Proyecto: Detector de URLs Sospechosas

Mauricio Rodriguez Duarte

Ciberseguridad virtual nivel explorador.

Resumen Ejecutivo

Este proyecto presenta el desarrollo de un Detector de URLs Sospechosas, una herramienta programada en Python diseñada para analizar una lista de URLs e identificar patrones comúnmente asociados con enlaces maliciosos, como aquellos utilizados en campañas de phishing. El programa permite al usuario ingresar URLs mediante copiado y pegado directo o cargándolas desde un archivo de texto (.txt). A través del análisis de características como la longitud de la URL, el uso de caracteres inusuales, la estructura del dominio, la presencia de palabras clave sospechosas y TLDs de baja reputación, la herramienta asigna una puntuación de riesgo y ofrece un desglose de los hallazgos. El objetivo principal es no solo detectar posibles amenazas, sino también educar a los usuarios sobre las tácticas empleadas por los ciberdelincuentes, fomentando así una mayor concienciación y precaución al navegar por internet.

Justificación

El phishing continúa siendo una de las ciberamenazas más frecuentes y efectivas a nivel mundial, provocando pérdidas económicas sustanciales y poniendo en riesgo la seguridad de información tanto personal como corporativa. Los atacantes perfeccionan constantemente sus técnicas, especialmente en la creación de URLs que imitan sitios legítimos con el fin de engañar a los usuarios y obtener credenciales, datos financieros u otra información sensible. Esta situación representa un desafío considerable, sobre todo para usuarios con conocimientos técnicos limitados, quienes muchas veces no logran identificar visualmente una URL fraudulenta.

En este contexto, se hace evidente la necesidad de herramientas accesibles y educativas que permitan a los usuarios evaluar la legitimidad de los enlaces antes de interactuar con ellos. Aunque existen soluciones de seguridad avanzadas, estas no siempre están al alcance de todos los usuarios o no promueven una comprensión activa del problema. Por ello, este proyecto propone el desarrollo de una herramienta sencilla que, además de detectar patrones sospechosos en las URLs, cumpla una función pedagógica. El objetivo es empoderar a los usuarios mediante el aprendizaje, ayudándoles a reconocer señales de alerta en los enlaces y así reducir su exposición a ataques de phishing.

Objetivos

Objetivo General

Desarrollar un programa en Python capaz de analizar URLs proporcionadas por el usuario, identificar patrones sospechosos comúnmente encontrados en enlaces maliciosos, y presentar los resultados de forma clara y educativa para ayudar a prevenir ataques de phishing.

Objetivos Específicos

Implementar la funcionalidad de entrada de URLs mediante pegado directo en la consola.

Implementar la funcionalidad de entrada de URLs mediante la carga de un archivo de texto (.txt).

Desarrollar un módulo de análisis que examine características de la URL como:

Longitud excesiva.

Presencia de caracteres no estándar o sospechosos.

Uso de direcciones IP en lugar de nombres de dominio.

Número anómalo de subdominios o guiones.

Presencia de TLDs (Top-Level Domains) de baja reputación.

Uso de palabras clave comúnmente asociadas al phishing.

Identificación de dominios de acortadores de URL.

Verificación del uso de HTTPS.

Establecer un sistema de puntuación para cuantificar el nivel de sospecha de cada URL.

Presentar al usuario un informe detallado por cada URL, incluyendo los patrones detectados y el nivel de riesgo asociado.

Proporcionar mensajes educativos que expliquen por qué ciertos patrones son considerados sospechosos.

Alcance

El programa analizará URLs basándose en características sintácticas y estructurales predefinidas.

Aceptará URLs como cadenas de texto, ya sea ingresadas directamente o leídas desde un archivo.

Proporcionará una evaluación de riesgo (Bajo, Medio, Alto) y una lista de hallazgos para cada URL.

Funcionará como una aplicación de consola (CLI).

El código fuente será modular y comentado para facilitar su comprensión y posible extensión.

Limitaciones

El detector no realizará análisis de contenido de la página web destino (ej. no descargará ni analizará HTML/JavaScript).

No se integrará con APIs externas de listas negras en tiempo real (ej. VirusTotal, Google Safe Browsing) en esta fase inicial.

La detección se basa en heurísticas y patrones conocidos; no es infalible y podría generar falsos positivos o falsos negativos, especialmente con técnicas de phishing muy sofisticadas o novedosas.

No garantiza al 100% la seguridad o malicia de una URL, sino que ofrece una evaluación de riesgo basada en los criterios implementados.

No analiza la reputación histórica del dominio ni realiza búsquedas WHOIS.

Herramientas y Tecnologías

Lenguaje de Programación: Python 3.x.

Librerías Estándar de Python:

re: Para el uso de expresiones regulares en la identificación de patrones.

urllib.parse: Para descomponer y analizar los componentes de una URL.

ipaddress: Para validar si un hostname es una dirección IP.

Entorno de Desarrollo: Cualquier editor de texto o IDE compatible con Python (ej. VS Code, PyCharm, Sublime Text).

Criterios de Detección

El núcleo del detector se basa en un conjunto de reglas y heurísticas definidas a partir de características comunes en URLs maliciosas:

Longitud de la URL: URLs excesivamente largas.

Esquema: Preferencia por HTTPS, alerta ante HTTP para sitios sensibles.

Dominio vs. IP: Uso de direcciones IP como hostname.

Caracteres Especiales: Presencia del símbolo '@' en el hostname, caracteres no ASCII.

Estructura del Dominio: Número excesivo de subdominios o guiones.

TLDs: Comparación con una lista de TLDs de alto riesgo.

Palabras Clave: Búsqueda de términos comunes en phishing (login, bank, verify, etc.).

Acortadores de URL: Identificación de dominios conocidos de servicios de acortamiento.

Ofuscación: Patrones como doble barra ("//") en la ruta.

Características Principales del Programa

Entrada Flexible de URLs: Permite al usuario ingresar URLs individualmente, pegar una lista o cargar un archivo .txt.

Análisis Multifactorial: Evalúa múltiples aspectos de la URL para determinar su potencial riesgo.

Sistema de Puntuación: Asigna una puntuación numérica que refleja el grado de sospecha.

Niveles de Riesgo Claros: Clasifica las URLs en niveles de sospecha (Bajo, Medio, Alto) para una fácil interpretación.

Informes Detallados: Para cada URL, enumera los patrones específicos que activaron las alertas.

Feedback Educativo: Los mensajes de alerta explican por qué un patrón particular es preocupante, ayudando al usuario a aprender.

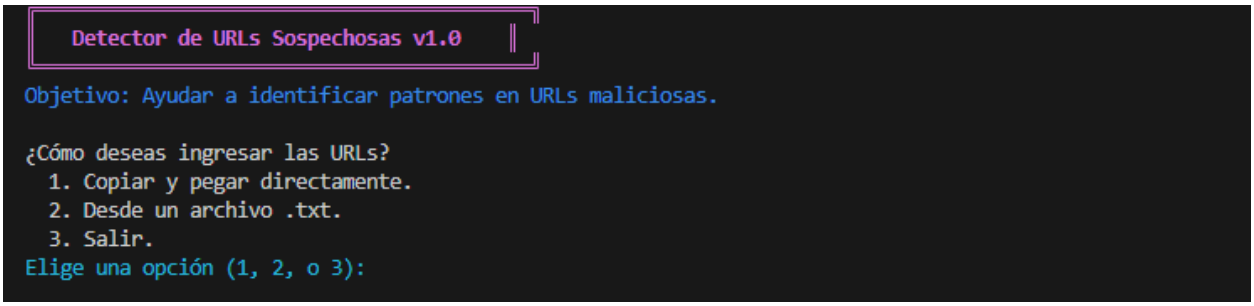
Interfaz de Línea de Comandos (CLI): Sencilla e intuitiva para la interacción con el usuario.

Portabilidad: Al estar escrito en Python y usar principalmente librerías estándar, es fácilmente ejecutable en diferentes sistemas operativos.

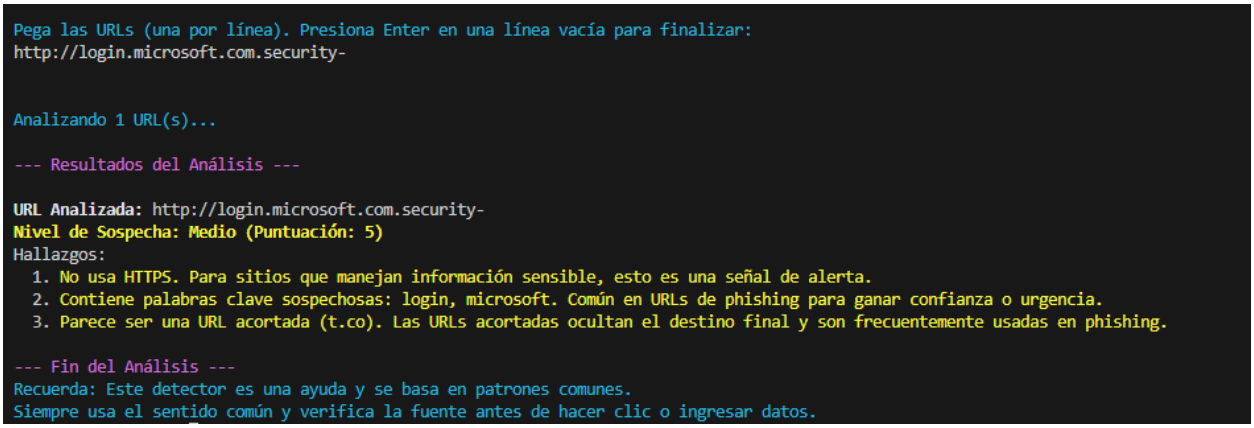
Personalizable: Las constantes (como MAX_URL_LENGTH, SUSPICIOUS_TLDS) pueden ser ajustadas o ampliadas fácilmente.

Demostración de Uso y Resultados Esperados

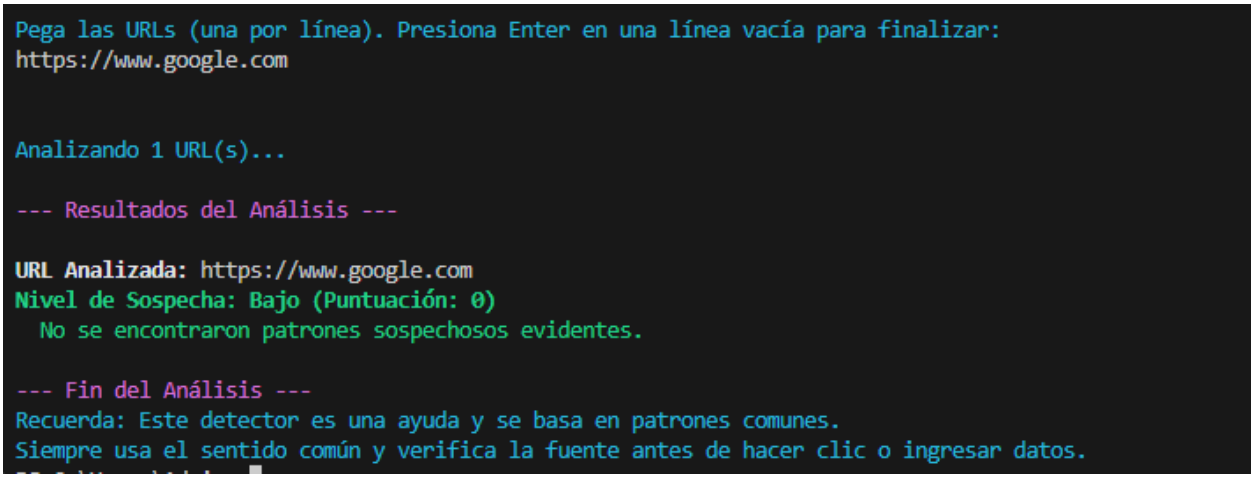
Al ejecutar el programa, el usuario será recibido con un menú para elegir el método de entrada de URLs:



Si el usuario elige "1", podrá pegar las URLs. Si elige "2", se le pedirá la ruta del archivo. Tras el análisis, se mostrarán los resultados para cada URL. Ejemplo de salida para una URL sospechosa:



Para una URL aparentemente segura:



Código Completo

```
import re
from urllib.parse import urlparse
import ipaddress

MAX_URL_LENGTH = 100
SUSPICIOUS_TLDS = ['.xyz', '.top', '.tk', '.link', '.club', '.online', '.site', '.pw', '.biz', '.info', '.ws', '.cc']
COMMON_PHISHING_KEYWORDS = ['login', 'signin', 'account', 'secure', 'verify', 'update', 'password', 'bank', 'confirm', 'support', 'appleid', 'microsoft']
SHORTENER_DOMAINS = ['bit.ly', 'tinyurl.com', 'goo.gl', 't.co', 'is.gd', 'ow.ly', 'buff.ly', 'clck.ru']
MAX_SUBDOMAINS = 4
MAX_HYPHENS_IN_DOMAIN = 3
```

```
class bcolors:
    HEADER = '\033[95m'
    OKBLUE = '\033[94m'
    OKCYAN = '\033[96m'
    OKGREEN = '\033[92m'
    WARNING = '\033[93m'
    FAIL = '\033[91m'
    ENDC = '\033[0m'
    BOLD = '\033[1m'
    UNDERLINE = '\033[4m'


def analyze_url(url):
    findings = []
    score = 0
    original_url  = url

    if not re.match(r'^[a-zA-Z]+://',  url):
```

```
            url = "http://" + url

        try:
            parsed_url = urlparse(url)
            hostname = parsed_url.hostname if parsed_url.hostname else ""
            path = parsed_url.path
            scheme = parsed_url.scheme
        except ValueError:
            findings.append("URL mal formada o inválida.")
            return {"url": original_url, "score": 10, "level": "Alto", "findings": findings,
"parsed": None}

        if len(original_url) > MAX_URL_LENGTH:
            findings.append(f"URL demasiado larga ({len(original_url)} caracteres, máx.
sugerido {MAX_URL_LENGTH}). Las URLs largas pueden ocultar el verdadero dominio o
parámetros maliciosos.")
            score += 2

        if scheme != 'https':
```

```
            findings.append("No usa HTTPS. Para sitios que manejan información sensible, esto es una señal de alerta.")
            score += 1

        if hostname:
            try:
                ipaddress.ip_address(hostname)
                findings.append("El dominio es una dirección IP. Muy sospechoso para servicios web legítimos (e.g., bancos, redes sociales).")
                score += 5
            except ValueError:
                pass

        if '@' in parsed_url.netloc and '@' in hostname:
            if parsed_url.username:
                findings.append("Contiene '@' en el nombre de dominio/autoridad. Táctica común de phishing para ofuscar el dominio real.")
                score += 5
```

```
        if hostname and hostname.count('.') >= MAX_SUBDOMAINS:
            findings.append(f"Número excesivo de subdominios ({hostname.count('.')}). Podría intentar ofuscar el dominio real o imitar marcas conocidas (e.g., login.microsoft.com.security-update.com).")
            score += 3

        if hostname:
            for tld in SUSPICIOUS_TLDS:
                if hostname.lower().endswith(tld):
                    findings.append(f"Usa un TLD ('{tld}') a menudo asociado con spam, malware o campañas de phishing de corta duración.")
                    score += 3
                    break

        if hostname and any(ord(char) > 127 for char in hostname):
            findings.append("Caracteres no ASCII (extraños) en el nombre de dominio. Podría ser un intento de ataque homógrafo.")
            score += 4
        if any(ord(char) > 127 for char in path):
```

```
            findings.append("Caracteres no ASCII (extraños) en la ruta de la URL.")
            score += 2

        combined_domain_path = (hostname.lower() if hostname else "") + path.lower()
        found_keywords = []
        for keyword in COMMON_PHISHING_KEYWORDS:
            if keyword in combined_domain_path:
                found_keywords.append(keyword)
        if found_keywords:
            findings.append(f"Contiene palabras clave sospechosas: {', '.join(found_keywords)}. Común en URLs de phishing para ganar confianza o urgencia.")
            score += 1

        if hostname:
            for shortener in SHORTENER_DOMAINS:
                if shortener in hostname.lower():
                    findings.append(f"Parece ser una URL acortada ({shortener}). Las URLs acortadas ocultan el destino final y son frecuentemente usadas en phishing.")
                    score += 3
```

```
                break


        if '//' in path:
            findings.append("Doble barra ('//') en la ruta de la URL (después del 'http://' o 'https://'). Podría ser un intento de ofuscación o evasión de filtros.")
            score += 2


        if hostname and hostname.count('-') > MAX_HYPHENS_IN_DOMAIN:
            findings.append(f"Excesivos guiones ('-') en el nombre de dominio ({hostname.count('-')}). A veces usado para imitar marcas (e.g., 'my-bank-online-security.com').")
            score += 2


        if parsed_url.username or parsed_url.password:
            findings.append("La URL contiene información de usuario/contraseña directamente. Práctica insegura y obsoleta, a veces vista en URLs maliciosas.")
            score += 3


        level = "Bajo"
```

```
    color = bcolors.OKGREEN
    if score >= 8:
        level = "Alto"
        color = bcolors.FAIL
    elif score >= 4:
        level = "Medio"
        color = bcolors.WARNING

    return {
        "url": original_url,
        "score": score,
        "level": level,
        "findings": findings,
        "color": color,
        "parsed": parsed_url
    }

def get_urls_from_input():
```

```
        print(f"\n{bcolors.OKCYAN}Pega las URLs (una por línea). Presiona Enter en una línea vacía para finalizar:{bcolors.ENDC}")
        urls = []
        while True:
            try:
                line = input()
                if not line.strip():
                    if not urls:
                        print(f"{bcolors.WARNING}No se ingresaron URLs. Intenta de nuevo o elige otra opción.{bcolors.ENDC}")
                        continue
                    break
                urls.append(line.strip())
            except EOFError:
                break
        return urls

    def get_urls_from_file():
```

```
        filepath = input(f"\n{bcolors.OKCYAN}Introduce la ruta al archivo .txt con las URLs:
{bcolors.ENDC}").strip()
        urls = []
        if not filepath:
            print(f"{bcolors.WARNING}No se especificó una ruta de
archivo.{bcolors.ENDC}")
            return []
        try:
            with open(filepath, 'r', encoding='utf-8') as f:
                for line in f:
                    stripped_line = line.strip()
                    if stripped_line and not stripped_line.startswith("#"):
                        urls.append(stripped_line)
            if not urls:
                print(f"{bcolors.WARNING}El archivo '{filepath}' está vacío o no contiene
URLs válidas.{bcolors.ENDC}")
        except FileNotFoundError:
            print(f"{bcolors.FAIL}Error: Archivo '{filepath}' no encontrado.{bcolors.ENDC}")
        except Exception as e:
```

```
            print(f"{bcolors.FAIL}Error al leer el archivo '{filepath}': {e}{bcolors.ENDC}")
        return urls

    def display_results(results):
        if not results:
            print(f"{bcolors.WARNING}No se analizaron URLs.{bcolors.ENDC}")
            return

        print(f"\n{bcolors.HEADER}--- Resultados del Análisis ---{bcolors.ENDC}")
        for result in results:
            print(f"\n{bcolors.BOLD}URL Analizada:{bcolors.ENDC} {result['url']}")
            print(f"{result['color']}{bcolors.BOLD}Nivel de Sospecha: {result['level']}
(Puntuación: {result['score']}){bcolors.ENDC}")

            if result['findings']:
                print(f"{bcolors.UNDERLINE}Hallazgos:{bcolors.ENDC}")
                for i, finding in enumerate(result['findings'], 1):
                    print(f"  {i}. {result['color']}{finding}{bcolors.ENDC}")
            else:
```

```
            print(f"{bcolors.OKGREEN}  No se encontraron patrones sospechosos evidentes.{bcolors.ENDC}")

        print(f"\n{bcolors.HEADER}--- Fin del Análisis ---{bcolors.ENDC}")
        print(f"{bcolors.OKCYAN}Recuerda: Este detector es una ayuda y se basa en patrones comunes.{bcolors.ENDC}")
        print(f"{bcolors.OKCYAN}Siempre usa el sentido común y verifica la fuente antes de hacer clic o ingresar datos.{bcolors.ENDC}")

    def main():
        print(f"{bcolors.HEADER}{bcolors.BOLD}╔═══════════════════════════════════════════╗")
        print(f"║   Detector de URLs Sospechosas v1.0   ║")
        print(f"╚═══════════════════════════════════════════╝{bcolors.ENDC}")
        print(f"{bcolors.OKBLUE}Objetivo: Ayudar a identificar patrones en URLs maliciosas.{bcolors.ENDC}")

        urls_to_analyze = []
        while True:
```

```
        print("\n¿Cómo deseas ingresar las URLs?")
        print("  1. Copiar y pegar directamente.")
        print("  2. Desde un archivo .txt.")
        print("  3. Salir.")
        choice = input(f"{bcolors.OKCYAN}Elige una opción (1, 2, o 3): {bcolors.ENDC}").strip()

        if choice == '1':
            urls_to_analyze = get_urls_from_input()
            break
        elif choice == '2':
            urls_to_analyze = get_urls_from_file()
            break
        elif choice == '3':
            print(f"{bcolors.OKBLUE}Saliendo del programa. ¡Mantente seguro online!{bcolors.ENDC}")
            return
        else:
```

```
                print(f"{bcolors.FAIL}Opción no válida. Por favor, elige 1, 2 o
3.{bcolors.ENDC}")

        if urls_to_analyze:
            print(f"\n{bcolors.OKCYAN}Analizando {len(urls_to_analyze)}
URL(s)...{bcolors.ENDC}")
            all_results = []
            for url in urls_to_analyze:
                if url:
                    analysis_result = analyze_url(url)
                    all_results.append(analysis_result)
            display_results(all_results)
        elif choice != '3':
            print(f"{bcolors.WARNING}No se proporcionaron URLs para
analizar.{bcolors.ENDC}")

    if __name__ == "__main__":
        main()
```