que la latencia se viera significativamente afectada o el servidor comprometido.

4.2.4 Clasificador de aprendizaje automático

El clasificador de aprendizaje automático utilizado para detectar URLs maliciosas en este estudio se basó en el modelo "*codebert base Malicious URLs*", desarrollado por DunnBC22 y disponible como código abierto en el repositorio *Hugging Face* [28].

Este modelo utiliza una arquitectura de transformadores diseñada para tareas de procesamiento de lenguaje natural. Procesa las URLs de entrada para predecir si son maliciosas o seguras. Además, se incorporaron métodos como *Random Forest* para hacer el modelo más robusto y preciso.

El clasificador fue entrenado en un conjunto de datos sustancial que incluía más de 650,000 URLs, abarcando URLs benignas, de *defacement*, de *phishing* y de *malware*. Este extenso conjunto de datos cubre varios tipos de URLs maliciosas, mejorando la precisión de la detección. En este trabajo, no se evaluó la precisión de este clasificador, sino que se utilizó como una herramienta para medir el tiempo de clasificación y su impacto en la latencia del sistema.

4.2.2 Envío de solicitudes de prueba

Para evaluar la latencia en ambas ubicaciones, se enviaron solicitudes de prueba a través del proxy. Cada solicitud contenía datos necesarios para determinar si la URL era maliciosa o benigna. Se tuvo en cuenta que las solicitudes podían ser tanto maliciosas como benignas, y en este proceso, se analizaron por separado en cada ubicación.

Este proceso se realizó en dos ubicaciones distintas: una configuración local, que representaba la infraestructura interna de una organización, y una configuración en la nube,

que representaba un entorno virtualizado. En ambas ubicaciones, se enviaron solicitudes a través de un proxy especialmente configurado para este propósito.

Cada solicitud incluyó datos relevantes para determinar si la URL era maliciosa o benigna. Estas solicitudes se generaron para cubrir una amplia variedad de situaciones, incluyendo tanto URLs maliciosas como benignas, y se gestionaron de forma individual tanto en la ubicación local como en la nube.

Además, se aplicaron las solicitudes a los dos enfoques principales: clasificación basada en listas negras y modelos de aprendizaje automático supervisado. Esto permitió una evaluación detallada de cómo la ubicación y la técnica de detección influían en el tiempo de respuesta del sistema. Cada solicitud fue evaluada y sus tiempos de respuesta se registraron meticulosamente para análisis posteriores.

En la fase de recopilación de datos, se utilizó una fuente confiable y en constante actualización; el sitio web urlhaus.abuse.ch [24]. Esta plataforma es una valiosa fuente de URLs maliciosas, ya que se mantiene actualizada de manera constante y alberga una gran cantidad de URLs que incluyen dominios e IPs. Esto permitió tener acceso a una amplia gama de amenazas en línea y garantizó que nuestro conjunto de datos estuviera alineado con la evolución constante de las amenazas en la web.

4.3 Evaluación del impacto en la latencia (Objetivo Específico 3)

Como parte de este proceso, se planteó el tercer objetivo, el cual es medir y cuantificar el impacto de la ubicación y la técnica de detección en la latencia del sistema. Para lograr esto, se diseñaron dos experimentos.

El primer experimento se enfocó en evaluar cómo la ubicación (entorno local o en la nube) y el tipo de técnica de detección (*Machine Learning* o *Blacklist*) influían en el tiempo

de respuesta de nuestro sistema. Se enviaron solicitudes a través del proxy configurado en ambos entornos, local y en la nube, y se midió la latencia de cada solicitud. Este experimento permitió determinar cómo cada configuración afectaba la rapidez con la que el sistema podía procesar y responder a las solicitudes de URL.

El segundo experimento se concentró en evaluar la capacidad de carga del sistema. Se incrementó progresivamente el número de solicitudes simultáneas enviadas al proxy para entender cuántas solicitudes podía manejar el sistema antes de que la latencia se viera afectada de manera significativa. Durante este proceso, se detectó que el proxy mantenía un rendimiento óptimo en términos de latencia hasta aproximadamente 500 solicitudes simultáneas. Más allá de este punto, el sistema comenzó a perder conexiones, indicando un límite en la capacidad de manejo de carga del proxy. Este experimento fue crucial para identificar los puntos de quiebre del sistema y para comprender su comportamiento bajo diferentes niveles de carga.

Estos dos experimentos, en conjunto, proporcionaron una visión integral del rendimiento del sistema en términos de latencia y capacidad de manejo de carga, permitiendo comparar las diferentes configuraciones y técnicas de detección utilizadas.

4.3.1 Selección de Técnicas de Detección y ubicación

En esta etapa, se evaluó la efectividad de dos técnicas de detección de URL ampliamente utilizadas: *Machine Learning* y *Blacklist* de tamaño fijo. Las solicitudes enviadas en las fases anteriores fueron sometidas a ambas técnicas en ambas ubicaciones (entorno local y en la nube), y se registraron los tiempos requeridos para determinar su carácter malicioso o benigno.

Cada solicitud fue evaluada individualmente en ambas ubicaciones, y se documentaron los tiempos de respuesta correspondientes. Este procedimiento permitió la comparación de los tiempos de respuesta en función de la técnica utilizada y la ubicación de ejecución. Al comparar los tiempos de respuesta, se pudo determinar cómo la latencia varía según la técnica de detección empleada (*Machine Learning* o *Blacklist*) y la ubicación del sistema (local o en la nube).

Esta comparación fue fundamental para identificar la técnica de detección más eficiente y cómo la ubicación influye en el rendimiento del sistema. Al documentar meticulosamente los tiempos de respuesta, se obtuvo una comprensión detallada del impacto de cada variable en el desempeño del sistema, lo que permitió hacer recomendaciones informadas para la implementación de sistemas de detección de URL maliciosas en diferentes entornos.

En este análisis, el modelo de ML, muestra una precisión del 72.7%. En comparación, la técnica de lista negra presenta una precisión del 74.2%, lo que sugiere que, aunque el modelo de ML tiene un rendimiento aceptable, la técnica de lista negra es más efectiva para identificar correctamente las URLs maliciosas. Además, las listas negras se actualizan constantemente, incorporando nuevas URLs maliciosas de manera regular.

Es importante destacar que las dos técnicas no son del todo comparables, ya que el modelo de ML fue entrenado de una manera cuya metodología exacta se desconoce, tratándose como una "caja negra". Esta falta de transparencia en el proceso de entrenamiento implica que no se pueden analizar fácilmente sus características. A pesar de las ventajas de flexibilidad y adaptabilidad que ofrece el modelo de ML, su rendimiento depende de actualizaciones regulares. Por otro lado, las listas negras, con su actualización constante,

garantizan un rendimiento estable al basarse en URLs previamente identificadas. Para optimizar la detección, podría ser beneficioso combinar ambas técnicas.

4.3.3 Selección de carga

El propósito de este proceso es evaluar la capacidad de carga del proxy utilizando diferentes técnicas en el sistema de detección de URLs maliciosas en tiempo real.

Durante el proceso de prueba, se detectó que el proxy perdía conexiones tanto en el entorno local como en la nube después de un número específico de solicitudes simultáneas, indicando un límite en la capacidad de manejo de carga del proxy. Específicamente, se observó que el proxy mantenía un rendimiento aceptable en términos de latencia, pero comenzaba a perder conexiones después de aproximadamente 500 solicitudes simultáneas, lo que señalaba un problema de capacidad de manejo de conexiones en lugar de un problema de latencia.

Este proceso de selección de carga es importante para comprender los límites y la capacidad efectiva del sistema de detección en situaciones de alta demanda, además, contribuiyó de manera significativa a la respuesta de nuestra tercera pregunta de investigación "¿Cuál es la capacidad de carga efectiva del proxy implementado en el sistema de detección de URLs maliciosas en tiempo real?".

La información recopilada se utiliza para entender los límites y la capacidad efectiva del sistema de detección en situaciones de alta demanda. Además, se realiza un análisis comparativo entre los resultados obtenidos en la ubicación local y en la nube, así como entre las técnicas de detección basadas en listas negras y aprendizaje automático.

Dado el análisis de prueba realizado, se determinó que la capacidad máxima que el proxy soporta es aproximadamente 500 solicitudes simultáneas. Por lo tanto, se decidió

utilizar tres niveles de carga para los experimentos: un nivel bajo, un nivel medio y un nivel alto. Esto permitió evaluar de manera más precisa el comportamiento del sistema bajo diferentes condiciones de carga y entender mejor sus límites y rendimiento en escenarios de baja, media y alta demanda.

En la siguiente sección, se explicará a detalle el diseño experimentar utilizado.

4.4 Diseño experimental

En esta sección, describiremos el diseño experimental que será implementado para responder a las preguntas de investigación planteadas.

4.4.1 Hipótesis Experimental

Hipótesis General:

- **H0 (Hipótesis Nula):** No hay diferencias significativas en la latencia del sistema de detección de URLs maliciosas en tiempo real entre las diferentes combinaciones de ubicación (local y nube), técnica de detección (lista negra y aprendizaje automático) y carga de solicitudes (10, 100, 500).
- **H1 (Hipótesis Alternativa):** Existen diferencias significativas en la latencia del sistema de detección de URLs maliciosas en tiempo real entre al menos una de las combinaciones de ubicación (local y nube), técnica de detección (lista negra y aprendizaje automático) y carga de solicitudes (10, 100, 500).

Hipótesis para la Ubicación:

- **H0 (Hipótesis Nula):** $\mu_{local} = \mu_{nube}$

  No hay diferencias significativas en la latencia del sistema entre la ubicación local y en la nube.

- **H1 (Hipótesis Alternativa):** $\mu_{local} \neq \mu_{nube}$

Existen diferencias significativas en la latencia del sistema entre la ubicación local y en la nube.

Hipótesis para la Técnica de Detección:

- **H0 (Hipótesis Nula):** $\mu_{lista\_negral} = \mu_{aprendizaje\_automático}$

  No hay diferencias significativas en la latencia del sistema entre la clasificación basada en listas negras y el aprendizaje automático supervisado.

- **H1 (Hipótesis Alternativa):** $\mu_{lista\_negral} \neq \mu_{aprendizaje\_automático}$

  Existen diferencias significativas en la latencia del sistema entre la clasificación basada en listas negras y el aprendizaje automático supervisado.

Hipótesis para la Carga de Solicitudes:

- **H0 (Hipótesis Nula):** $\mu_{carga\_10} = \mu_{carga\_100} = \mu_{carga\_500}$

  No hay diferencias significativas en la latencia del sistema entre diferentes cargas de solicitudes (10, 100, 500)..

- **H1 (Hipótesis Alternativa):** $\mu_{carga\_10} \neq \mu_{carga\_100} \neq \mu_{carga\_500}$

  Existen diferencias significativas en la latencia del sistema entre diferentes cargas de solicitudes (10, 100, 500).

4.4.2 Variables de Respuesta

La variable de respuesta en este experimento es la "latencia del sistema". Medida en segundos (s), esta variable refleja el tiempo necesario desde que se recibe una solicitud DNS hasta que se verifica la legitimidad de una URL en el controlador DNS.

4.4.3 Factores

1. **Ubicación**: La ubicación se refiere al entorno en el cual se ejecuta el sistema de detección. En este experimento, se distinguen dos ubicaciones principales:
    - local
    - nube
2. **Técnica de Detección**: La técnica utilizada para detectar URLs maliciosas es un factor crítico. En este experimento, se comparan dos técnicas principales:
    - la clasificación basada en listas negras
    - el aprendizaje automático supervisado
3. **Carga de Solicitudes**: La carga se refiere al número de solicitudes simultáneas que el sistema de detección debe manejar. Se realizarán pruebas con diferentes cargas por segundo para evaluar cómo responde el sistema a diferentes niveles de demanda.
    - 10
    - 100
    - 500

4.4.4 Modelo estadístico

Para asegurar la validez y rigurosidad de los análisis estadísticos realizados en este estudio, un grupo de estudiantes de estadística se encargó de toda la preparación, análisis y demás tareas relacionadas con el tratamiento de datos. Este grupo de estudiantes estaba realizando un proyecto final de un curso, donde debían poner a prueba los conocimientos adquiridos a lo largo de su formación académica. Los estudiantes estuvieron guiados por un profesor experto en la materia estadística, asegurando así la calidad y precisión del análisis.

Se empleó un modelo mixto de efectos para analizar los datos de latencia, tomando en cuenta los efectos de la técnica de detección, el entorno y la carga de solicitudes. Se utilizó

un Análisis de Varianza (ANOVA) para evaluar las diferencias significativas entre los factores considerados. La variable respuesta fue analizada en centisegundos para facilitar la interpretación de los resultados.

Para asegurar la validez de los resultados, los estudiantes realizan una prueba estadística. Utilizando un modelo lineal de efectos mixtos para analizar los datos de latencia, considerando los efectos de interacción entre el método de detección, el entorno y la carga de solicitudes.

El modelo es el siguiente:

$$\mu_{ijk} = \mu + \alpha_i + \beta_j + \delta_k + (\alpha\beta)_{ij} + v_l \qquad (1)$$

Donde:

- μ es la media general
- α_i es el efecto del i-ésimo nivel del factor de método de detección.
- β_j es el efecto del j-ésimo nivel del factor de ubicación.
- δ_k es el efecto del k-ésimo nivel del factor de carga de solicitudes.
- $(\alpha\beta)_{ij}$ es el efecto de interacción entre método de detección y ubicación
- v_l es el efecto aleatorio del lote.

Los estudiantes de estadística se encargaron de la preparación y limpieza de los datos, eliminando cualquier parcela que pudiera causar desbalance en el análisis. Además, se aseguraron de ajustar el modelo para cumplir con los supuestos de normalidad y homocedasticidad.

Se realizaron varias sesiones informativas con el autor, Diego Orozco, para comprender a fondo cómo se recolectaron los datos y asegurar que el análisis estadístico reflejara adecuadamente las condiciones experimentales. Los estudiantes realizaron

análisis descriptivos y gráficos preliminares para explorar la distribución de los datos y las posibles interacciones entre los factores.

Para una explicación detallada del reporte generado por los estudiantes de estadística, se puede revisar el Anexo 1. Este anexo proporciona una explicación de los procedimientos estadísticos llevados a cabo.

CAPÍTULO 5: RESULTADOS

A continuación, se presentan los resultados obtenidos para cada uno de los objetivos de esta investigación, iniciando por la revisión de literatura, seguido de la implementación del proxy y su evaluación.

5.1. Revisión de literatura

En esta sección, se detallan los resultados de la actualización de la revisión de literatura. Se identificaron las técnicas comunes utilizadas en la detección de URLs maliciosas, y se desarrolló una propuesta de taxonomía.

5.2.1 Propuesta de Taxonomía

Basado en nuestro artículo "*Taxonomy of Malicious URL Detection Techniques*" [4], se presenta una taxonomía que clasifica las características de las URLs más utilizadas en diferentes tipos de aprendizaje automático. Esta taxonomía se estructura en torno a dos enfoques principales: Detección Basada en Aprendizaje Automático y Detección Basada en Reglas (ver Figura 2). Estos dos enfoques cubren una amplia gama de estrategias utilizadas para identificar URLs maliciosas en entornos en línea, además facilita la comprensión y comparación de las distintas técnicas de detección y sus enfoques en relación con las propiedades de las URLs.

Por un lado, la detección basada en reglas se fundamenta en el uso de heurísticas y patrones predefinidos para clasificar una URL. Este enfoque consiste en un análisis heurístico que examina características como la longitud de la URL, la similitud con dominios legítimos, la presencia de palabras clave sospechosas, las redirecciones, el descubrimiento de dominios emergentes y la ubicación geográfica del servidor. También se utiliza la coincidencia de

patrones, donde se buscan firmas y expresiones regulares que coincidan con URLs maliciosas conocidas.

Por otro lado, las técnicas de detección basadas en aprendizaje automático analizan características específicas de las URLs para determinar si son maliciosas o benignas. Los clasificadores consideran características como la longitud de la URL, el dominio y subdominio, las palabras clave y el seguimiento de redirecciones, entre otros. Además, se exploran técnicas avanzadas como el aprendizaje profundo y métodos ensamblados como *Random Forests y AdaBoost* para mejorar la precisión de la detección.

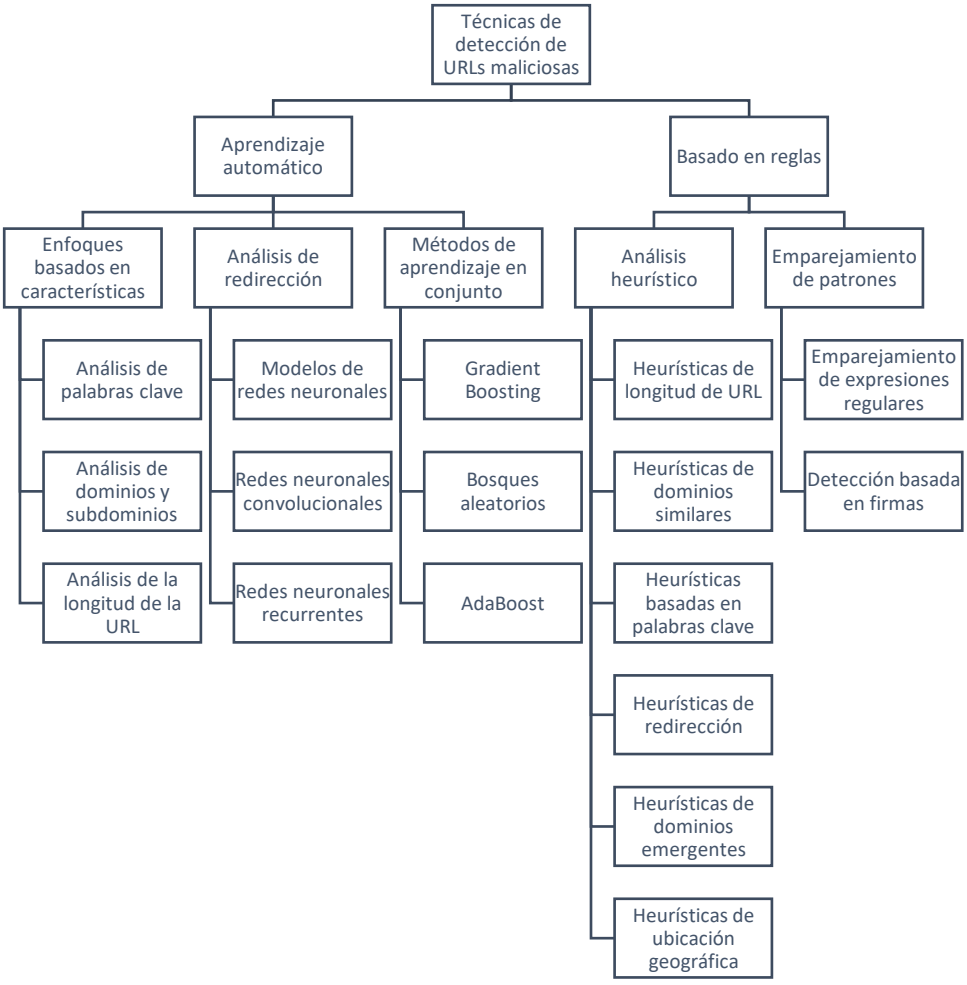

Figura 2: Técnicas de detección de URLs maliciosas

5.1.2 Comparación de técnicas de detección

La comparación entre las técnicas de detección basadas en aprendizaje automático y las basadas en reglas revela diferencias significativas en términos de efectividad, facilidad de implementación y capacidad de adaptación a nuevas amenazas. La Tabla I presenta un resumen de estos enfoques, destacando sus usos comunes, ventajas y desventajas.

Tabla I: Comparación de Técnicas de Detección de URLs Maliciosas

Técnica de Detección	Usos Comunes	Ventajas	Desventajas
Detección Basada en Reglas	Filtrado inicial, detección de patrones conocidos	Simple de implementar, rápido en ejecución	Difícil de actualizar, menos efectiva contra nuevas amenazas
Detección Basada en Aprendizaje Automático	Detección avanzada, adaptación a nuevas amenazas	Alta precisión, capacidad de aprendizaje y adaptación	Requiere más recursos computacionales, complejidad en el entrenamiento del modelo

La comparación muestra que, aunque las técnicas basadas en reglas son más fáciles de implementar y rápidas en su ejecución, las técnicas basadas en aprendizaje automático ofrecen una mayor precisión y capacidad de adaptación a nuevas amenazas. La elección de la técnica de detección depende del contexto específico y de los requisitos del sistema, como la velocidad de detección, la capacidad de adaptación y los recursos disponibles.

5.2 Impacto en la latencia con la implementación del proxy

En esta sección, se presentan los resultados de la evaluación del impacto en la latencia del sistema de detección de URLs maliciosas, considerando la ubicación, la técnica de detección y la cantidad de solicitudes simultáneas. Los resultados de esta investigación han sido enviados a la conferencia Latincom 2024, en Medellín, Colombia, para su evaluación y publicación (ver Anexo 4).

5.2.1 Latencia en Diferentes Ubicaciones

Se evaluó el rendimiento del proxy en dos configuraciones principales: local y en la nube. La configuración local se realizó en un servidor interno que representaba la infraestructura interna de una organización, mientras que la configuración en la nube se llevó a cabo en *Amazon Web Services* (AWS).

Los resultados mostraron diferencias significativas en la latencia entre las dos configuraciones. La configuración local presentó una latencia más alta, especialmente al utilizar técnicas basadas en aprendizaje automático. Esto se debe a que los recursos disponibles en un entorno local son limitados y no pueden escalarse fácilmente para manejar cargas pesadas de procesamiento. En contraste, las soluciones en la nube aprovecharon recursos escalables, lo que redujo la latencia general. Esta diferencia fue más pronunciada con la detección basada en aprendizaje automático, demostrando la ventaja de la nube al manejar tareas computacionalmente intensivas.

La Figura 3 muestra el promedio de latencia por método en ambas configuraciones. Se observa que la latencia en la nube es considerablemente menor en comparación con la configuración local, tanto para el método basado en listas negras como para el método basado en aprendizaje automático. Esto resalta las ventajas de utilizar una infraestructura en la nube para sistemas que requieren alta capacidad de procesamiento y bajas latencias.

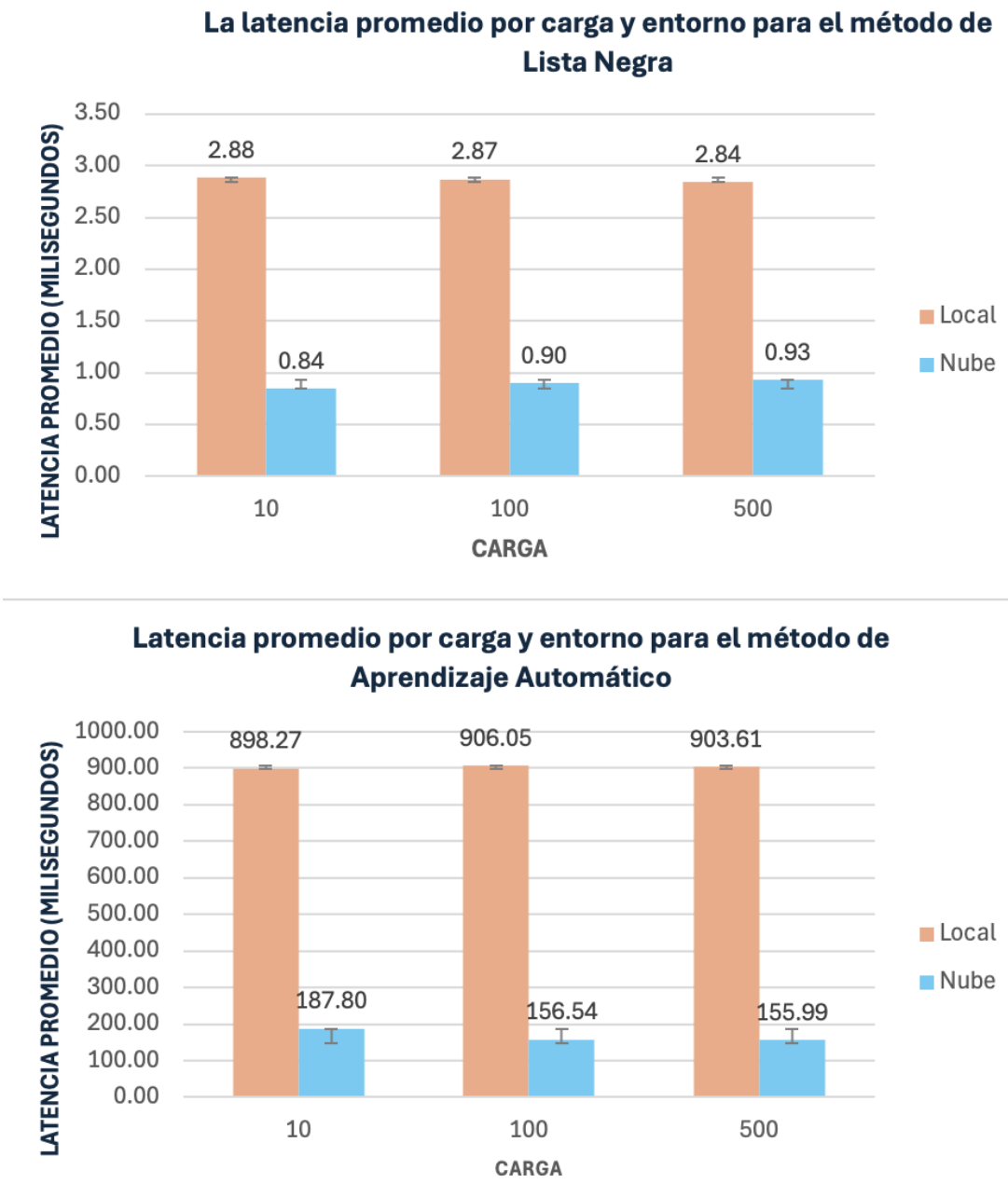

Figura 3: Promedio de latencia por método

5.2.2 Latencia según la técnica de detección

El método de listas negras demostró una menor latencia en comparación con el aprendizaje automático en ambos entornos, local y en la nube, como se muestra en la Figura 3. Estos resultados indican que el método de listas negras mantiene una latencia baja en diferentes niveles de carga, con un rendimiento ligeramente mejor en el entorno de la nube

que en el entorno local. Las diferencias en la latencia son mínimas, lo que hace que el método de listas negras sea adecuado para la detección en tiempo real con un rendimiento consistente.

Por otro lado, el método de aprendizaje automático mostró una latencia más alta, especialmente en el entorno local. Esta mayor latencia sugiere que la detección basada en aprendizaje automático es más intensiva en recursos, lo que lleva a tiempos de procesamiento más largos. En el entorno de la nube, aunque la latencia es menor en comparación con el entorno local, sigue siendo significativamente mayor que la del método de listas negras.

Los resultados específicos de la latencia para cada método y entorno se pueden observar en la Figura 3. Para la técnica de listas negras, la latencia en el entorno local varió entre 2.84 y 2.88 milisegundos, mientras que en la nube osciló entre 0.84 y 0.93 milisegundos. Esto demuestra que la infraestructura en la nube reduce la latencia significativamente al manejar este tipo de detección.

Para la técnica de aprendizaje automático, la diferencia es aún más pronunciada. La latencia en el entorno local varió entre 898.27 y 906.05 milisegundos, mientras que en la nube estuvo entre 155.99 y 187.80 milisegundos. La capacidad de la nube para manejar cargas intensivas de procesamiento de manera más eficiente es evidente, lo que la convierte en la opción preferida para aplicaciones que requieren la detección de URLs maliciosas en tiempo real.

Estas observaciones resaltan la importancia de seleccionar la infraestructura adecuada según los requisitos de latencia y carga del sistema. La nube ofrece ventajas claras en términos de reducción de latencia y manejo de procesamiento intensivo, lo que la hace ideal para aplicaciones críticas donde el tiempo de respuesta es esencial.

5.2.3 Tasa de fallo

La Figura 4 muestra la tasa de fallos tanto en los entornos locales como en la nube a medida que aumenta la carga. Comprender la confiabilidad y el rendimiento del sistema en situaciones reales requiere analizar su resiliencia bajo una demanda creciente.

Ambos entornos manejaron las conexiones sin problemas significativos hasta una carga de 500 solicitudes por segundo, lo que sugiere un rendimiento consistente. Sin embargo, cuando la carga superó este punto, el número de fallos aumentó, pero de diferentes maneras para cada entorno. Esta variación indica que, si bien ambas configuraciones pueden gestionar cargas moderadas de manera efectiva, las cargas más altas exponen diferencias en su capacidad para mantener conexiones estables.

A medida que la carga aumentaba en la configuración local, el número de fallos aumentaba progresivamente. Esto implica que cuando la máquina local recibe demasiadas solicitudes, comienza a tener dificultades. Esto podría ser el resultado de recursos insuficientes, lo que hace que pierda conexiones ya que no puede manejar efectivamente el mayor número de hilos. Sin embargo, los fallos en la configuración en la nube también aumentaron, aunque de manera más gradual y constante.

El gráfico se extiende hasta una carga de 3000 solicitudes para proporcionar una vista clara y continua de cómo aumentan los fallos en ambos entornos. Muestra que, más allá de las 500 solicitudes por segundo, el proxy ya no es capaz de atender satisfactoriamente las solicitudes recibidas, especialmente en el entorno local. Esto indica que, bajo una alta demanda, la configuración local podría necesitar optimización o escalabilidad para manejar mejor la carga sin perder muchas conexiones.

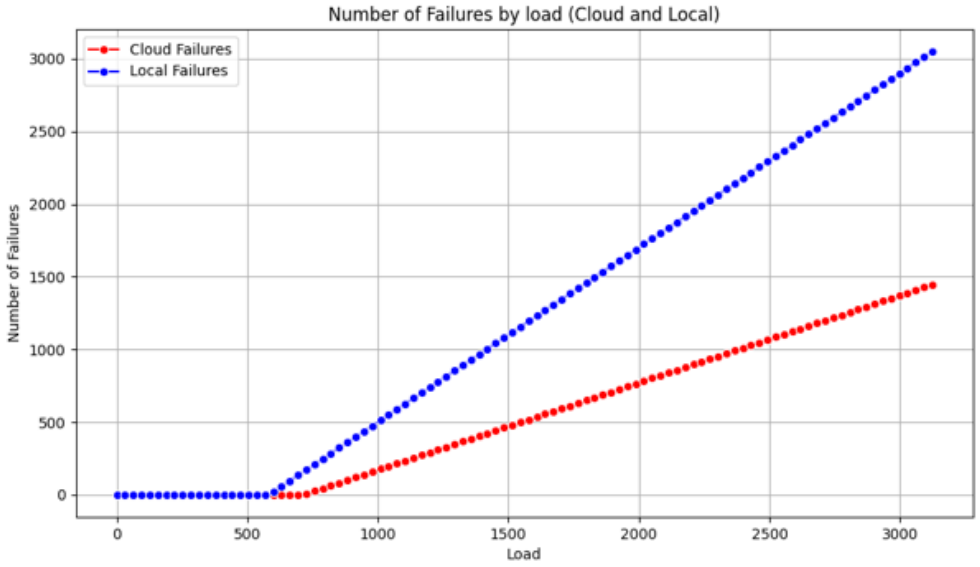

Figura 4: *Promedio de latencia por método*

La Figura 4 muestra claramente esta tendencia, donde el entorno local experimenta un aumento más pronunciado en los fallos a medida que la carga se incrementa, en comparación con el entorno en la nube. Esta observación es importante para determinar la viabilidad de la implementación de sistemas de detección de URLs maliciosas en tiempo real en diferentes entornos.

Estos resultados subrayan la importancia de elegir el entorno adecuado según los requisitos de rendimiento y confiabilidad del sistema. La configuración en la nube ofrece ventajas claras en términos de escalabilidad y estabilidad bajo cargas elevadas, mientras que la configuración local podría enfrentar desafíos significativos en escenarios de alta demanda.

La Tabla II muestra el tiempo promedio tanto para los entornos locales como para los de la nube. El tiempo base representa el tiempo de ida y vuelta para realizar una solicitud sin ningún tipo de verificación. Este tiempo base es importante para comprender la sobrecarga adicional introducida por los métodos de detección. Estos tiempos base indican la latencia mínima involucrada en una solicitud de ida y vuelta simple, sin ningún procesamiento

adicional. La mayor parte del tiempo total se gasta en el proceso de verificación, que es donde entran en juego los métodos de detección.

Tabla II: Tiempo promedio en milisegundos

Entorno	Local (ms)	Nube (ms)
Tiempo Base	1.3	0.1
Método de Detección	453.3	78.7

Los experimentos confirmaron que la elección del método de detección y del entorno impacta significativamente en la latencia del sistema. La detección basada en listas negras superó consistentemente al aprendizaje automático en términos de velocidad, aunque el aprendizaje automático ofrece una precisión de detección potencialmente más alta. El entorno en la nube proporcionó un rendimiento superior, particularmente para manejar altas cargas y tareas intensivas en cómputo como la detección basada en aprendizaje automático.

Además, estos resultados y hallazgos fueron compilados en un artículo que fue enviado a Latincom 2024, Medellín, Colombia, para su revisión y posible publicación. Esto subraya la relevancia y el impacto de descubrimientos en la comunidad académica y profesional del área de redes.

5.4 Modelo estadístico

El análisis estadístico de los datos de latencia se llevó a cabo utilizando un modelo mixto de efectos para garantizar la validez de los resultados. Este modelo permitió considerar las interacciones entre el método de detección, el entorno y la carga de solicitudes, proporcionando una comprensión detallada de cómo estos factores influyen en la latencia del sistema de detección de URLs maliciosas.

El gráfico de normalidad Q-Q (Fig. 5) muestra cómo los residuos de nuestros datos se comparan con una distribución normal perfecta. La mayoría de los puntos están cerca de la línea diagonal, lo que significa que nuestros residuos generalmente siguen un patrón normal. Sin embargo, en los extremos del gráfico, algunos puntos se desvían de la línea, lo que indica la presencia de algunos valores atípicos o puntos de datos con valores más extremos de lo esperado. Estas observaciones sugieren que, aunque nuestros datos en su mayoría se ajustan a la distribución normal, hay algunas anomalías que podrían necesitar una inspección más detallada.

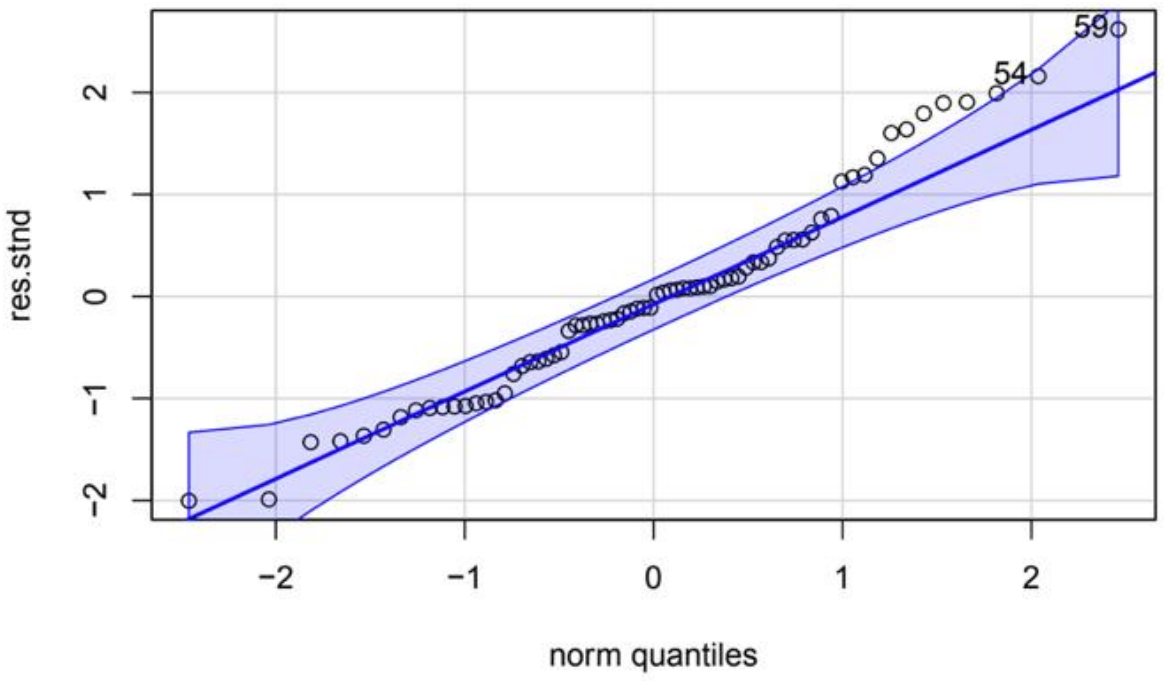

Figura 5: Normalidad de los residuos

Para asegurar la validez de los resultados, se utilizó un modelo mixto de efectos para analizar los datos de latencia, considerando las interacciones entre el método de detección, el entorno y la carga de solicitudes.

En el proceso de análisis, se realizaron varias pruebas de hipótesis. La hipótesis de interacción triple, que verificaba si los tres factores (método de detección, ubicación y carga de solicitudes) eran independientes, no fue rechazada, indicando que no hay interacción triple. De manera similar, las hipótesis de interacción entre la carga de solicitudes y el método de detección, así como entre la carga de solicitudes y la ubicación, tampoco fueron rechazadas, sugiriendo que estos factores son independientes.

Sin embargo, la hipótesis de interacción entre el método de detección y la ubicación sí fue rechazada, indicando que estos factores no son independientes y deben ser considerados conjuntamente. Esto significa que la combinación de método de detección y ubicación tiene un efecto significativo en la latencia del sistema.

Además, se verificó el efecto de la carga de solicitudes de manera independiente y se encontró que no había un efecto significativo en la latencia. Esto implica que la cantidad de solicitudes simultáneas no afecta considerablemente el tiempo de respuesta del sistema.

En cuanto a la comparación de los métodos de detección en diferentes ubicaciones, se descubrió que la latencia del método de aprendizaje automático era significativamente mayor que la del método de listas negras, tanto en la nube como en el entorno local. Específicamente, la diferencia en la latencia fue de 15.53 centisegundos en la nube y 88.57 centisegundos en el entorno local, siendo siempre mayor la latencia del método de aprendizaje automático.

Estos resultados indican que la elección del método de detección y la ubicación afectan significativamente la latencia del sistema. En particular, el método de listas negras mostró una menor latencia en comparación con el método de aprendizaje automático, siendo esta diferencia más pronunciada en el entorno local.

Para obtener más detalles sobre el trabajo realizado por los estudiantes de estadística, consulte el Anexo 1, donde se encuentra el documento final producto de este análisis.

5.5 Discusión

Los resultados del experimento demuestran diferencias claras en el rendimiento entre los entornos en la nube y local, especialmente en términos de tasas de fallo bajo diferentes cargas. Hasta una carga de 500 solicitudes, ambos entornos manejaron las conexiones sin problemas significativos, sugiriendo que el sistema funciona de manera confiable dentro de este rango.

Sin embargo, a medida que la carga aumenta más allá de este punto, el número de fallos comienza a incrementarse, con el entorno local experimentando un aumento más pronunciado. Esto indica que la configuración local empieza a tener dificultades bajo cargas más altas, probablemente debido a limitaciones de recursos. Por otro lado, el entorno en la nube, aunque también experimenta un aumento en los fallos, lo hace de manera más gradual, mostrando los beneficios de recursos escalables y optimizados.

Estos hallazgos son importantes para decidir dónde implementar sistemas de detección de URLs maliciosas en tiempo real. El aumento pronunciado de fallos en la configuración local bajo cargas altas sugiere que los sistemas locales pueden necesitar una mejor gestión de recursos o escalado para manejar la demanda aumentada. La configuración en la nube, con su aumento más gradual en los fallos, parece más adecuada para escenarios de alta carga debido a su capacidad de escalabilidad.

Un punto clave es que, aunque la latencia no se ve significativamente afectada por la técnica de detección utilizada, ya sea aprendizaje automático o listas negras, la pérdida de conexiones se convierte en un problema mayor a medida que aumenta la carga. Tanto las

técnicas de aprendizaje automático como las de listas negras muestran tiempos de latencia bajos y muy similares, haciendo que la elección de la técnica sea insignificante para el usuario en términos de velocidad. Sin embargo, la estabilidad de las conexiones se ve muy afectada, y esto se convierte en un problema significativo con cargas más altas. Esto significa que mantener la estabilidad de las conexiones es más importante para el uso práctico que solo enfocarse en la latencia.

Para los usuarios o sistemas finales, el impacto de estos hallazgos es importante. La pregunta principal es si se puede utilizar el proxy sin causar problemas notables para los usuarios. El aumento en los fallos de conexión, especialmente en configuraciones locales bajo alta carga, podría llevar a interrupciones. Para los sistemas que necesitan un rendimiento de red consistente y confiable, la estabilidad de las conexiones es un factor crítico. Si el proxy causa demasiados fallos de conexión, podría llevar a una inestabilidad del sistema, afectando la experiencia del usuario.

La usabilidad de introducir este proxy depende de su impacto en el rendimiento general del sistema. Si los beneficios, como la detección efectiva de URLs maliciosas, superan las desventajas, como el aumento de fallos de conexión, el proxy podría ser valioso. Sin embargo, si el impacto en la estabilidad de las conexiones es demasiado significativo, se necesitarían más optimizaciones y pruebas para asegurar que pueda manejar cargas más altas sin causar interrupciones mayores.

En términos de latencia, según Hamidouche, esta juega un papel en la efectividad de tales sistemas. Ellos mencionan que una latencia de procesamiento de menos de 1 milisegundo (ms) es un aspecto importante para la detección y respuesta oportuna a amenazas [27]. En el experimento realizado, el tiempo base para el entorno local es de 1.39 ms,

ligeramente por encima del umbral recomendado, mientras que el entorno en la nube tiene un tiempo base de 0.12 ms, dentro del rango aceptable (Tabla II).

En cuanto al método de detección, el entorno local tiene una latencia promedio de 453.34 ms, y el entorno en la nube tiene una latencia promedio de 78.76 ms. En la evaluación, se logró observar lo siguiente:

- **Entorno Local:** Añade aproximadamente 451.95 ms.
- **Entorno en la Nube:** Añade aproximadamente 78.64 ms.

Estos resultados enfatizan que la mayor parte de la latencia se debe al proceso de verificación. Por lo tanto, aunque el proxy muestra promesa en términos de capacidades de detección, su rendimiento actual en la configuración local puede llevar a demoras notables y potencialmente afectar la experiencia del usuario. Se requiere una mayor optimización para cumplir con los requisitos de latencia para la detección de amenazas en tiempo real.

En última instancia, la decisión de desplegar este proxy en un entorno real dependerá de los requisitos específicos y las limitaciones del sistema. Si la baja latencia y la estabilidad de las conexiones son cruciales, especialmente bajo altas cargas, una solución basada en la nube con detección por listas negras podría ser más adecuada. Para escenarios donde la precisión de la detección es importante y se puede tolerar cierto nivel de latencia, un enfoque de aprendizaje automático basado en la nube podría ser una opción viable.

En la literatura relacionada, se ha notado una falta de atención hacia las limitaciones del uso de modelos de aprendizaje automático para la detección de URLs maliciosas. Es importante reconocer que, con el tiempo, estos modelos pueden desactualizarse debido a la evolución constante de las amenazas cibernéticas. Por ello, es fundamental establecer un tiempo de actualización periódico para garantizar que el modelo se mantenga eficaz y

relevante en su capacidad de detección. Esta necesidad de actualización conlleva tiempo y recursos, lo que debe ser considerado en el contexto de su implementación en entornos de producción.

Dado que el sistema está diseñado para operar en un entorno de producción, es viable llevar a cabo estas actualizaciones en un entorno de pruebas (DEV). De esta manera, el modelo puede ser entrenado y validado antes de su despliegue. Una vez que se complete la actualización y el nuevo modelo esté listo, se puede realizar la transición sin afectar la técnica de detección en uso. Esta estrategia no solo asegura la integridad y efectividad del sistema, sino que también minimiza el impacto en los usuarios finales, permitiendo una actualización fluida y continua de las capacidades de detección del sistema.

Según estos resultados, es posible integrar la detección de URLs maliciosas en tiempo real en la infraestructura de DNS; sin embargo, la elección del entorno de implementación y del método de detección impacta significativamente en el rendimiento. Una solución basada en la nube con detección de listas negras podría ser mejor para aplicaciones que necesitan alto rendimiento y baja latencia. No obstante, el aprendizaje automático basado en la nube puede ser una buena opción en situaciones donde la precisión de la detección es importante y se puede tolerar un cierto nivel de latencia.

CAPÍTULO 6: CONCLUSIONES

El objetivo principal de este trabajo fue evaluar cómo la ubicación, el tipo de técnica usada y la carga recibida pueden afectar el tiempo de respuesta al integrar un clasificador de URLs en tiempo real en un sistema de DNS. Para lograr esto, se plantearon tres objetivos específicos:

1. Identificar las técnicas comunes utilizadas en la detección de URLs maliciosas a través de una revisión literaria.
2. Desarrollar un proxy capaz de ejecutarse en diversas ubicaciones y permitir la integración de distintas técnicas de clasificación de URLs.
3. Evaluar el impacto en la latencia del sistema de detección de URLs maliciosas considerando la ubicación, la técnica de detección y la cantidad de solicitudes simultáneas.

A lo largo del estudio, se llevaron a cabo varios experimentos que permitieron analizar estas variables y obtener resultados significativos.

La comparación entre las técnicas de detección basadas en listas negras y las basadas en aprendizaje automático reveló diferencias importantes. La detección basada en listas negras mostró consistentemente una menor latencia en comparación con el aprendizaje automático en ambos entornos, local y en la nube. Esta diferencia es más pronunciada en el entorno local, donde la latencia del aprendizaje automático es significativamente mayor.

Estos resultados indican que, si la baja latencia es un requisito relevante, especialmente bajo altas cargas, una solución basada en listas negras es más adecuada. Por otro lado, si la

precisión de la detección es una prioridad y se puede tolerar un nivel de latencia más alto, el aprendizaje automático basado en la nube es una opción viable.

Los experimentos también demostraron que la ubicación del sistema de detección tiene un impacto significativo en la latencia. El entorno en la nube presentó una latencia considerablemente menor en comparación con el entorno local, tanto para las listas negras como para el aprendizaje automático. Esto se debe a la capacidad de la nube para escalar recursos de manera más eficiente, manejando mejor las cargas intensivas de procesamiento.

En términos de capacidad de carga, ambos entornos manejaron las conexiones sin problemas significativos hasta una carga de 500 solicitudes aproximadamente. Sin embargo, más allá de este punto, el número de fallos aumentó, especialmente en el entorno local. Esto sugiere que, bajo una alta demanda, la configuración local podría necesitar optimización o escalabilidad para manejar mejor la carga sin perder muchas conexiones.

La pérdida de paquetes con más de 500 solicitudes, puede afectar el promedio de todos los tiempos de respuesta, ya que algunos paquetes pueden regresar con éxito mientras que otros presentan una latencia "infinita", distorsionando así la medición general del rendimiento del sistema.

Para la implementación de un sistema de detección de URLs maliciosas en tiempo real, las siguientes recomendaciones son importantes tomar en consideración:

1. **Entorno de Despliegue**: Considerar la implementación en la nube para aplicaciones que requieren alta capacidad de procesamiento y bajas latencias. La nube ofrece ventajas claras en términos de escalabilidad y estabilidad bajo cargas elevadas.

2. **Método de Detección**: Utilizar listas negras si la baja latencia es crucial. En casos donde la precisión es más importante y se puede tolerar una mayor latencia, optar por el aprendizaje automático.
3. **Optimización del Sistema**: En entornos locales, es fundamental optimizar y posiblemente escalar la infraestructura para manejar cargas más altas sin comprometer la estabilidad del sistema.

Este estudio abre varias líneas de investigación futuras. Sería valioso explorar enfoques híbridos que combinen listas negras y aprendizaje automático para aprovechar las ventajas de ambos métodos. Además, investigar técnicas de optimización específicas para entornos locales podría mejorar la capacidad de manejo de carga y reducir la latencia.

En conclusión, la investigación demuestra que es posible integrar la detección de URLs maliciosas en tiempo real en la infraestructura de DNS. Sin embargo, la elección del entorno de implementación y del método de detección tiene un impacto significativo en el rendimiento. La nube con detección basada en listas negras es más adecuada para aplicaciones que requieren alta eficiencia y baja latencia, mientras que el aprendizaje automático basado en la nube es ideal para situaciones donde la precisión de la detección es fundamental.

El problema principal identificado no estuvo en la capacidad de recibir solicitudes, sino en la cantidad de pérdidas de paquetes que estas solicitudes experimentaban. A medida que la carga aumentaba, la pérdida de paquetes se volvía más significativa, afectando la estabilidad y confiabilidad del sistema.

En futuros trabajos, nos gustaría mejorar nuestro proxy para que pueda manejar un mayor número de solicitudes sin perderlas. Como punto de referencia, el DNS público de Google permite hasta 1500 solicitudes por segundo. Si podemos mejorar nuestro proxy para igualar esta capacidad, mejoraría significativamente su fiabilidad y robustez, especialmente bajo alta demanda. Esto haría que nuestra solución sea mucho más efectiva para la detección de URLs maliciosas en tiempo real.

Finalmente, también nos gustaría explorar otros lugares para introducir la detección durante el ciclo de vida de cada solicitud. Una posibilidad es trabajar en un servidor DNS completamente funcional en lugar de un proxy.

REFERENCIAS

[1] Dsouza, R. (2020). *Malicious URL (s) classification* (Doctoral dissertation, Dublin, National College of Ireland)

[2] Cersosimo Morales, M. M. Desarrollo de un modelo de detección de URLs maliciosas usando aprendizaje automático supervisado.

[3] Kumar, J., Santhanavijayan, A., Janet, B., Rajendran, B., & Bindhumadhava, B. S. (2020, January). Phishing website classification and detection using machine learning. In *2020 international conference on computer communication and informatics (ICCCI)* (pp. 1-6). IEEE.

[4] Orozco, D, Lara, A, Marin, G. Taxonomy of Malicious URL Detection Techniques. 2023

[5] Liu, C., & Albitz, P. (2006). *DNS and Bind*. " O'Reilly Media, Inc

[6] Alani, M. M., & Tawfik, H. (2022). PhishNot: a cloud-based machine-learning approach to phishing URL detection. *Computer Networks*, *218*, 109407

[7] Cheng, D., Liu, Z., Zhang, P., Zeng, Y., Cui, J., & Kong, L. (2018, May). Profiling Malicious Domain by Multidimensional Features. In *2018 International Conference on Robots & Intelligent System (ICRIS)* (pp. 489-495). IEEE.

[8] M. C. Morales, Desarrollo de un modelo de derección de URLs maliciosas usando aprendizaje automático supervisado. Trabajo final de investigación aplicada., Programa de Posgrado en Computación e Informática. Universidad de Costa Rica, 2023.

[9] Do Xuan, C., Nguyen, H. D., & Tisenko, V. N. (2020). Malicious URL detection based on machine learning. *International Journal of Advanced Computer Science and Applications*, *11*(1).

[10] Joshi, A., Lloyd, L., Westin, P., & Seethapathy, S. (2019). Using lexical features for malicious URL detection--a machine learning approach. arXiv preprint arXiv:1910.06277.

[11] Sahoo, D., Liu, C., & Hoi, S. C. (2017). Malicious URL detection using machine learning: A survey. *arXiv preprint arXiv:1701.07179*.

[12] Gburi, Raed Hameed. (2021). Detection of malicious URLs using machine learning. (Yayınlanmamış yüksek lisans tezi). Altınbaş Üniversitesi, Lisansüstü Eğitim Enstitüsü, İstanbul.

[13] Zhauniarovich, Y., Khalil, I., Yu, T., & Dacier, M. (2018). A survey on malicious domains detection through DNS data analysis. *ACM Computing Surveys (CSUR)*, *51*(4), 1-36.

[14] Do Xuan, C., Nguyen, H. D., & Tisenko, V. N. (2020). Malicious URL detection based on machine learning. *International Journal of Advanced Computer Science and Applications*, *11*(1).

[15] A. D. Gabriel, D. T. Gavrilut, B. I. Alexandru and P. A. Stefan, "Detecting Malicious URLs: A Semi-Supervised Machine Learning System Approach," 2016 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Timisoara, Romania, 2016, pp. 233-239, doi: 10.1109/SYNASC.2016.045.

[16] Janet, B., & Kumar, R. J. A. (2021, March). Malicious URL detection: a comparative study. In *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)* (pp. 1147-1151). IEEE.

[17] Khormali, A., Park, J., Alasmary, H., Anwar, A., Saad, M., & Mohaisen, D. (2021). Domain name system security and privacy: A contemporary survey. *Computer Networks*, *185*, 107699.

[18] F. Zou, S. Zhang, B. Pei, L. Pan, L. Li and J. Li, "Survey on Domain Name System Security," 2016 IEEE First International Conference on Data Science in Cyberspace (DSC), Changsha, China, 2016, pp. 602-607, doi: 10.1109/DSC.2016.96.

[19] Antonakakis, M., Perdisci, R., Dagon, D., Lee, W., & Feamster, N. (2010). Building a dynamic reputation system for {DNS}. In *19th USENIX Security Symposium (USENIX Security 10)*.

[20] Aljabri, M et al.. Detecting malicious URLs using machine learning techniques: review and research directions. IEEE Access. 2022.

[21] Madhubala, R., et al. "Survey on Malicious URL Detection Techniques." IEEE International Conference on Trends in Electronics and Informatics (ICOEI), 2022.

[22] Huang, X., Susilo, W., Mu, Y., & Wu, W. (2006). Proxy signature without random oracles. In Mobile Ad-hoc and Sensor Networks: Second International Conference, MSN 2006, Hong Kong, China, December 13-15, 2006. Proceedings 2 (pp. 473-484). Springer Berlin Heidelberg.

[23] Abiona, O., Oluwaranti, A., Oluwatope, A., Bello, S., Onime, C., Sanni, M., & Kehinde, L. (2014). Proxy server experiment and network security with changing nature of the web. International Journal of Communications, Network and System Sciences, 7(12), 519.

[24] URLhaus, "URLhaus - Malicious URL Sharing," [Online]. Available: https://urlhaus.abuse.ch.

[25] G. Cena, S. Scanzio and A. Valenzano, "Design guidelines to improve reliability of seamless redundancy in Wi-Fi networks," *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, Berlin, Germany, 2016, pp. 1-8, doi: 10.1109/ETFA.2016.7733500.

[27] Hamidouche, M., Demissie, B. F., & Cherif, B. (2024). Real-time Threat Detection Strategies for Resource-constrained Devices. arXiv preprint arXiv:2403.15078

[28] DunnBC22. (2023). Codebert base Malicious URLs. Hugging Face. Retrieved from https://huggingface.co/DunnBC22/codebert-base-Malicious_URLs/tree/main

[29] Aljabri, M., Altamimi, H. S., Albelali, S. A., Al-Harbi, M., Alhuraib, H. T., Alotaibi, N. K., ... & Salah, K. (2022). Detecting malicious URLs using machine learning techniques: review and research directions. IEEE Access, 10, 121395-121417.

[30] Torres, Á. (2017). La importancia de los idiomas de las publicaciones científicas. Revista Comunicar, Escuela para autores. https://doi. org/10.3916/escuela-de-autores-036.

Anexos

Anexo 1: Reporte final modelo estadístico

Latencia en sistemas de detección de URLs maliciosas en tiempo real.

Daniel Sibaja[1]; José Pablo Mora[1]; Pamela Argueta[1]
daniel.sibajasalazar@ucr.ac.cr, jose.mora20@ucr.ac.cr,
pamela.argueta@ucr.ac.cr

RESUMEN

En los tiempos actuales, la ciberseguridad y la velocidad en la navegación por la web, se han vuelto cruciales para la protección contra la URLs maliciosas y comodidad del usuario. Este estudio investigó métodos para detectar URLs maliciosas en sistemas de nombres de dominio (DNS) y su implementación en tiempo real, centrándose en la latencia del sistema. Se compararon dos enfoques: listas negras (BL) y modelos de aprendizaje automático (ML), evaluando su rendimiento en ubicaciones locales y en la nube bajo diferentes volúmenes de solicitudes (10, 100 y 500 solicitudes). Los resultados indicaron que los métodos de listas negras locales superaron significativamente a los modelos de aprendizaje automático en términos de latencia, lo que se tradujo en una identificación más rápida de URLs maliciosas. Esta ventaja se mantuvo incluso con grandes volúmenes de tráfico, demostrando la capacidad de escalabilidad de los métodos BL para entornos de alto tráfico. Además, se encontró que la ubicación influye en la eficiencia del método de detección, con listas negras mostrando un mejor rendimiento tanto en la nube como en local. Sin embargo, el aprendizaje automático ofrece ventajas importantes, como la capacidad de adaptarse a nuevos datos y detectar patrones complejos en URLs maliciosas. Aunque los métodos BL tienen menor latencia, los modelos ML pueden ser más adecuados para detectar amenazas emergentes, lo cual es crucial para la protección continua contra nuevas formas de ataques cibernéticos. La potencia del estudio se calculó para asegurar la validez de las conclusiones, encontrando una alta probabilidad de detectar diferencias significativas en la latencia entre los métodos BL y ML. Esto proporciona una alta confianza en la validez de los resultados. En conclusión, mientras que los métodos de listas negras son superiores en términos de latencia, los modelos de aprendizaje automático son valiosos por su adaptabilidad y capacidad de detección precisa, lo que los hace ideales para entornos donde la detección de nuevas amenazas es prioritaria.

PALABRAS CLAVE: local, nube, aprendizaje automático, listas negras, amenazas cibernéticas.

INTRODUCCIÓN

[1] Estudiante de Bachillerato en estadística de la Universidad de Costa Rica

Esta investigación examinó métodos para detectar URLs maliciosas en sistemas de nombres de dominio (DNS) y su implementación en tiempo real para prevenir el acceso a contenido dañino, un problema creciente en la era digital (Dsouza, 2020; Orozco et al., 2023). El DNS es un componente crítico de Internet, encargado de traducir nombres de dominio legibles para humanos a direcciones IP legibles por computadoras, una función esencial para el correcto funcionamiento de servicios en línea como acceso a sitios web, correo electrónico y más (Liu & Albitz, 2006). Sin embargo, la mayoría de las redes tradicionales no clasifican ni filtran las URLs, lo que aumenta significativamente la posibilidad de ataques y amenazas maliciosas en línea (Kumar et al., 2020).

Para abordar esta problemática, se han desarrollado diversos clasificadores de URLs basados en técnicas de aprendizaje automático supervisado, machine learning, que permiten identificar patrones y características asociadas a URLs maliciosas (Alani & Tawfik, 2022; Do Xuan et al., 2020). No obstante, implementar estas técnicas de detección en tiempo real dentro del flujo de trabajo del DNS presenta desafíos adicionales, como el impacto en la latencia del sistema, definida como el tiempo que tarda la información en viajar de un punto a otro en la red (Zhauniarovich et al., 2018). Mantener una baja latencia es un factor crítico para garantizar una buena experiencia del usuario y un alto rendimiento de las aplicaciones en línea (Joshi et al., 2019).

En este contexto, un proxy actuó como intermediario entre los usuarios y los servidores web, recibiendo las solicitudes de los usuarios y reenviándolas a los servidores de destino, y trayendo las respuestas de vuelta (Huang et al., 2006). En el presente estudio, el proxy fue configurado para revisar si las páginas web solicitadas eran seguras o potencialmente maliciosas. El objetivo principal fue investigar la viabilidad de la detección de URLs maliciosas en un entorno de tiempo real y cómo diferentes enfoques, tanto basados en listas negras como en modelos de aprendizaje automático, podrían afectar la latencia del sistema (Cersosimo Morales, 2023).

Diego Orozco Fonseca, en su trabajo final de graduación, llevó a cabo este experimento, aportando su experiencia como experto en la implementación y evaluación de sistemas de detección de URLs maliciosas. Para lograr esto, implementó un proxy en el experimento capaz de recibir y redirigir solicitudes en un entorno de producción real, tanto en una ubicación local como en la nube, utilizando clasificación basada en listas negras y modelos de aprendizaje automático supervisado (figura 1). Además, se realizaron pruebas bajo diferentes cargas de solicitudes para determinar la capacidad y resistencia del proxy ante distintos volúmenes de tráfico. Si bien existen enfoques asincrónicos relacionados con la detección de URLs maliciosas en la literatura (Gabriel et al., 2016), esta investigación representó un avance al poner a prueba estos métodos en un escenario de tiempo real. Los resultados obtenidos permitieron comprender el impacto real de cada enfoque en la latencia del sistema, proporcionando información valiosa para el diseño e implementación de sistemas de detección de URLs maliciosas integrados en el DNS (Aljabri et al., 2022). El objetivo específico establecido fue encontrar el enfoque de detección que produjera la menor latencia posible, maximizando así la velocidad de respuesta.

METODOLOGÍA

El experimento se centró en evaluar la latencia en el sistema de detección de URLs maliciosas en tiempo real, considerando factores como la técnica de detección, la cual es el factor de diseño del experimento, ubicación, y capacidad de carga. Se creó una red para realizar el experimento (Figura 1). Este entorno experimental controlado permitió evaluar y comparar los resultados obtenidos con el uso de ambos métodos de detección.

La red estuvo compuesta por dos ubicaciones: local y nube, y dos métodos de detección: basados en listas negras (BL) y modelos de aprendizaje automático (ML), para determinar si las URLs eran maliciosas o benignas. También se tomó en cuenta la cantidad de solicitudes utilizadas; se utilizaron tres tipos de paquetes de solicitudes: 10, 100 y 500 solicitudes. Además, se consideraron los efectos de interacción que se mencionan en el modelo siguiente, que representa nuestro modelo base, para entender cómo la combinación de estos factores influye en el desempeño. La variable dependiente de este estudio fue el tiempo de respuesta medido en centisegundos.

A partir de estos factores del experimento, se definieron los siguientes tratamientos:

- Blacklist-Local
- Machine Learning-Local
- Blacklist-Nube
- Machine Learning-Nube

Por lo tanto, se definieron tres parcelas en función de la cantidad de solicitudes (10, 100, 500), a las cuales se les aplicaron los cuatro tratamientos mencionados. Cada bloque fue creado de forma aleatoria a partir de una población de aproximadamente 300,000 URLs, incluyendo URLs malignas y no malignas. Una vez creados, cada bloque pasó de forma aleatoria por cada uno de los tratamientos, hasta obtener las latencias de cada bloque con los cuatro tratamientos aleatorizados.

Con base en esto, se determinó que el experimento era de parcelas divididas, ya que se esperaba un efecto de la cantidad de solicitudes. Por ello, el modelo que mejor se adaptó a nuestras necesidades era un modelo mixto, ya que permite examinar los efectos de los múltiples factores y sus interacciones en la latencia final de la solicitud, lo cual es crucial para entender cómo estos factores pueden influir en la eficiencia de la detección de URLs maliciosas.

Para el experimento se utilizó el siguiente modelo:

$$\mu_{ijk} = \mu + \alpha_i + \beta_j + \delta_k + (\alpha\beta)_{ij} + (\alpha\delta)_{ik} + (\beta\delta)_{jk} + (\alpha\beta\delta)_{ijk} + \nu_l$$

Donde,
μ: Es la media general, es decir, sin tomar en cuenta ningún tratamiento.
α_i: es el efecto del nivel i del Factor Método de detección.
β_j: es el efecto del nivel j del Factor Ubicación.
δ_k: es el efecto del nivel k del Factor Solicitudes.

$(\alpha\beta)_{ij}$: es el efecto de la interacción entre Método de detección y Ubicación.

$(\alpha\delta)_{ik}$: es el efecto de la interacción entre Método de detección y Solicitudes.

$(\beta\delta)_{jk}$: es el efecto de la interacción entre Ubicación y Solicitudes.

$(\alpha\beta\delta)_{ijk}$: es el efecto de la interacción entre los tres factores: método, ubicación y solicitudes.

v_l: Es el efecto de la parcela. Es decir, el efecto que cada bloque de 10,100 y 500 URLs trae sobre el modelo.

También se verificó, de forma gráfica y con pruebas formales, si las interacciones triples y dobles tenían un efecto significativo sobre la variable respuesta. Otros supuestos que fueron de interés incluyeron el supuesto de normalidad, verificado con una prueba Kolmogorov-Smirnov, y el de homocedasticidad, verificado con una prueba de Levene.

Todos los análisis fueron realizados utilizando R, un lenguaje de programación especializado en estadística (Mendoza Vega, 2018), en su versión 4.3.3 (R Core Team, 2023), a través de la interfaz RStudio (2022). Asimismo, se emplearon los paquetes lattice (Sarkar D, 2008), car (Fox, J., y Weisberg, S., 2019), readxl (Wickham y Bryan, 2022), y otros paquetes más.

RESULTADOS

En el análisis inicial, de manera descriptiva, se observó un desbalance en los tratamientos. Para mitigar este problema, se decidió promediar las respuestas de cada subparcela. Esto significó que, dentro de cada bloque de 10, 100 y 500 URLs, se tomó el promedio por tratamiento, resultando en 4 respuestas por bloque (figura 2), a pesar de esta medida es importante mencionar que los datos continúan teniendo desbalance.

Análisis descriptivo:

Tabla 1

Promedio por tratamiento

	Nube 10	**Local 10**	**Nube 100**	**Local 100**	**Nube 500**	**Local 500**
Black List	0.09	0.27	0.09	0.29	0.09	0.29
Machine Learning	15.97	89.34	15.62	88.79	15.62	89.88

Desde un punto de vista descriptivo, se observó una diferencia muy marcada entre Black List y Machine Learning para cada nivel de ubicación y cantidad de solicitudes. También se observó una diferencia entre los diferentes niveles de ubicación, pero no se detectó una diferencia significativa para cada cantidad de solicitudes. Fue necesario realizar pruebas adicionales para determinar la significancia de cada factor y sus interacciones.

Pruebas formales

Se realizaron diferentes pruebas de hipótesis para determinar la independencia de los factores y su significancia. Al verificar la independencia de los tres factores entre sí (interacción triple), no se rechazó la hipótesis nula, por lo que no se tomó en cuenta. Seguidamente, se verificaron tres hipótesis de interacción doble, encontrando que solo una era significativa: la interacción entre Ubicación y Método de detección (figura 3). Por esta razón, estos factores no pudieron ser analizados independientemente.

Además, se verificó la significancia de la capacidad de carga, concluyendo que no tenía un efecto significativo sobre la latencia. Asimismo, se verificó si las diferencias entre Black List y Machine Learning para cada ubicación eran significativas, concluyendo que en ambos casos existían diferencias significativamente distintas de 0 con una confianza global del 95%.

Tabla 2

Intervalos de confianza según ubicación

Ubicación	Límite Inferior	Límite Superior
Local	88.41	88.73
Nube	15.49	15.59

Se obtuvieron límites de confianza para las diferencias entre Black List y Machine Learning para cada nivel del factor "ubicación". Con una confianza global del 95%, se concluyó que la diferencia promedio entre Machine Learning y Black List en la latencia estaba entre 88.41 y 88.73 centisegundos cuando se utilizó una ubicación "Local", y entre 15.49 y 15.59 centisegundos cuando se utilizó una ubicación en la "Nube".

Dado que no existe una diferencia relevante conocida y hay poca investigación relacionada con la latencia en la detección de URLs malignas utilizando métodos de detección (específicamente Black List y Machine Learning), este experimento concluyó que si existe una diferencia significativa en la latencia promedio entre Black List y Machine Learning, para cada diferente ubicación, sin embargo no sé concluye si estas diferencias son lo suficientemente elevadas como para ser consideradas relevantes.

Una vez verificadas las hipótesis, se ajustó el modelo en base a los resultados obtenidos. El modelo ajustado es el siguiente:

$$\mu_{ijk} = \mu + \alpha_i + \beta_j + \delta_k + (\alpha\beta)_{ij} + \nu_l$$

Donde,

μ: Es la media general, es decir, sin tomar en cuenta ningún tratamiento.

α_i: es el efecto del nivel i del Factor Método de detección.

β_j: es el efecto del nivel j del Factor Ubicación.

δ_k: es el efecto del nivel k del Factor Solicitudes.

$(\alpha\beta)_{ij}$: es el efecto de la interacción entre Método de detección y Ubicación.

ν_l: Es el efecto de la parcela. Es decir, el efecto que cada bloque de 10,100,500 trae sobre el modelo.

Este ajuste aseguró que se consideraron y evaluaron adecuadamente todas las interacciones y efectos relevantes para el estudio. También fue necesario calcular la potencia del estudio. La potencia permite determinar la probabilidad de detectar diferencias cuando realmente existen. Dado que el experto no proporcionó una diferencia relevante y no se encontró información relevante en las investigaciones académicas relacionadas, se utilizaron dos diferencias relevantes: 7 y 0.20 centisegundos (que corresponden a la mediana y el primer cuartil de las observaciones de la variable respuesta, respectivamente) para medir la potencia de la prueba. Se encontró que hay una probabilidad de 1 de concluir que hay diferencias entre Machine Learning y Blacklist cuando en realidad las medias de latencia difieren al menos en 7 o 0.20 centisegundos. La potencia obtenida fue alta, lo que indica que hay una alta confianza en la validez de las conclusiones.

CONCLUSIONES

En la actual era digital, la protección contra las amenazas cibernéticas es vital para organizaciones de todos los sectores. La detección rápida y precisa de URLs maliciosas es crucial para mantener la seguridad de los sistemas de información y la integridad de los datos.

Los resultados del estudio mostraron que los métodos tradicionales de listas negras locales (BL) superan significativamente a los modelos de aprendizaje automático (ML) en términos de latencia, definida como el tiempo necesario para procesar una solicitud y generar una respuesta. Esta menor latencia se traduce en una identificación más rápida de URLs maliciosas, lo cual es esencial para proteger a usuarios y sistemas en tiempo real. Además, la ventaja de los métodos de BL en términos de latencia se mantiene incluso cuando aumenta el número de solicitudes. Esto implica que estos métodos pueden gestionar eficazmente grandes volúmenes de tráfico sin comprometer la eficacia de la detección. Esta capacidad de escalabilidad los hace especialmente adecuados para entornos de alto tráfico y dinámicos, como plataformas de comercio electrónico y sitios web de gran escala.

Otro aspecto destacado por la investigación es la importancia de la ubicación en la eficiencia del método de detección. Los métodos de listas negras demostraron un rendimiento superior en comparación con los modelos de ML tanto en la nube como en local. Esto sugiere que, aunque la infraestructura y los recursos computacionales de la nube pueden contribuir significativamente, los métodos de listas negras siguen siendo más eficientes en términos de latencia. Sin embargo, es importante reconocer que el aprendizaje automático ofrece ventajas significativas en otros aspectos. Según Alani y Tawfik (2022), una de las ventajas de los modelos de ML es que pueden adaptarse y aprender de nuevos datos, permitiendo una detección más precisa y efectiva de amenazas emergentes que no estarían en una lista negra predefinida. Do Xuan et al. (2020) también resaltan que el ML puede identificar patrones complejos y características sutiles en URLs maliciosas que podrían ser pasadas por alto por los métodos tradicionales.

En términos de experiencia del usuario, el libro "Usability Engineering" de Nielsen menciona que 0.1 segundos es el límite en el que un usuario siente que un sistema está funcionando eficazmente y que 10 segundos es el tiempo límite para mantener la atención del mismo (como se citó en Gutiérrez, 2021). Considerando que la latencia promedio en la detección de URLs malignas usando ML es de aproxidamente 0.15 y 0.89 segundos (en los niveles de Local y Nube respectivamente), se pueden utilizar las ventajas del ML sin perjudicar gravemente la experiencia del usuario en la web. Además, el experto resaltó que, dado que el método de ML tiene capacidad de decisión y adaptación, la diferencia en milisegundos puede ser irrelevante en casos donde se debe priorizar la detección sobre el tiempo de espera.

Por lo tanto, aunque los métodos de listas negras son superiores en términos de latencia, los modelos de ML pueden ser más adecuados en situaciones donde la adaptabilidad y la capacidad de detección de amenazas emergentes son más críticas que la velocidad de respuesta inmediata. Esta adaptabilidad puede ser crucial para proteger contra nuevas formas de ataques cibernéticos y para mejorar continuamente la seguridad de los sistemas a medida que evolucionan las amenazas.

REFERENCIAS BIBLIOGRÁFICAS

Alani, M. M., & Tawfik, H. (2022). PhishNot: A cloud-based machine-learning approach to phishing URL detection. *Computer Networks, 218*, 109407. https://doi.org/10.1016/j.comnet.2022.109407

Aljabri, M., Alhurayyis, I., Alsadhan, A., & Alsalman, H. (2022). Detecting malicious URLs using machine learning techniques: Review and research directions. *IEEE Access, 10*, 24587-24602. https://doi.org/10.1109/ACCESS.2022.3154350

Cersosimo Morales, M. M. (2023). *Desarrollo de un modelo de detección de URLs maliciosas usando aprendizaje automático supervisado* [Trabajo final de investigación aplicada]. Universidad de Costa Rica.

Do Xuan, C., Nguyen, H. D., & Tisenko, V. N. (2020). Malicious URL detection based on machine learning. *International Journal of Advanced Computer Science and Applications, 11*(1). https://doi.org/10.14569/IJACSA.2020.0110177

Dsouza, R. (2020). *Malicious URL(s) classification* [Doctoral dissertation, National College of Ireland]. https://norma.ncirl.ie/4567/

Gabriel, A. D., Gavrilut, D. T., Alexandru, B. I., & Stefan, P. A. (2016). Detecting malicious URLs: A semi-supervised machine learning system approach. En *2016 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)* (pp. 233-239). IEEE. https://doi.org/10.1109/SYNASC.2016.045

Gutiérrez, N. (2021). *Malicious websites blocking system using Deep Learning algorithms* [Proyecto de tesis, Universidad Politécnica de Cataluña]. telecosBCN.

https://upcommons.upc.edu/bitstream/handle/2117/357059/Degree_Thesis_Norma_Gutierrez_Final.pdf?sequence=2&isAllowed=y

Huang, X., Susilo, W., Mu, Y., & Wu, W. (2006). Proxy signature without random oracles. En *Mobile Ad-hoc and Sensor Networks: Second International Conference, MSN 2006, Hong Kong, China, December 13-15, 2006. Proceedings 2* (pp. 473-484). Springer. https://doi.org/10.1007/11943952_38

Joshi, A., Lloyd, L., Westin, P., & Seethapathy, S. (2019). Using lexical features for malicious URL detection–A machine learning approach. *arXiv*. https://arxiv.org/abs/1910.06277

Kumar, J., Santhanavijayan, A., Janet, B., Rajendran, B., & Bindhumadhava, B. S. (2020). Phishing website classification and detection using machine learning. En *2020 International Conference on Computer Communication and Informatics (ICCCI)* (pp. 1-6). IEEE. https://doi.org/10.1109/ICCCI48352.2020.9104127

Liu, C., & Albitz, P. (2006). *DNS and BIND*. O'Reilly Media.

Mendoza Vega, J. B. (2018). *R para principiantes*. https://bookdown.org/jboscomendoza/r-principiantes4/introduccion-que-es-r-y-para-que-es-usado.html

Orozco, D., Lara, A., & Marin, G. (2023). *Taxonomy of malicious URL detection techniques*.

Orozco Fonseca, D. (2023). Figura 1: Topología del experimento. En *Propuesta de Trabajo Final de Investigación Aplicada para optar al grado y título de Maestría Profesional en Ciencias de la Computación e Informática*. Universidad de Costa Rica.

R Core Team. (2023). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

RStudio Team. (2022). *RStudio: Integrated Development For R*. RStudio. http://www.rstudio.com/

Wickham, H., & Bryan, J. (2022). *readxl: Read Excel Files* (versión 1.4.1). https://cran.r-project.org/web/packages/readxl/index.htm

Zhauniarovich, Y., Khalil, I., Yu, T., & Dacier, M. (2018). A survey on malicious domains detection through DNS data analysis. *ACM Computing Surveys, 51*(4), 1-36. https://doi.org/10.1145/3191685

Anexo 2: ICITs 2024

Taxonomy of Malicious URL Detection Techniques

Diego Orozco Fonseca, Gabriela Marín and Adrian Lara

[1] Universidad de Costa Rica, San José, Costa Rica, diego.orozco@ucr.ac.cr
[2]

Abstract. Malicious URLs are often used by phishing campaigns, botnets and other attacks. Indeed, DNS traffic is necessary for the Internet to function correctly, which means that this data flow cannot be blocked. For these reasons, detecting malicious URLs is both important, challenging and still an open research problem. There are two types of techniques used to detect malicious URLs: rules-based and machine learning-based. The traditional, rules-based techniques rely on blacklists and heuristics. These techniques struggle to keep up with a rapidly changing array of malicious URLs. Therefore, machine learning-based techniques have emerged. Both detection techniques rely on URL characteristics such as length, number of vowels and others to classify them as legitimate or malicious. The main contribution of this paper is to propose a taxonomy of detection techniques and to point out which URL characteristics are used by each method. While surveys on the topic exist, a precise mapping between the detection methods and the characteristics is not available. We also compare these techniques, highlighting that machine learning-based techniques are more complex to implement but better at keeping up with rapidly incoming new malicious URLs. In contrast, rules-based techniques are simpler and easier to implement, but they struggle to update fast enough to identify new malicious URLs.

Keywords: Malicious URLs, Machine learning, Blacklist-based classification, URL classification.

1. Introduction

In the field of cybersecurity, the detection of malicious Uniform Resource Locators (URLs) remains a constant challenge due to the increase in online threats and increasingly sophisticated tactics used by malicious actors. Accurate identification of malicious URLs is essential to protect users and safeguard the integrity of computer systems. However, this task is far from simple, as continuous technological advances give cybercriminals the ability to devise new methods to deceive and compromise online security. This issue is further complicated by the fact that malicious URLs are a common component in a wide variety of cyber-attacks, including phishing campaigns and botnet operations. These attacks leverage malicious URLs in their strategies, making them highly elusive and difficult to counter. According to Sahoo et al. [2], approximately one-third of all websites have potentially malicious nature. Attempting to block these URLs directly presents a complex dilemma, as doing so could inadvertently disrupt legitimate online services. Indeed, blocking the DNS port is not a viable solution as it would negatively impact the functioning of the Internet. Additionally, users are prone to falling victim to common attacks like phishing when interacting with malicious URLs, underscoring the importance of addressing this problem.

The only effective way to detect malicious URLs is to perform a direct analysis of the URL in question and evaluate its characteristics. Malicious URLs are designed to appear legitimate, making them difficult to detect. Therefore, effective detection of malicious URLs requires a thorough analysis of URLs and their attributes. This process involves considering both the static and dynamic characteristics of an URL. Static features include URL length,

domain structure and path, and other attributes that do not change over time. On the other hand, dynamic features are related to URL behavior and may vary over time, such as the analysis of web traffic generated by the URL [3].

To address this challenge, several approaches have been proposed, including the use of machine learning models and heuristics. Machine learning models are trained using data sets that contain examples of both malicious and benign URLs. On the other hand, heuristic techniques are based on specific rules and guidelines to evaluate the authenticity of an URL. These rules may involve reviewing features such as URL length, domain structure, presence of specific keywords, etc.

This study aims to address the following research questions:

1. What are the most used characteristics to identify malicious URLs?
2. Which URL characteristics are more common in rules-based techniques?
3. Which URL characteristics are more common in machine learning-based techniques?

The main contribution of this paper is to identify which URL characteristics are most used by each type of detection technique. We begin by explaining the characteristics of URLs that are commonly used to identify malicious URLs, including static and dynamic attributes as described above. After that, we describe machine learning-based and rules-based methods that can be used to implement new URL classifiers. While surveys on URL classification techniques exist [1, 2, 7], they focus primarily on machine-learning techniques, or they are not complete. In this work, we do not attempt to provide a systematic literature review. Instead, we only cite recent papers to update the state of the art and we focus on clearly explaining the characteristics of URLs and the taxonomy of detection techniques.

The rest of the paper is structured as follows. In Section 2, we describe characteristics that can be used to differentiate malicious and legitimate URLs. In Section 3 we first introduce the taxonomy of detection techniques (rules-based and machine learning-based) and then we explain and compare both techniques. We draw our conclusions in Section 4.

2. Characteristics of URLs

Processing an URL is a fundamental step in detecting malicious URLs, as it allows you to break it down into its key components for detailed analysis. Key features include domain and route, which play an important role in risk assessment and identification of known malicious patterns.

URL features are divided into two main categories: static and dynamic (see Fig 1). Static features are attributes that can be obtained without interacting directly with the URL. These features include the length of different sections of the URL, the presence of file extensions, the number of digits, vowels and consonants, as well as the detection of IP addresses in the URL. These lexical features have been used in recent research to characterize malware URLs and discern malicious trends. Metrics such as entropy and consecutive counting of specific characters have also been used for this purpose. On the other hand, dynamic features require deeper interaction with the URL and, although they can achieve high accuracy in detecting malicious URLs, their collection is often time- and resource-intensive, which makes them less suitable for real-time analysis. This highlights the importance of static features, which offer an effective and quick way to classify an URL [1].

In the area of malicious domain detection, some methods rely entirely on the use of lexical features. These approaches focus on attributes such as the proportion of significant

characters, n-gram scores, length of subdomain names, and the number of alphanumeric character exchanges. In addition, they use techniques such as TF-IDF (Term Frequency-Inverse Document Frequency) to extract relevant terms and apply dimensional reduction methods, such as principal component analysis (PCA), to obtain representative vectors [2].
Next, we propose a taxonomy of detection methods that rely on the features described above. In this taxonomy, we explore the different ways these methods work with the characteristics of URLs.

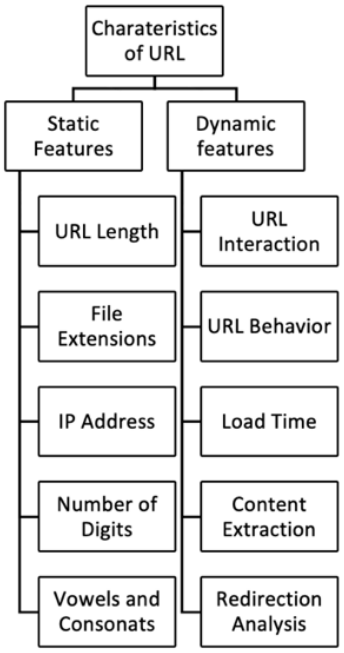

***Fig*. 1.** Characteristics of the URLs

3. Taxonomy of detection techniques

The proposed taxonomy for malicious URL detection techniques is structured around two main approaches: Machine Learning-Based Detection and Rules-Based Detection. These two approaches cover a wide range of strategies used to identify malicious URLs in online environments (see Fig. 2).

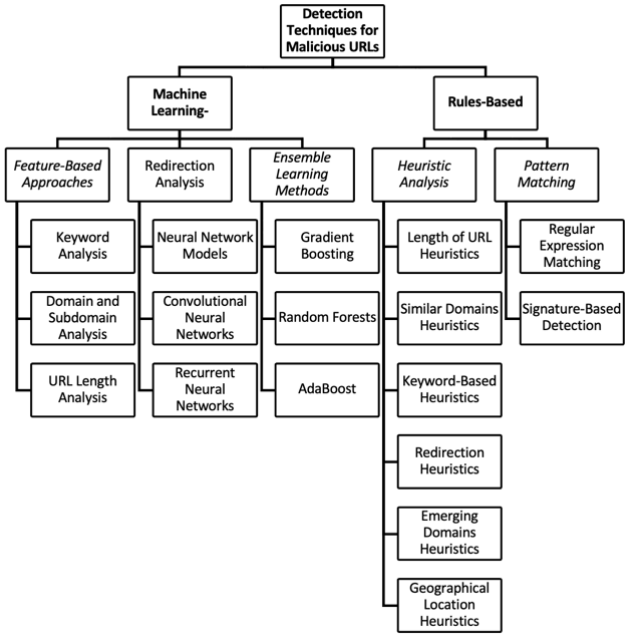

***Fig*. 2.** Detection Techniques for Malicious URLs

On the one hand, rule-based detection is based on the use of predefined heuristics and patterns to classify an URL. This approach consists of a heuristic analysis that examines features such as URL length, similarity to legitimate domains, presence of suspicious keywords, redirections, discovery of emerging domains, and server geographic location. Pattern matching is also used, where signatures and regular expressions matching known malicious URLs are searched. We describe these techniques in Section 3.1

On the other hand, machine learning-based detection techniques analyze specific features of URLs and determine whether they are malicious or benign. Classifiers consider features such as URL length, domain and subdomain, keyword, and redirect tracking, among others. In addition, advanced techniques such as deep learning and assembly methods such as Random Forests and AdaBoost are explored to improve detection accuracy. We describe these techniques in Section 3.2.

3.1 Rules-based detection techniques

Previous studies have examined and evaluated rule-based techniques, such as blacklists and whitelists, in terms of their effectiveness in detecting malicious URLs. However, these approaches can have limitations, especially regarding the latency they generate. Rule-based processing may require significant time to query a list of known URLs and make decisions based on that list [7].

The use of blacklists to detect malicious URLs through traditional techniques is commonly employed due to its effectiveness and simplicity. Blacklists contain records of previously identified malicious URLs. According to Joshi et al. [10], the traditional detection method based on blacklists remains a classic technique in malicious URL detection because of its ability to identify and block URLs with known reputations for being malicious.

Next, we describe in Table I some of the key heuristics used to detect malicious URLs: URL length, similar domains, keywords, redirections, emergin domains and geo-graphical server location.

Table I. URL features

Feature	Description
Length of URL	Malicious URLs often have unusual lengths, either very long or very short. This heuristic can consider URLs that deviate significantly from the average length as potentially malicious.
Similar domains	This heuristic looks for domains that are very similar to legitimate domains but with small variations, such as adding or removing characters. For example, "google.com" and "goog1e.com". Attackers often use similar domains to trick users.
Keywords	This heuristic searches for keywords associated with malicious content in the URL, such as terms related to malware, phishing, scams, etc. If such keywords are detected, the URL may be tagged as suspicious.
Redirects	Malicious URLs often involve multiple redirects to hide their actual destination. This heuristic can track and count the number of redirects in an URL to identify suspicious behaviors.
Emerging Domains	This heuristic identifies domains that have been recently registered. Attackers often use new domains to conduct malicious campaigns before they are reported and blocked. Identifying newly registered domains can help identify malicious URLs.
Geographical location	Attackers create malicious URLs that resemble legitimate websites of local institutions (banks, companies, etc.) in a specific region to fool users in that area.

After reviewing several papers, we conclude that rules-based detection techniques often use the following URL characteristics: URL structure, content, and domain information.
Next, we move on to machine learning-based detection techniques. The motivation behind this approach is usually related to how rules-based techniques struggle to deal with often-changing URLs.

3.2 Machine learning-based detection techniques

A literature review conducted in 2022 highlighted the main techniques used for detecting malicious URLs using machine learning models [7, 17]. Authors identified challenges and limitations in existing studies, as well as the diversity of data sources used for training and evaluating the models.

Another study focused on deep learning approaches for phishing detection and found that most models employed supervised deep learning algorithms [3]. This research also highlighted key challenges in phishing detection and provided a foundation for future investigations.

In another study, authors conducted a systematic literature review with the aim of identifying, evaluating, and synthesizing results related to deep learning approaches for phishing detection. A notable finding in their study was that, except for one study, all analyzed models applied supervised deep learning algorithms in their phishing detection approaches [3].

These values are based on empirical studies that have analyzed malicious URL data sets. For example, the study by Aung, E et al. [4] found that "URL Length Heuristics" prevails with 28% preference, highlighting the importance of length as a threat indicator. Exploration

of "Similar Docs" covers 22% of elections, where related domain analysis can unravel suspicious patterns.

By considering URL length as a relevant feature, machine learning models can identify common patterns associated with malicious URLs. For example, malicious URLs may have unusually long or short lengths compared to benign URLs. By capturing this feature and employing machine learning techniques, models can learn to distinguish between malicious and benign URLs with greater accuracy.

After reviewing the related work, we conclude that the most common URL characteristics used in machine learning techniques include features related to the URL structure, content, and numerical information.

3.3 Comparison of detection techniques

We wrap up this section with a comparison between machine learning-based and rules-based techniques. Table II presents an overview of these approaches, highlighting their common uses, disadvantages and advantages.

Table II. Comparison of Machine Learning-Based and Rules-Based techniques

Machine learning-based	Rules-based
Common uses	**Common uses**
These techniques are employed by antivirus companies and other cybersecurity professionals to enhance the detection of sophisticated cyberattacks such as phishing and drive-by downloads [11].	These techniques are used by cybersecurity companies, email service providers, and web browsers to safeguard users against phishing attacks and various types of malware threats [13].
Disadvantages	**Disadvantages**
There are challenges associated with feature selection and detecting changes in the features that differentiate between legitimate and suspicious URLs.	Traditional methods can be ineffective against botnets that employ sophisticated Domain Generation Algorithm (DGA) to conceal command and control (C&C) server domain names [12].
Other challenges include the lack of analysis and detection of obfuscated JavaScript on web pages, handling outliers, feature selection, and the effectiveness of features [14].	Inability to discover new, unknown sites not in the database and the computationally intensive nature of searching a large database for an URL [16].
The proposed approach carries the potential of incorrectly identifying benign URLs as malicious.	Challenges in keeping an updated and all-encompassing blacklist due to the frequent creation of new malicious URLs by attackers.
It may necessitate a substantial volume of training data and computational resources for both model training and execution.	Ineffectiveness in identifying malicious URLs not included in the blacklist [13].
Often exhibit greater complexity compared to traditional methods, potentially making their implementation and maintenance more challenging [11, 12].	Inability to uncover new or unknown sites absent from the database and the resource-intensive nature of searching an extensive database for an URL [16].
Machine learning models excel at uncovering intricate data patterns that might prove challenging for traditional methods. Machine learning models are effective at diminishing data noise.	This method can be more straightforward and easier to explain compared to more complex techniques. Blacklist services are integrated into browser toolbars, mobile apps, and search engines to prevent access to malicious URLs.
Advantages	**Advantages**
The proposed approach is scalable, capable of handling substantial volumes of URL data, and suitable for high-demand environments.	This technique is well-established and widely adopted. Antivirus providers maintain extensive lists of harmful URLs as part of their blacklist strategy, and users are promptly alerted when clicking on a link matching the current list of harmful URLs [13].
The proposed approach can be employed to detect various types of malicious URLs, including	

phishing and drive-by downloads [11, 12].

4. Conclusions

Three research questions motivated this paper. First, *what are the most used characteristics to identify malicious URLs?* The most used URL characteristics used to identify malicious URLs were listed and described in Section 2: Length of URL, similar domains, keywords, redirects, emerging domains and geographical location. Second, *which URL characteristics are more common in rules-based techniques?* We showed in Section 3 that URL characteristics used in rules-based techniques include features related to the URL structure, content, and domain information. Third, *which URL characteristics are more common in machine learning-based techniques?* We explained in Section 3 that the most common URL characteristics used in machine learning techniques include features related to the URL structure, content, and numerical information.

Detecting malicious URLs is an open research problem, with multiple papers published in the last year. In this paper, we first described the most relevant characteristics used for malicious URL detection. After that, we provided a taxonomy of detection techniques including machine learning and rules-based methods. Next, we focused on identifying which characteristics are more common for each type of detection method.

References

1. Aljabri, M et al.. Detecting malicious URLs using machine learning techniques: review and research directions. IEEE Access. 2022.
2. Sahoo, Doyen, Chenghao Liu, and Steven CH Hoi. "Malicious URL detection using machine learning: A survey." arXiv preprint arXiv:1701.07179, 2017.
3. Cheng, Du, et al. "Profiling Malicious Domain by Multidimensional Features." IEEE International Conference on Robots & Intelligent System (ICRIS), 2018.
4. Bharadwaj, Rohit, et al. "Is this URL safe: detection of malicious URLs using global vector for word representation." IEEE Interna-tional Conference on Information Networking (ICOIN), 2022.
5. Vijayalakshmi, M., Mercy Shalinie, S., Yang, M. H., & U, R. M. (2020). Web phishing detection techniques: a survey on the state-of-the-art, taxonomy and future directions. Iet Networks, 9(5), 235-246.
6. S. Mahdavifar, N. Maleki, H. A. Lashkari, M. Broda y A. H. Razavi, “Classifying Malicious Domains using DNS Traffic Analysis,” de The 19th IEEE International Conference on Dependable, Autonomic, and Secure Computing (DASC), 2021.
7. Madhubala, R., et al. "Survey on Malicious URL Detection Techniques." IEEE International Conference on Trends in Electronics and Informatics (ICOEI), 2022.
8. Bhoj, Naman, et al. "Naive and neighbor approach for phishing detection." 2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT). IEEE, 2021.
9. Aung, E. S., & Yamana, H. (2020). Malicious URL detection: a survey. In DEIM Forum F6-3 (Vol. 290).
10. Joshi, A., Lloyd, L., Westin, P., & Seethapathy, S. (2019). Using lexical features for malicious URL detection--a machine learning approach. arXiv preprint arXiv:1910.06277.
11. N. A. ALfouzan and N. C, "A Systematic Approach for Malware URL Recognition," 2022 2nd International Conference on Computing and Information Technology (ICCIT), Tabuk, Saudi Arabia, 2022, pp. 325-329, doi: 10.1109/ICCIT52419.2022.9711614.
12. Saeed, A.M.H., Wang, D., Alnedhari, H.A.M., Mei, K., Wang, J. (2022). A Survey of Machine Learning and Deep Learning Based DGA Detection Techniques. In: Qiu, M., Gai, K., Qiu, H. (eds) Smart Computing and Communication. SmartCom 2021. Lecture Notes in Computer Science, vol 13202. Springer, Cham. https://doi-org.ezproxy.sibdi.ucr.ac.cr/10.1007/978-3-030-97774-0_12

13. Madhubala, R., Rajesh, N., Shaheetha, L., & Arulkumar, N. (2022, April). Survey on Malicious URL Detection Techniques. In 2022 6th International Conference on Trends in Electronics and Informatics (ICOEI) (pp. 778-781). IEEE.
14. A. Kumar and S. Maity, "Detecting Malicious URLs using Lexical Analysis and Network Activities," 2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2022, pp. 570-575.
15. Yogesh, B., & Reddy, G. S., Detection of Malware in the Network Using Machine Learning Techniques. In 2022 International Conference on Recent Trends in Microelectronics, Automation, Computing and Communications Systems (ICMACC) (pp. 204-211). IEEE.
16. Saxena, A., Arora, A., Saxena, S., & Kumar, A. (2022, June). Detection of web attacks using machine learning based URL classification techniques. In 2022 2nd International Conference on Intelligent Technologies (CONIT) (pp. 1-13). IEEE.
17. T. Wang y L. Chen, “Detecting Algorithmically Generated Domains Using Data Visualization and N-Grams Methods,” de Proceedings of Student-Faculty Research Day, CSIS, 2017.
18. T. Kelley y E. Furey, “Getting Prepared for the Next Botnet Attack : Detecting Algorithmically Generated Domains in Botnet Command and Control,” de 29th Irish Signals and Systems Conference (ISSC), 2018.
19. J. Kumar, A. Santhanavijayan y B. Janet, “Phishing Website Classification and Detection,” International Conference on Computer Communication and Informatics, 2020.
20. Z. Bao, W. Wang y Y. Lan, “Using Passive DNS to Detect Malicious Domain Name,” de International Conference on Vision, Image and Signal Processing, Vancouver, BC, Canada, 2019.

Anexo 3: JIFI 2024

Detección de URLs maliciosas en tiempo real Detecting malicious URLs in real time

Diego Orozco, Luis Quesada, Kryscia Ramírez y Adrián

Lara Investigadores, Escuela de Ciencias de la Computación

e Informática e-mail: diego.orozco@ ucr.ac.cr

luis.quesada@ucr

.ac.cr

kryscia.ramirez@

ucr.ac.cr

adrian.lara@ucr.a

c.cr

Resumen:

Una URL (Uniform Resource Locator) maliciosa es un enlace creado para lograr que las personas visiten sitios web maliciosos, donde ocurren ataques de seguridad como phishing o spyware. Estas URLs maliciosas suelen tener patrones definidos, por lo que varios artículos han propuesto técnicas para detectarlas usando aprendizaje automático. Sin embargo, hay poco trabajo dedicado a estudiar el impacto que tiene esta detección en tiempo real. En este artículo, proponemos un mecanismo de detección implementado en el sistema de dominio (DNS) usando listas negras y analizamos el impacto en la latencia. Nuestros resultados indican que la latencia aumenta conforme crece el tamaño de la lista negra.

Palabras clave: Detección en tiempo real, DNS, latencia, seguridad , URL maliciosa Abstract:

A malicious URL is a link created to trick users into visiting illegitimate website, where phishing, spyware and other security threats exist. These URLs commonly have specific patterns that can be detected by machine learning classifiers. Several papers have proposed classification techniques for malicious URLs. However, very few works have addressed the problem of incorporating these detection techniques into a real time scenario. In this paper, we propose a real time detection tool that is implemented in the

domain name system (DNS). Our experimental design focuses on measuring the impact of the classifier on latency using black lists. Our results show that a progressive increase in latency as the size of the blacklist size increases.

Keywords: DNS, latency, malicious URL, real time detection, security

1. INTRODUCTION

Malicious URLs have become a constant concern, as attackers use them for criminal activities like phishing, malware distribution, and theft of sensitive information. Identifying these harmful URLs is crucial to safeguarding users from potential threats. Unlike other security threats, URLs cannot be blocked. Indeed, URLs are needed for the Internet to function. The domain name system (DNS) translates URLs into IP addresses that computers understand. This function is essential for the proper operation of online services like website access and email communication. Therefore, effective mechanisms for detecting malicious URLs within the DNS are vital for online safety, as early and accurate detection can prevent access to harmful content and protect users from attacks. In most traditional networks, URLs are not classified or filtered, increasing the risk of phishing, botnets, and other malicious threats.

Real-time detection of malicious URLs is a challenging problem that has received little attention. Numerous supervised machine learning classifiers for URLs have been developed. Likewise, many articles have proposed using black lists. These are extensive lists of URLs that are known to be used with malicious intent. However, these techniques work offline and are not capable of actually stopping malicious URLs. To implement a real-time detection techniques, some classification technique (machine learning, black-list or other) must be implemented inside a DNS proxy, so that each URL received can be scanned before translation.

In this paper, we implemented a DNS proxy capable of detecting malicious URLs using black lists. Our goal was to evaluate the impact of adding the proxy between the end user and the DNS in terms of latency (the time needed for the request to be handled). Indeed, the DNS is used every time that a new URLs is clicked on and performance is important. Therefore, we were interested in investigating the tradeoff between increased security when adding URL classification and a potencial delay in the response time.

The remaining of this paper is organized as follows. In section II we describe the background and related work. Next, in section III, we explain our experimental design. Finally, we analize the results in section IV and we draw our conclusions in section V.

2. BACKGROUND AND RELATED WORK

In this section, we present the relevant background related to the detection of malicious URLs and the impact on the latency of domain name systems.

2.1 Malicious URL Detection through Rule-Based Models

The use of blacklists to detect malicious URLs through traditional techniques is commonly employed due to its effectiveness and simplicity. Blacklists contain records of previously identified malicious URLs. According to Joshi et al. [5], the traditional method of blacklist-based detection remains a classic technique in malicious URL detection due to its ability to identify and block URLs with known reputations for being malicious.

Previous research has examined and evaluated rule-based techniques, such as blacklists and whitelists, in terms of their effectiveness in malicious URL detection. However, these approaches may have limitations, especially concerning the latency they generate. Rule-based processing can require a significant amount of time to query a list of known URLs and make decisions based on that list [6].

Studies demonstrate the effectiveness of blacklists as a classification approach for identifying known malicious URLs [5]. However, it is also noted that this approach may be limited due to the infrequent updating of blacklists, which hinders the detection of new threats.

Previous studies have examined and evaluated rule-based techniques, such as blacklists and whitelists, in terms of their effectiveness in detecting malicious URLs. However, these approaches can have limitations, especially regarding the latency they generate. Rule-based processing may require significant time to query a list of known URLs and make decisions based on that list [7].

The use of blacklists to detect malicious URLs through traditional techniques is

commonly employed due to its effectiveness and simplicity. Blacklists contain records of previously identified malicious URLs. According to Joshi et al. [5], the traditional detection method based on blacklists remains a classic technique in malicious URL detection because of its ability to identify and block URLs with known reputations for being malicious.

2.2 Latency in Malicious URL Detection

Determining the most suitable algorithm to use requires considering various factors, such as computational efficiency and real-time performance. Some machine learning algorithms, such as those based on decision trees, are known for their fast-processing capabilities and low latency [8]. On the other hand, neural networks may offer higher accuracy but may require more computational resources, potentially leading to increased latency in domain name systems [7].

Regarding the impact on domain name system latency, the introduction of new mechanisms for detecting malicious URLs has been observed to affect DNS response times [9]. Increased latency can be problematic in environments where quick domain name resolution is required, such as in critical applications or high-traffic networks. Therefore, it is essential to evaluate and compare the latency impact of different malicious URL detection approaches, including non-classification, blacklist-based classification, and the incorporation of machine learning models in real-time DNS controllers.

Next, we describe our experimental design.

3. EXPERIMENTAL DESIGN

The goal of the experiment was to evaluate the influence of two primary factors: "Blacklist Size" and "Location" on the latency of DNS queries. The former represents the manipulation of the Blacklist size, featuring three distinct levels (1000 elements, 5000 elements, and 10000 elements). On the other hand, "Execution location" has two levels: local and cloud. This factor signifies the environment in which the experiment. Together, these factors form a factorial 2x3 design. The response variable is the "system latency." This variable, expressed in seconds (s), reflects the time required from the moment a DNS request is received until the legitimacy of a URL in the DNS controller is verified. Furthermore, we identified two key categorical factors that influence latency. A set of URLs is selected for experimentation. These URLs are obtained from Mendeley Data [1], which contains many URLs. A sample of non-malicious URLs is extracted, as well as a sample of malicious URLs.

TABLE II LATENCIES BY LOCATIONS

	Mean (s)	Variance	Stan. Dev
Local	10.06	216073.16	464.83
Cloud	0.008	0.02	0.15
Latency	5.03	108049.43	328.70

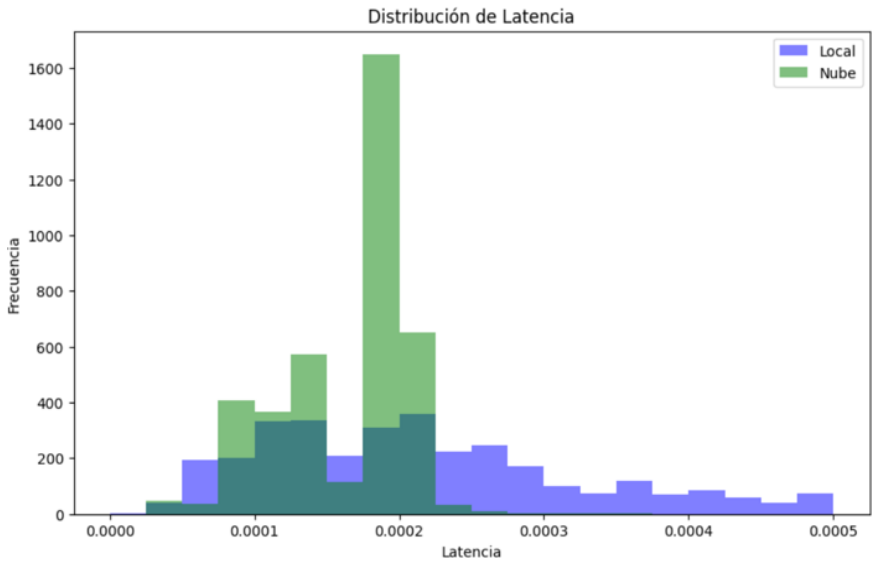

Figure 4: Latency distribution

4. RESULTS

Table II provides key latency statistics for "Local" and "Cloud" locations. It highlights mean, variance, and standard deviation for both, emphasizing significant differences. Notably, the cloud location exhibits substantially lower mean latency and variance, suggesting more consistent data compared to the local location.

Figure 4 displays latency distribution, representing local latency in blue and cloud latency in green. The x-axis measures latency in seconds, and the y-axis indicates frequency. Taller bars signal higher frequency in respective intervals. A shared x-axis range eases comparison,

emphasizing relevant latency values for analysis.

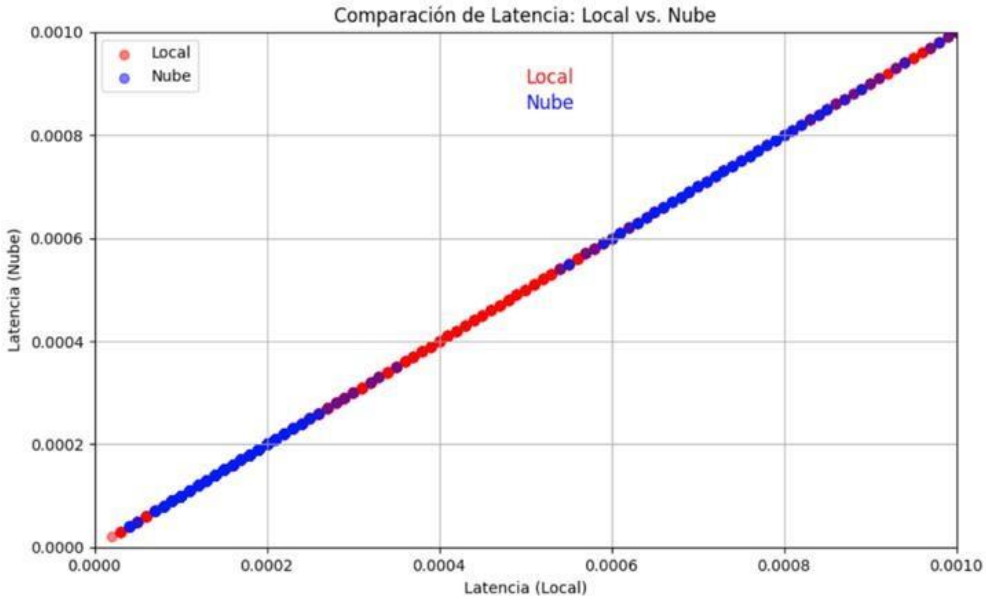

Figure 5: Latency comparison by location

Figure 5 shows the correlation or pattern in latency data. A noticeable upward slope from the bottom-left to the top-right may suggest generally lower latency in the "Cloud" location compared to the "Local" location. Points tend to cluster toward the upper-right corner, indicating lower latencies in the "Cloud." This suggests that, in this context, the cloud may offer faster response times compared to the local location.

Statistical tests were conducted to assess the necessary assumptions before performing analysis of variance (ANOVA) or non-parametric analysis. Results showed that non-parametric analysis was required. As a result, a Kruskal-Wallis test provided statistical evidence that there was at least one group that differed significantly from the others in terms of the evaluated variable The results of these tests are omitted due to space limitations. Below, we highlight the main findings of our experiments.

Impact of blacklist size. First, as the blacklist size increases (1000, 5000, 10000 elements), the system latency tends to increase. This phenomenon can be explained by the fact that larger blacklists

require more time to be processed and verified during URL queries.

Impact of Proxy Location. Second, significant differences are observed between latency in the local and cloud locations. The local location tends to have lower latencies compared to the cloud. This is attributed to the inherent characteristics of cloud queries, which may experience additional delays due to data transfer over the network.

Interaction of Factors. Third, the interaction between blacklist size and proxy location also influences the results. For example, in the cloud, the impact of blacklist size may be more pronounced than in the local location, contributing to variations in latency. The variability in the results is explained by a profound understanding of how experimental factors, such as black list size and proxy location, interact and affect system performance.

5. CONCLUSIONS AND FUTURE WORK

Our results show that both the size of the blacklist and the location have a statistically significant impact on latency.

Considering the current conclusions, we would like to address the identified limitations. To do so, looking into the non-linear relationship between blacklist size and latency could provide valuable insights for designing more efficient security systems. The inclusion of additional variables, such as network load and the complexity of DNS queries, could further enrich the understanding of factors influencing latency.

Furthermore, we recommend investigating specific approaches to mitigate the negative impact of blacklist size on latency, considering caching strategies or more efficient filtering algorithms. Validating these results in real-world environments and assessing their applicability in dynamic and changing scenarios would be crucial steps for the practical implementation of the current findings.

ACKNOWLEDGEMENTS

This work was partially funded by CITIC and the project “834-C2-145 Edificios inteligentes y computación afectiva para mejorar la interacción humano-robot.”

REFERENCIAS

[1] Mendeley, 2020. “Dataset of Malicious and Benign Webpages”. URL: https://data.mendeley.com/datasets/gdx3pkwp47/2

[2] Aljabri, M., Altamimi, H. S., Albelali, S. A., Al-Harbi, M., Alhuraib, H. T., Alotaibi, N.

K., ... & Salah, K. (2022). Detecting malicious URLs using machine learning techniques: review and research directions. IEEE Access, 10, 121395-121417.

[3] Doyen, S. (2019). Malicious URL detection using machine learning: a survey. ArXiv, 1701, v3.

[4] Orozco-Fonseca, D., Marín, G., & Lara, A. (2024). Taxonomy of Malicious URL Detection Techniques. In International Conference on Information Technology & Systems (pp. 73-81). Cham: Springer Nature Switzerland.

[5] Joshi, A., Lloyd, L., Westin, P., & Seethapathy, S. (2019). Using lexical features for malicious URL detection--a machine learning approach. arXiv preprint arXiv:1910.06277

[6] Madhubala, R., Rajesh, N., Shaheetha, L., & Arulkumar, N. (2022, April). Survey on malicious URL detection techniques. In 2022 6th International Conference on Trends in Electronics and Informatics (ICOEI) (pp. 778-781). IEEE.

[7] Cheng, D., Liu, Z., Zhang, P., Zeng, Y., Cui, J., & Kong, L. (2018). Profiling Malicious Domain by Multidimensional Features. In 2018 International Conference on Robots & Intelligent System (ICRIS) (pp. 489-495). IEEE.

[8] Janet, B., & Kumar, R. J. A. (2021). Malicious URL detection: a comparative study. In 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS) (pp. 1147-1151). IEEE.

[9] Khormali, A., Park, J., Alasmary, H., Anwar, A., Saad, M., & Mohaisen, D. (2021). Domain name system security and privacy: A contemporary survey. Computer Networks, 185, 107699.

[10] Cersosimo, M., & Lara, A. (2022). Detecting malicious domains using the splunk machine learning toolkit. In NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium (pp. 1-6). IEEE.

Anexo 4: Latincom 2024

Real-time malicious URL detection

Diego Orozco Fonseca, Luis Quesada Quirós, Kryscia Ramirez Benavides, Adrian Lara
Universidad de Costa Rica, San José, Costa Rica

***Abstract*--** This Malicious URLs are constantly used for phishing, malware distribution and other illegal activities. Because benign URLs are needed for the Internet to function, malicious URLs are hard to block. While several works have focused on offline classification of malicious URLs, real-time detection still needs to be looked into. This paper evaluates the performance of real-time malicious URL detection using two techniques: blacklist methods and machine learning methods, deployed in both local and cloud environments. The study highlights significant differences in latency and connection failure rates under various load conditions, providing insights into the strengths and limitations of each approach. The blacklist method consistently demonstrates lower latency, making it suitable for scenarios requiring quick response times, though its stability may be compromised under high loads in a local setup. In contrast, the machine learning method offers advanced detection capabilities but exhibits higher latency, particularly in local environments, due to its resource-intensive nature. The cloud environment mitigates some latency issues but still lags behind the blacklist method in terms of speed. The findings emphasize that most latency stems from the verification process, with the local environment requiring significant optimization to reduce delays. The study concludes that implementing a proxy for real-time URL detection is viable, especially in cloud environments, where resource management can better handle increased demand.

Introduction

In today's digital age, the increasing number of cyber threats presents significant challenges for computer system security and online user protection. Among these threats, malicious URLs have emerged as a constant concern, used by attackers to carry out various malicious activities such as phishing, malware distribution, and theft of sensitive information.

The Domain Name System (DNS) is a key component of the Internet's architecture It is responsible for converting human-readable domain names into computer-readable IP addresses. Blocking harmful content at the DNS level is challenging due to the vast and constantly changing nature of the internet. New domain names are continually created, and malicious actors often move quickly to exploit these new domains before they can be identified and blocked. Additionally, DNS queries are typically designed to be fast and efficient, leaving little room for extensive security checks. Despite these challenges, identifying malicious URLs accurately and quickly within the DNS is important for preventing unauthorized access to harmful content and maintaining data integrity.

Despite advancements in malicious URL detection, integrating real-time detection systems into DNS infrastructure remains a significant challenge. Adopting real-time detection mechanisms raises concerns about potential system latency degradation, which could negatively impact online application performance and user experience. While classifiers based on supervised machine learning (ML) and other detection techniques have been developed, real-time integration of these systems into DNS infrastructure still needs to be improved primarily due to the possibility of increased latency. Moreover, there is a notable gap in the existing literature regarding research focusing on real-time DNS detection mechanisms. The lack of studies addressing this aspect underscores the need for further investigation and exploration.

Our contribution is a DNS proxy capable of classifying malicious URLs using different techniques and running locally or in the cloud. This study examines how variables such as proxy location, the detection technique used, and system load affect latency and system responsiveness to assess the feasibility of integrating real-time malicious URL detection systems into DNS infrastructure.

We deployed our proxy locally and in the cloud (AWS). All incoming requests were received by this proxy, which then directed them for validation to either a blacklist and an ML classifier. If a URL was considered malicious, the proxy refused the request and informed the client that it was unsuccessful. If the URL was not malicious, the proxy sent the request to Google's DNS to get the relevant IP address before returning it to the client. The entire process was timed to evaluate latency, starting from the client's initial request and ending with the final response.

Another valuable aspect of this study is the analysis of how the proxy handles different loads. We tested the system by sending requests at various levels of demand. This approach helped us understand how the proxy's location, the detection method, and the number of requests affect system latency and overall performance.

The experiments conducted in both cloud and local environments using blacklist and machine learning methods showed that the detection

technique and proxy location influence system latency. Our results indicate that while machine learning tends to have higher latency compared to blacklist methods, this difference does not significantly affect the system's functionality. Despite the higher latency with machine learning, both methods are still usable and perform effectively in real-world scenarios.

The remainder of this article is organized as follows: Section 2 discusses related work in malicious URL detection. Section 3 describes the experimental setup. Section 4 describes the results and analysis of the experiments. Finally, Section 5 concludes the paper and discusses future work.

Related Work

Due to the increasing prevalence of cyber threats, detecting malicious URLs has been a significant area of research. Various methods have been developed and studied extensively, each with advantages and limitations.

Offline detection methods

Black lists are one of the most popular methods for identifying malicious URLs. One can use this method to cross-reference newly created URLs by keeping track of known harmful URLs. Because the blacklist is updated slowly, this method is ineffective against emerging threats but successful against known ones. While showing the effectiveness of blacklist-based detection, studies like those conducted by Joshi has also brought attention to some of its drawbacks [1].

Machine learning (ML) techniques have recently become popular for detecting malicious URLs [3]. These techniques involve training models on features extracted from URLs, like length, domain name, and the presence of suspicious keywords. The models can classify URLs as benign or malicious based on these features. Sahoo thoroughly review the different ML techniques used for this purpose, highlighting the potential for higher accuracy than traditional methods [2].

URL characteristics are divided into two main categories: static and dynamic. Static features are attributes that can be obtained without directly interacting with the URL. On the other hand, dynamic features require a deeper interaction with the URL. While they can achieve high accuracy in detecting malicious URLs, their collection often consumes time and resources, making them less suitable for real-time analysis [3]. In previous work [4], we proposed a taxonomy of URL characteristics, describing which ones were more useful for each classification technique.

In the field of malicious URL detection, Morales developed a model for detecting malicious URLs using supervised machine learning techniques [7]. This model leverages various features extracted from URLs, such as lexical and host-based characteristics, to identify potentially harmful links. By training the model on a dataset of known malicious and benign URLs, the system can effectively distinguish between safe and dangerous URLs.

There have also been attempts to evaluate various detection strategies and create hybrid algorithms that combine the best features of several approaches. For instance, Xuan's study contrasts machine learning models with conventional methods and suggests that combining both approaches can result in more reliable security [5].

Real-Time detection methods

High latency can negatively impact online application performance and user experience, so it is important to consider how detection mechanisms affect latency when integrating them into real-time systems. Research by Khormali has looked at how different DNS security measures affect latency, emphasizing the need for effective detection mechanisms that do not sacrifice performance [6].

In the context of live streaming platforms like Twitch, detecting malicious URLs in real-time is important for protecting users from cyber-attacks. According to the authors, Twitch chatrooms are an ideal target for phishing attacks and other cyber threats due to constant interactions between users and streamers. Attackers often use these interactions to post harmful URLs, leading users to download malicious software or reveal sensitive information [8].

Janet uses lexical and semantic features to detect malicious URLs quickly and effectively. This combination of features allows the machine learning model, specifically a Random Forest classifier, to accurately determine whether a URL is malicious or safe. The results showed that this approach is not only accurate but also fast enough for real-time applications, which is important for maintaining security and integrity in Twitch chatrooms [8].

The work by Hamidouche provides a significant contribution to understanding the latency implications for resource-constrained devices. Their study focuses on implementing a lightweight model for DNS tunnel detection, optimized for devices with limited computational resources, such as IoT devices. A key finding of their research is the importance of maintaining processing latencies under 1 millisecond to ensure timely threat detection and response [9].

Next, we describe our experimental design.

Experimental design

Proxy Implementation

A custom proxy was created to receive all incoming URL requests. Every request was routed through this proxy to the proper detection mechanism (the blacklist or the ML classifier), which then handled the ensuing actions according to the detection results—a local server with the proxy installed processed requests from an internal network in the local configuration. The proxy was put up on AWS for the cloud setup, allowing

requests to be handled in a scalable, virtualized environment.

The experimental configuration included two main environments: a cloud setup utilizing Amazon Web Services (AWS) and a local setup representing an organization's internal architecture. The two environments allowed for a thorough examination of how well the system performed in various scenarios (see Figure 1).

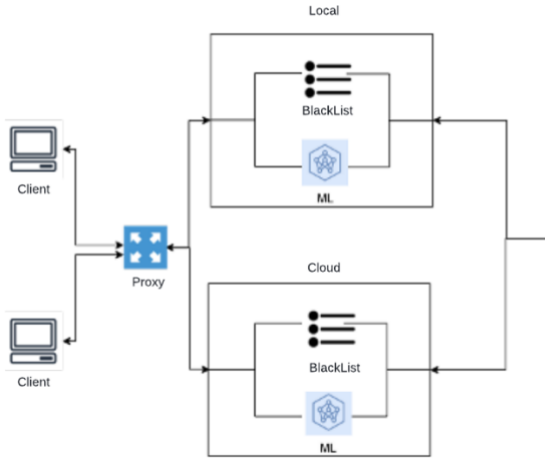

Fig. 1. Experiment topology

The proxy system handled and verified URL requests according to a predetermined procedure. Initially, customers sent URL queries to the proxy. Every request was subsequently forwarded to the detection mechanisms. The request was matched to a pre-compiled list of known malicious URLs for blacklist-based detection. The URL was marked as malicious if it was discovered to be on the blacklist. A machine learning model trained to recognize dangerous patterns based on different URL attributes was used to assess the URL for machine learning-based detection.

The entire process was timed, from receiving the request to delivering the response. This timing included the detection process and the DNS resolution to provide an accurate measure of latency.

The experiment involved submitting requests with different loads at the same time to evaluate the system's performance under various scenarios. Every load scenario was examined in local and cloud settings, and each setup's latency was calculated. The objective was to compare the performance of the local and cloud installations and assess how various loads affected the system's responsiveness.

The machine learning classifier used for detecting malicious URLs in this study is based on the "codebert-base-Malicious_URLs" model, developed by DunnBC22 and available as open-source on the Hugging Face repository, a well-known platform for machine learning models. This model uses the transformer architecture, designed for natural language processing (NLP) tasks. It processes input URLs to predict whether they are malicious or safe [10]. Additionally, it incorporates methods like Random Forest to make the model more robust and accurate. he classifier was trained on a substantial dataset of 651,191 URLs, including 428,103 benign URLs, 96,457 defacement URLs, 94,111 phishing URLs, and 32,520 malware URLs. This extensive dataset helps cover various types of malicious URLs, improving the accuracy of detection [10]. In this paper, we were not interested in evaluating the accuracy of this classifier. We simply used it as a black box to measure classification time.

Experimental design

The main objective of this study was to determine how different parameters affected system latency when detecting malicious URLs. The hypotheses tested included the null hypothesis (H0) and the alternative hypothesis (H1). The null hypothesis stated that there are no significant differences in the latency of the real-time malicious URL detection system between local and cloud locations, regardless of the detection technique and request load. In contrast, the alternative hypothesis suggested that the latency would vary significantly depending on these factors.

Three main factors were considered in this experiment. The first factor was the location, which has two levels: local, representing the internal infrastructure of an organization, and cloud, using Amazon Web Services (AWS). The second factor was the detection method, including blacklist-based and machine learning-based detection. The blacklist method involved using a pre-compiled list of known malicious URLs, while the machine learning method used a model trained to detect malicious URLs based on various features. The third factor was the request load, which was varied to include low load (10 requests per second), medium load (100 requests per second), and high load (500 requests per second).

This experimental design evaluated how location, detection technique, and request load affected the latency and performance of real-time malicious URL detection systems in DNS operations.

Statistical model

To ensure the validity of our results, we conducted a statistical test. We used a linear mixed-effects model to analyze the latency data, considering the interaction effects between the detection method, environment, and request load. The model is as follows:

$$\mu_{ijk} = \mu + \alpha_i + \beta_j + \delta_k + (\alpha\beta)_{ij} + v_l \quad (1)$$

Where:

- μ is the overall mean.
- α_i is the effect of the i-th level of the blacklist factor (detection method).

- β_j is the effect of the j-th level of the cloud factor (location).
- δ_k is the effect of the k-th level of the request load factor.
- $(\alpha\beta)_{ij}$ is the interaction effect between blacklist and cloud.
- v_l is the random effect of the parcel.

Next, we describe and discuss our results.

Results

This section focuses on analyzing the latency measurements obtained from different configurations and loads, as well as the performance comparisons between the blacklist and machine learning detection methods in both local and cloud environments.

Our primary focus was on measuring system latency, which we defined as the interval between sending a request for a URL and receiving a response. We measured both the DNS resolution and the detection procedure. The findings showed that delay varied significantly based on the environment and the detection technique.

Latency by detection method

The blacklist method demonstrated lower latency compared to machine learning across both local and cloud environments, as shown in Figure 2. These results indicate that the blacklist method maintains low latency across different load levels, with the cloud environment showing slightly better performance than the local environment. The differences in latency are minimal, making the blacklist method suitable for real-time detection with consistent performance.

On the other hand, the machine learning method showed higher latency, particularly in the local environment. This higher latency suggests that machine learning detection is more resource-intensive, leading to longer processing times. In the cloud environment, while the latency is lower compared to the local environment, it is still significantly higher than that of the blacklist method.

The graphs clearly show that the blacklist method provides a more efficient solution in terms of latency for real-time malicious URL detection. The machine learning method, while potentially offering more advanced detection capabilities, incurs a higher latency, especially in local environments. Therefore, the choice between these methods will depend on the specific requirements and constraints of the deployment environment.

Besides, the small error bars in both charts show that the standard deviation is low, meaning the latency measurements are very close to the average value. This indicates that the results are very consistent. The latency times are quite repetitive, with the local environment always around 900 milliseconds and the cloud environment always between 160 and 190 milliseconds, no matter the load. This lack of difference between the values means that both methods produce stable and reliable results under different conditions.

Latency by environment

In general, as shown in Figure 2, the local configuration had higher latency, particularly when it came to machine learning detection. Cloud-based solutions, like those provided by AWS, can leverage scalable resources, making them more powerful and optimized than local servers. This scalable and well-optimized architecture of the cloud reduces overall latency, a difference that is more pronounced with machine learning detection. This demonstrates the cloud's advantage when managing computationally demanding tasks.

The choice between these methods will depend on the specific requirements and constraints of the deployment environment. The cloud's ability to handle high computational demands with lower latency highlights its suitability for tasks requiring significant processing power.

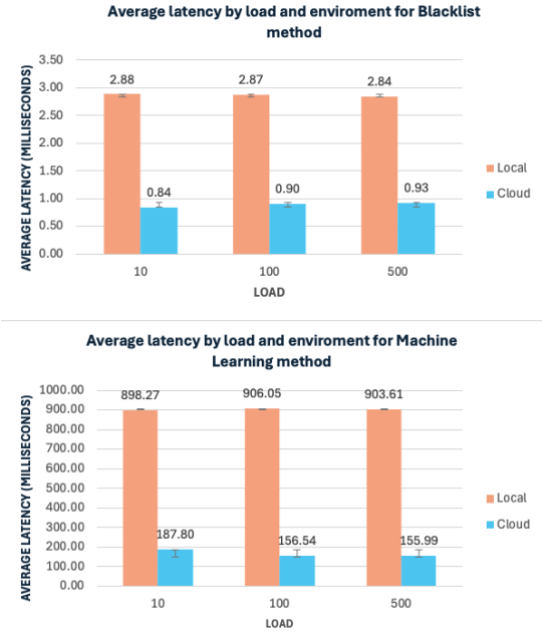

Fig. 2. Average latency by method

Normality test

The Q-Q plot (fig. 3) is used to check the normality of the residuals from our mixed-effects model. The plot compares the quantiles of the residuals to the quantiles of a standard normal distribution. The points should lie approximately along the reference line if the residuals are normally distributed. In our plot, the residuals mostly follow the reference line, with some deviations at the extremes. This suggests that the assumption of normality is reasonably met. While there is no homoscedasticity (constant variance), the model accounts for heteroscedasticity (varying variance) by including weights, which better estimates the standard error for comparisons. This method

ensures that our model can accurately capture the variability in the data.

The normal Q-Q plot (Fig 3) shows how our data's residuals compare to a perfect normal distribution. Most of the points are close to the diagonal line, meaning our residuals generally follow a normal pattern. However, at the far ends of the plot, some points deviate from the line, indicating the presence of a few outliers or data points with more extreme values than expected. These observations suggest that while our data mostly fits the normal distribution, there are some anomalies that might need closer examination.

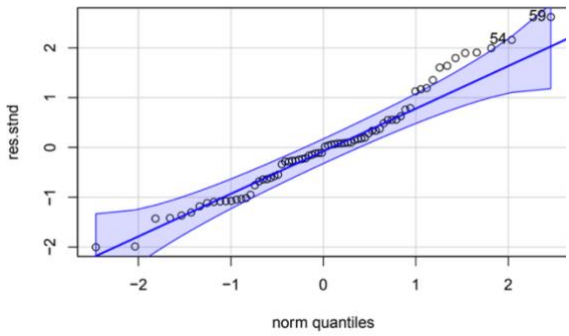

Fig. 3. Normality of the residuals

Request load

We evaluated the system under different request loads to understand how it handles varying levels of demand. The loads were categorized as low (10 requests), medium (100 requests), and high (500 requests), representing a small, intermediate, and large number of simultaneous requests, respectively.

- Low Load: Both detection methods performed well in terms of latency under low load conditions. The difference between blacklist and machine learning detection was minimal, indicating that the system can effectively handle low traffic with any approach,
- Medium Load: As the load increased to medium, the latency for machine learning detection started to rise more noticeably compared to blacklist detection. This tendency was seen in both local and cloud contexts, however it was less pronounced in the cloud.
- High Load: At high load levels, the latency for machine learning detection increased significantly, especially in the local environment. The cloud setup managed high loads better than the local setup, maintaining relatively lower latency even under heavy traffic.

Failure rate

Figure 4 shows the failure rate for both local and cloud environments as the load increases. Understanding the system's reliability and performance in real-world situations requires analyzing its resilience under growing demand.

Both environments handle the connections without significant problems up to a load of 500, suggesting consistent performance. However, when the load goes beyond this point, the number of failures increases, but in different ways for each environment. This variation indicates that while both setups can manage moderate loads effectively, higher loads expose differences in their ability to maintain stable connections.

As the load increases in the local setup, the number of failures rises progressively. This implies that when the local machine receives too many requests, it begins to struggle. This might be the result of insufficient resources, which makes it lose connections since it can't effectively handle the higher number of threads. However, failures in the cloud setup are also increasing, albeit more gradually and persistently.

The graph goes up to a load of 3000 to provide a clear and continuous view of how failures increase in both environments. It shows that beyond a certain point, the number of lost connections becomes somewhat proportional to the load, especially in the local environment. This indicates that, under high demand, the local setup might need optimization or scaling to handle the load better without losing many connections.

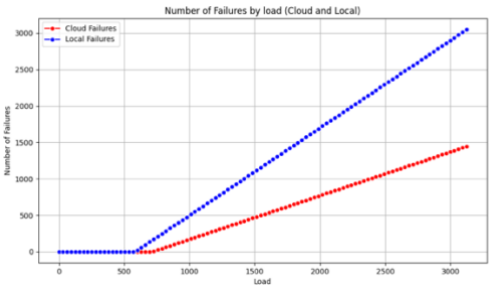

Fig. 4. Failures by load

Table I shows the average time for both the local and cloud environments. The base time represents the round-trip time to make a request without any type of verification. This base time is important for understanding the additional overhead introduced by the detection methods. These base times indicate the minimal latency involved in a simple round-trip request, without any additional processing. Most of the total time is spent on the verification process, which is where the detection methods come into play.

TABLE I
AVERAGE TIME IN MS

	Local	Cloud
Base	1.39	0.12
Detection Method	453.34	78.76

The experiments confirmed that the detection method and environment choice significantly impact system latency. Blacklist detection consistently outperformed machine learning in terms of speed, though machine learning offers potentially higher detection accuracy. The cloud environment provided superior performance, particularly for handling high loads and computationally intensive tasks like machine learning detection.

According to these results, it is possible to integrate real-time dangerous URL detection into DNS infrastructure; however, the deployment environment and detection method selection significantly impact performance. A cloud-based solution with blacklist detection might be better for applications that need high throughput and low latency. Nonetheless, cloud-based machine learning may be a good choice in situations when detection accuracy is important, and a certain level of latency is tolerable.

Discussion

The results from our experiments indicate a clear difference in performance between the cloud and local environments, particularly in terms of failure rates under varying loads. Up to a load of 500, both environments handled connections without significant issues, suggesting that the system performs reliably within this range. However, as the load increases beyond this point, the number of failures starts to rise, with the local environment experiencing a more pronounced increase. This indicates that the local setup begins to struggle under higher loads, likely due to resource limitations. On the other hand, the cloud environment, while also experiencing an increase in failures, does so more gradually, showcasing the benefits of scalable and optimized resources.

These findings are important for deciding where to deploy real-time malicious URL detection systems. The sharp rise in failures in the local setup under high loads suggests that local systems may need better resource management or scaling to handle increased demand. With its more gradual increase in failures, the cloud setup seems better suited for high-load scenarios because of its scalability.

A key point is that while latency isn't significantly affected by the detection technique used, whether machine learning or blacklist, the loss of connections becomes a major issue as the load increases. Both machine learning and blacklist techniques show very similar, low latency times, making the choice of technique insignificant to the user in terms of speed. However, the stability of connections is greatly impacted, and this becomes a significant problem with higher loads. This means that maintaining connection stability is more important for practical use than just focusing on latency.

For end users or final systems, the impact of these findings is important. The primary question is whether the proxy can be used without causing noticeable problems for users. The increase in connection failures, especially in local setups under high load, might lead to disruptions. For systems that need consistent and reliable network performance, the stability of connections is a critical factor. If the proxy causes too many connection failures, it could lead to system instability, affecting the user experience.

The usability of introducing this proxy depends on its impact on the overall system performance. If the benefits, such as effective malicious URL detection, outweigh the downsides, like increased connection failures, the proxy could be valuable. However, if the impact on connection stability is too significant, more optimization and testing would be needed to ensure it can handle higher loads without causing major disruptions.

According to Hamidouche, latency plays a role in the effectiveness of such systems. They mention that a processing latency of less than 1 millisecond (ms) is critical for timely detection and response to threats [9]. In our evaluation, the base time for the local environment is 1.39 ms, which is slightly above the recommended threshold, while the cloud environment has a base time of 0.12 ms, which is well within the acceptable range (Table I).

In terms of the detection method, the local environment has an average latency of 453.34 ms, and the cloud environment has an average latency of 78.76 ms. In our evaluation, we observed the following:

- Local Environment: Adds approximately 451.95 ms.
- Cloud Environment: Adds approximately 78.64 ms.

These results emphasize that most of the latency is due to the verification process. Therefore, while the proxy shows promise in terms of detection capabilities, its current performance in the local setup may lead to noticeable delays and potentially affect the user experience. Further optimization is required to meet the stringent latency requirements for real-time threat detection.

Ultimately, the decision to deploy this proxy in a real-world environment will depend on the specific requirements and constraints of the system. If low latency and stable connections are important, particularly under high loads, a cloud-based solution with blacklist detection might be more suitable. For scenarios where detection accuracy is paramount, and some level of latency is acceptable, cloud-based machine learning could be a viable option. Ensuring that the system can handle the intended load without compromising stability will be key to its successful implementation.

Conclusion

This study provides a comprehensive evaluation of real-time malicious URL detection systems in both local and cloud environments, focusing on the performance of two detection techniques: blacklist and machine learning. Our findings highlight the significant differences in latency and failure rates under varying load conditions, offering valuable insights into the strengths and limitations of each approach.

Implementing a proxy for real-time malicious URL detection is a viable solution, but its effectiveness depends heavily on the deployment environment and the detection technique used. The results show that:

Blacklist method

This method consistently demonstrated lower latency compared to the machine learning method in both local and cloud environments. Its minimal latency and consistent performance under different loads make it suitable for scenarios where quick response times are critical.

However, as the load increases, the number of connection failures also rises, especially in the local environment. This suggests that while efficient, the blacklist method's stability may be compromised under high loads in a local setup.

Machine Learning method

This method, although offering potentially more advanced detection capabilities, exhibited significantly higher latency, particularly in the local environment. This high latency highlights the resource-intensive nature of machine learning, which may not be ideal for environments where low latency is essential.

The cloud environment managed to reduce latency compared to the local setup, but it was still higher than the blacklist method. Additionally, the machine learning method also experienced an increase in connection failures under heavy loads.

The decision to implement this proxy in a real-world environment depends on the specific requirements and constraints of the system. If low latency and connection stability are crucial, especially under high loads, a cloud-based solution with blacklist detection might be more suitable. On the other hand, in situations where detection accuracy is important and some level of latency can be tolerated, a cloud-based machine learning approach could be a feasible solution.

In future work, we would like to improve our proxy so that it can handle a larger number of requests without dropping requests. As a point of reference, Google’s public DNS allows up to 1500 requests per second. After that, they may start blocking requests [11]. If we can improve our proxy to match this capability, it would greatly enhance its reliability and robustness, especially under high demand. This would make our solution much more effective for real-time malicious URL detection [11].

References

Joshi, A., Lloyd, L., Westin, P., & Seethapathy, S. (2019). Using lexical features for malicious URL detection--a machine learning approach. arXiv preprint arXiv:1910.06277

Sahoo, D., Liu, C., & Hoi, S. C. (2017). Malicious URL detection using machine learning: A survey. *arXiv preprint arXiv:1701.07179*.

Aljabri, M., Altamimi, H. S., Albelali, S. A., Al-Harbi, M., Alhuraib, H. T., Alotaibi, N. K., ... & Salah, K. (2022). Detecting malicious URLs using machine learning techniques: review and research directions. IEEE Access, 10, 121395-121417.

Orozco-Fonseca, D., Marín, G., & Lara, A. (2024). Taxonomy of Malicious URL Detection Techniques. In International Conference on Information Technology & Systems (pp. 73-81). Cham: Springer Nature Switzerland.

Do Xuan, C., Nguyen, H. D., & Tisenko, V. N. (2020). Malicious URL detection based on machine learning. *International Journal of Advanced Computer Science and Applications*,

Khormali, A., Park, J., Alasmary, H., Anwar, A., Saad, M., & Mohaisen, D. (2021). Domain name system security and privacy: A contemporary survey. Computer Networks, 185, 107699..

M. Cersosimo and A. Lara, "Detecting Malicious Domains using the Splunk Machine Learning Toolkit," NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 2022, pp. 1-6, doi: 10.1109/NOMS54207.2022.9789899.

B. Janet, A. Nikam and J. A. Kumar R, "Real Time Malicious URL Detection on twitch using Machine Learning," 2022 International Conference on Electronics and Renewable Systems (ICEARS), Tuticorin, India, 2022, pp. 1185-1189, doi: 10.1109/ICEARS53579.2022.9751862.

Hamidouche, M., Demissie, B. F., & Cherif, B. (2024). Real-time Threat Detection Strategies for Resource-constrained Devices. arXiv preprint arXiv:2403.15078.

DunnBC22. (2023). Codebert base Malicious URLs. Hugging Face. Retrieved from https://huggingface.co/DunnBC22/codebert-base-Malicious_URLs/tree/main

Google. (2023). Public DNS: Get started with public DNS. Google Developers. Retrieved from https://developers.google.com/speed/public-dns/docs/isp?hl=es-419