Universidad de Valladolid

Máster universitario

Ingeniería Informática

Trabajo Fin de Máster

phishingutils: Herramientas y comparativas para el estudio de métodos de detección de URLs *phishing*

Realizado por **Sergio Agudelo Bernal**

✠✠✠

Universidad de Valladolid

13 de septiembre de 2024

Tutor: D. Jesús María Vegas Hernández

Universidad de Valladolid

Máster universitario en Ingeniería Informática

D. Jesús María Vegas Hernández, profesor del departamento de Informática, área de Ciencias de la Computación e Inteligencia Artificial.

Expone:

Que el alumno D. Sergio Agudelo Bernal, ha realizado el Trabajo final de Máster en Ingeniería Informática titulado "*PHISHINGUTILS*: Herramientas y comparativas para el estudio de métodos de detección de URLs *PHISHING*".

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Valladolid, 13 de septiembre de 2024

V°. B°. del Tutor:

D. Jesús María Vegas Hernández

Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

Descriptores

Palabras separadas por comas que identifiquen el contenido del proyecto Ej: servidor web, buscador de vuelos, android ...

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

keywords separated by commas.

Índice general

Índice de figuras

Índice de tablas

1: Introducción

En un mundo cada vez más dependiente de las tecnologías de la información y comunicaciones, el acontecimiento de ciberataques trae mayores riesgos y posible impacto negativo en las sociedades. Como evidencia, 2023 fue un año en que ocurrieron importantes filtraciones de datos masivas, e incidentes que comprometieron el funcionamiento de entidades de ámbito nacional [10].

Entre las amenazas principales, el *phishing* se ha mantenido como el vector de acceso inicial más común por su inherente relación con la comunicación humana, y por la diversidad de métodos de ataque, los cuales se han vuelto más sofisticados ante la aparición de nuevos avances, como la IA generativa [5]. El medio más usual para transmitir ataques de *phishing* es a través de páginas web, ya que sus URL pueden ser incrustadas en cualquier tipo de mensajería virtual. Por lo anterior, se evidencia la relevancia en contribuir con la investigación de herramientas para la detección de *phishing* en URLs.

1.1. Motivación

Citar surveys o compilados de fuentes de datos de phishing para apoyar la motivación en crear un corpus con mayor volumen.

1.2. Objetivos

Desarrollar un corpus y un marco de trabajo para selección y aplicación de diferentes técnicas de *Machine Learning* en el contexto de detección y prevención de *phishing* en páginas web.

Objetivos específicos

- Definir los conjuntos de datos más relevantes en el ámbito académico de la investigación en detección de páginas de *phishing* en funcionamiento actualmente.

- Construir un marco de trabajo (*framework*) para facilitar la recopilación de datos de páginas de *phishing* para uso en investigación.
- Exponer una selección de atributos para aplicar en detección basada en *Machine Learning*, basados en el estado del arte.
- Componer un conjunto de datos curado (*corpus*) mediante uso del *framework* implementado, que incluya las fuentes de datos primarias definidas y la selección de atributos planteada.
- Determinar y caracterizar las principales técnicas de detección de páginas de *phishing* basadas en *Machine Learning*, investigadas en los últimos 5 años.

1.3. Estructura del documento

2: Marco teórico y estado del arte

2.1. Acerca del *phishing*

El *phishing* es un ataque de ingeniería social [12], bajo el cual un criminal busca apropiarse de la información personal de un individuo, como su nombre completo, credenciales de usuario, información bancaria; o en ocasiones con el objetivo de diseminar contenido malicioso en una red [7]. En la Figura 1 se pueden apreciar que existen diversos tipos de ataques, consolidando al *phishing* como una metodología de ataque amplia, y susceptible a ser transmitida por diferentes medios.

Figura 2.1: Tipos de ataques de *phishing* (adaptado de [15])

Los ataques de *phishing* pueden transmitirse por diferentes medios, por ejemplo:

- SMS;
- Correos electrónicos;

- Páginas web;
- Redes sociales (𝕏, Facebook, etc.).

El tipo más común y estudiado son las páginas web *phishing* [2], esto explicado por la gran dimensión de Internet y porque este método usualmente se superpone con otros, por ejemplo, cuando hay enlaces maliciosos en el cuerpo de un SMS, *tweet* o correo electrónico. Por esta relevancia subyacente se eligió como objeto de estudio las técnicas de detección de URLs *phishing*.

2.2. Metodologías de detección de *phishing* en páginas web

Para inferir si un sitio web tiene contenido de *phishing*, se suele valer de alguno de los siguientes atributos de la página, o de una combinación de ellos [16]:

- URL;
- Contenido de la página: literal, visual, y metadatos (HTML DOM);
- Información sobre el dominio;

Los métodos basados en contenido de la página poseen varias desventajas, como complejidad en accederlo para un volumen elevado de páginas, especialmente en etapas exploratoria y de diseño; también son vulnerables a *exploits* en el navegador [8] u ofuscación de código [1] para ocultar los propósitos maliciosos. Por tanto, se concluye que se encontraría el mejor desempeño y nivel de generalización al seleccionar atributos relacionados con la URL y el dominio de las páginas.

Durante décadas de investigación, varias técnicas de detección de URLs *phishing* han sido propuestas, desde basadas en listas negras o blancas (*blacklist-based* / *whitelist-based*); basadas en heurística (*heuristic-based*); basadas en la naturaleza (*Nature Inspired - NI*); hasta basadas en aprendizaje de máquinas (*Machine Learning based - ML-based*) [3].

Dentro de estas técnicas, las *ML-based* se destacan por su desempeño superior debido a su capacidad de generalizar basado en la información provista al momento del diseño. Eventualmente los ataques se volverían más sofisticados por su capacidad de evadir reconocimiento, y de forma más notoria con la aparición de los modelos de IA generativa (*GenAI*), y su habilidad de producir contenido orgánico a la vista de un ser humano.

La aplicación de estas tecnologías de IA en la detección misma ha hecho frente a estas vulnerabilidades emergentes, y han demostrado una mayor capacidad de detección debido a su habilidad de detectar patrones más complejos, y las funcionalidades que poseen para realizar sintonización fina, siendo adaptables a situaciones específicas [13, 17]. Dado el amplio uso de las técnicas *ML-based* y su evidente relevancia en el ámbito de la investigación, se eligió profundizar en ellas de cara a la ejecución de este proyecto.

2.3. Revisión literaria

Como los resultados del proyecto planteado tienen una dependencia directa en hacer una revisión estructurada del estado del arte, se partió por encontrar un guía conciso que indicase pasos claros para reunir la información literaria previa a la ejecución. Así, basado en [4], se determinaron las etapas clave para revisión de la literatura. En la Figura

1. **Establecer el procedimiento a usar para seleccionar la el trabajo previo objeto de revisión.**

   Se planteó una revisión del estado del arte con duración de 2 semanas, con el objetivo de identificar cuáles son las técnicas principales de detección de URLs *phishing*. Esta búsqueda se realizó en las principales publicaciones a las que se tiene acceso público o mediante la Universidad de Valladolid (ACM Digital Library, IEEE Xplore, ScienceDirect y arXiv), de artículos bajo la frase clave *"phishing url detection"*.

2. **Establecer la ventana temporal que cubrirá la revisión.**

   Se estimó que se demarcaría de forma adecuada la actualidad de los trabajos a analizar si fueron publicados con antigüedad no mayor a 5 años. Durante el tiempo dedicado a la búsqueda inicial, se recopilaron cerca de 200 artículos.

3. **Definir los elementos de cada trabajo que serán examinados en la revisión.**

   De acuerdo a la conclusión de la sección 2.2, habiendo reducido el alcance en el ámbito de metodologías de detección a las *ML-based*, se filtró la lista inicial de 200 artículos, mediante revisión de las secciones de abstract y metodología, pues se consideró suficiente y óptimo en cuanto a inversión del tiempo, ya que en esos apartados se obtendría el contexto del artículo y su relevancia al proyecto. Como resultado se documentó un consolidado de 16 trabajos relacionados, cuyo contenido fue verificado que fuese acerca de al menos un tipo de técnicas aplicadas de interés en detección de URLs *phishing*, o alternativamente un artículo de revisión literaria.

2.4. Bibliotecas de manipulación de datos

El lenguaje de programación elegido para desarrollar el proyecto fue Python, debido a la disponibilidad amplia de bibliotecas de manipulación de datos y *Machine Learning*.

2.5. Acerca de convenciones sobre elaboración de código y bibliotecas

Como uno de los objetivos señalados fue la implementación de un *framework* para apoyar la creación de modelos predictores, se encontró importante asimilar conceptos de (1) mejores prácticas en elaboración de código, así como (2) documentación apropiada compatible para visibilizar la forma de uso del *framework* en menús contextuales o cuando el desarrollador ejecute comandos de ayuda sobre algún método en particular.

Sobre el primer numeral, se usó el analizador de código estático Pylint [9], para identificar de forma automática errores o divergencias al estándar de elaboración de código, en donde errores notables durante el desarrollo fueron por ejemplo en la nomenclatura de funciones, clases y variables; sentencias sin efecto en la ejecución; o mal uso de variables globales.

Respecto al segundo numeral, se tomaron como referencia el PEP 257 - Docstring Conventions [6], el cual hace parte de las *Python Enhancement Proposals* (PEP), iniciativas de mejora para Python, en este caso relacionada con definir convenciones y semántica asociada con las *docstrings* de Python. Estas *docstrings* son el encabezado de módulos, funciones, clases y métodos, los cuales sirven para documentar el propósito del código desarrollado, pero así mismo poblar las referencias mostradas al ejecutar comandos de ayuda sobre uso de elementos de una biblioteca en Python. Complementando la especificación de la PEP, se tomó como inspiración el estilo de documentación del código fuente en el proyecto NumPy [11].

Como guía de creación de bibliotecas empaquetadas y definición de dependencias, se utilizaron las especificaciones PyPA (*Python Packaging*) [14]. La revisión de este aspecto permitió la creación de un paquete estándar para publicación a usuarios, como funciona con el resto de módulos estándar o de uso masivo por la comunidad en Python.

3: Desarrollo del proyecto

4: Resultados y discusión

5: Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Apéndices

Apéndice A

Plan de Proyecto

Este apéndice presentará el plan de proyecto elaborado para la realización del trabajo. En el caso de trabajos que supongan el desarrollo de software, será sustituído por el Plan de Desarrollo de Software.

A.1. Introducción

A.2. Planificación temporal

A.3. Estudio de viabilidad

Viabilidad económica

Viabilidad legal

Apéndice B

Documentación del Programador

En este apéndice se incluye la documentación necesaria para que el programador de aplicaciones pueda comprender la estructura de la solución software aportada y para poder modificarla.

B.1. Introducción

B.2. Estructura de directorios

B.3. Manual del programador

B.4. Compilación, instalación y ejecución del proyecto

B.5. Pruebas del sistema

Bibliografía

[1] Abdelnabi, S., Krombholz, K., and Fritz, M. Visualphishnet: Zero-day phishing website detection by visual similarity. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2020), CCS '20, Association for Computing Machinery, p. 16811698.

[2] Abdillah, R., Shukur, Z., Mohd, M., and Murah, T. M. Z. Phishing classification techniques: A systematic literature review. *IEEE Access 10* (2022), 41574–41591.

[3] Akinyelu, A. A. Machine learning and nature inspired based phishing detection: A literature survey. *International Journal on Artificial Intelligence Tools 28*, 05 (2019), 1930002.

[4] Chinn, P. L. The traditional literature review. *Nurse Author & Editor 31*, 3-4 (2021), 62–64.

[5] European Union Agency for Cybersecurity. ENISA Threat Landscape. Report/Study, European Union Agency for Cybersecurity, 2023.

[6] Goodger, D., and van Rossum, G. Pep 257 - docstring conventions | peps.python.org, 2001. [Internet; accedido a 09/09/2024].

[7] Gupta, B. B., Yadav, K., Razzak, I., Psannis, K., Castiglione, A., and Chang, X. A novel approach for phishing urls detection using lexical based machine learning in a real-time environment. *Computer Communications 175* (2021), 47–57.

[8] Kim, T., Park, N., Hong, J., and Kim, S.-W. Phishing url detection: A network-based approach robust to evasion. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2022), CCS '22, Association for Computing Machinery, p. 17691782.

[9] Logilab and Pylint contributors. Pylint 3.3.0-dev0 documentation, 2024. [Internet; accedido a 16/08/2024].

[10] Moore, M. Top Cybersecurity Threats in 2023 - University of San Diego Online Degrees, 2024. [Internet; accedido a 26/04/2024].

[11] NumPy team. numpy/numpy: The fundamental package for scientific computing with python., 2024. [Internet; accedido a 09/09/2024].

[12] Paliath, S., Qbeitah, M. A., and Aldwairi, M. Phishout: Effective phishing detection using selected features. In *2020 27th International Conference on Telecommunications (ICT)* (2020), pp. 1–5.

[13] Patel, H., Rehman, U., and Iqbal, F. Evaluating the efficacy of large language models in identifying phishing attempts, 2024.

[14] PyPA. Pypa specifications - python packaging user guide, 2024. [Internet; accedido a 09/09/2024].

[15] Safi, A., and Singh, S. A systematic literature review on phishing website detection techniques. *Journal of King Saud University - Computer and Information Sciences 35*, 2 (2023), 590–611.

[16] Vijayalakshmi, M., Mercy Shalinie, S., Yang, M. H., and U., R. M. Web phishing detection techniques: a survey on the stateoftheart, taxonomy and future directions. *IET Networks 9*, 5 (9 2020), 235246.

[17] Yigit, Y., Buchanan, W. J., Tehrani, M. G., and Maglaras, L. Review of generative ai methods in cybersecurity, 2024.