



PUESTA EN PRODUCCION SEGURA

Unidad 1. Trabajo



23 DE NOVIEMBRE DE 2023
CARLOS DÍAZ MONTES
ESPECIALIZACIÓN DE CIBERSEGURIDAD

Índice

URLLIB.....	2
¿Qué es URLLIB?	2
Como instalar urllib	2
¿Para qué sirve cada submódulo de URLLIB?	2
Ejemplos de usos.....	3

URLLIB

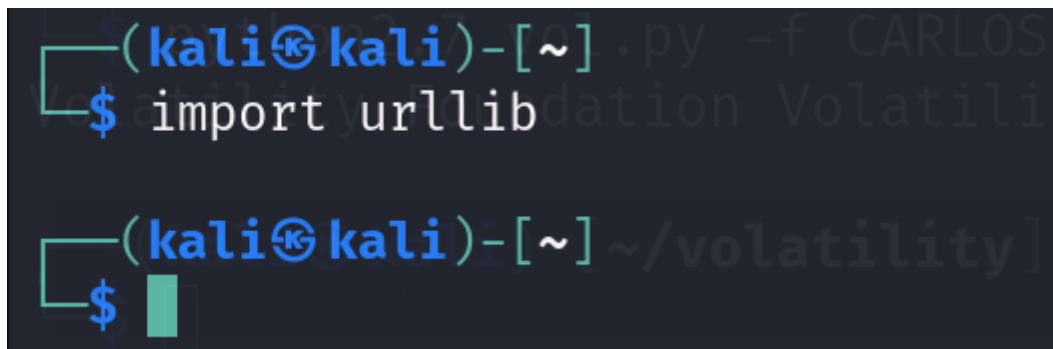
¿Qué es URLLIB?

El módulo `urllib` en Python es un conjunto de módulos que proporciona un conjunto de herramientas para trabajar con URL.

Como instalar urllib

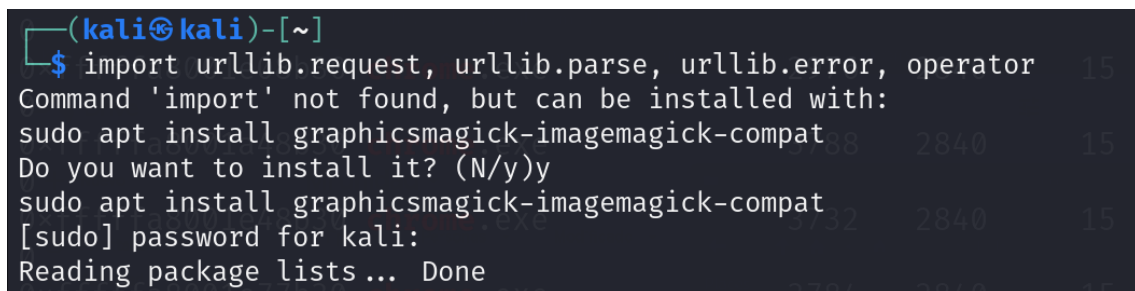
El módulo `urllib` es parte de la biblioteca estándar de Python, por lo que generalmente no es necesario instalarlo por separado. Sin embargo, si estás utilizando una versión de Python más antigua (como Python 2.x), es posible que ya esté instalado. En caso de que estés utilizando Python 3, el módulo debería estar disponible de forma predeterminada.

Para comprobar si `urllib` está disponible, simplemente intenta importarlo en tu script o en la consola de Python:



```
(kali@kali)-[~].py -f CARLOS
$ import urllib
(kali@kali)-[~] ~/volatility
$
```

Ahora lo que debes de importar son los submódulos :



```
(kali@kali)-[~]
$ import urllib.request, urllib.parse, urllib.error, operator
Command 'import' not found, but can be installed with:
sudo apt install graphicsmagick-imagemagick-compat
Do you want to install it? (N/y)y
sudo apt install graphicsmagick-imagemagick-compat
[sudo] password for kali:
Reading package lists... Done
```

¿Para qué sirve cada submódulo de URLLIB?

- **`urllib.request`:** Permite abrir y leer URLs. Con este módulo, puedes realizar solicitudes HTTP básicas, como GET y POST, y gestionar las respuestas del servidor.
- **`urllib.parse`:** Proporciona funciones para analizar (parsear) URLs, construir URLs y manipular sus componentes. Es útil para dividir una URL en partes y viceversa.
- **`urllib.error`:** Contiene excepciones específicas para manejar errores relacionados con operaciones de URL, como errores de red, errores HTTP, etc.
- **`urllib.robotparser`:** Ayuda a analizar el archivo `robots.txt` de un sitio web para determinar si ciertos agentes web pueden o no acceder a ciertas partes del sitio.

Ejemplos de usos

Urllib.request

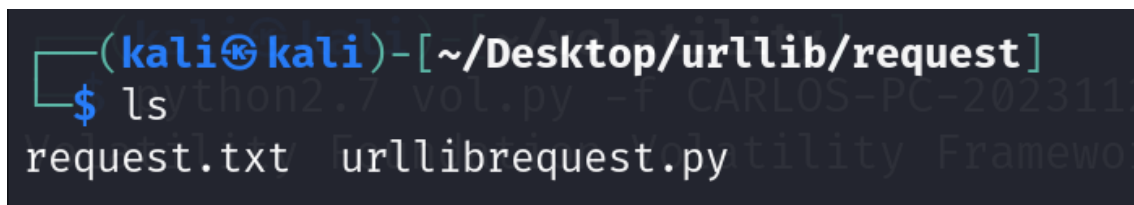
Vamos a realizar una solicitud HTTP básica:

```
GNU nano 7.2 urllibrequest.py *
# Importa la función urlopen del módulo urllib.request.
from urllib.request import urlopen

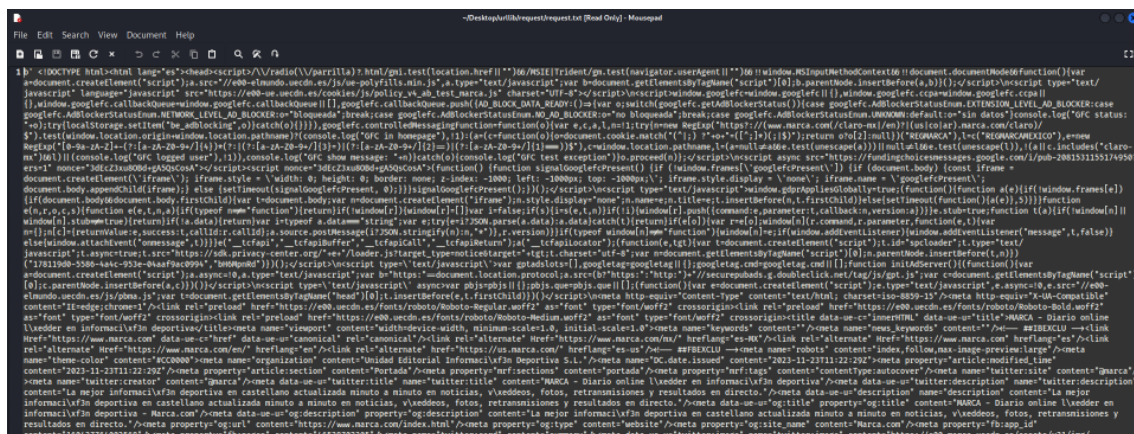
# Seleccionamos la pagina que queramos
url = 'https://www.marca.com'
# Realiza la solicitud a la URL y almacena la respuesta en la variable response
response = urlopen(url)
# Lee el contenido de la respuesta y lo almacena en la variable html.
html = response.read()

# Guardar el contenido en request.txt
with open('request.txt', 'w', encoding='utf-8') as file_request:
    file_request.write(str(html))
```

Ahora comprobamos que nos ha devuelto un archivo llamado request.txt con el contenido html de la pagina:



El contenido de la pagina:



urllib.parse

Analizamos y construimos URLs:

```
GNU nano 7.2 urlparse.py
# Importa las funciones urlparse y urlencode del módulo urllib.parse.
from urllib.parse import urlparse, urlencode

# URL que se va a analizar
url = 'http://www.edu4java.com/es/web/web30.html'

# Analizar la URL y obtener sus componentes
parsed_url = urlparse(url)

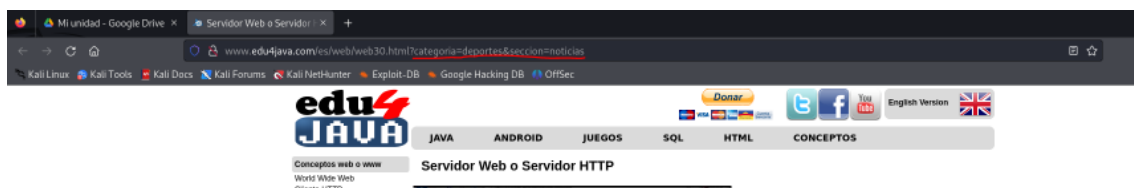
# Imprimir los componentes de la URL
print('Esquema:', parsed_url.scheme)
print('Netloc:', parsed_url.netloc)
print('Ruta:', parsed_url.path)

# Construir una nueva URL con parámetros
parametros = {'categoria': 'deportes', 'seccion': 'noticias'}
url_nueva = url + '?' + urlencode(parametros)

# Imprimir la URL construida
print('URL construida:', url_nueva)
```

Como resultado:

```
(kali@kali)-[~/Desktop/urllib/parse]
$ sudo python3 urlparse.py
Esquema: http
Netloc: www.edu4java.com
Ruta: /es/web/web30.html
URL construida: http://www.edu4java.com/es/web/web30.html?categoria=deportes&seccion=noticias
```



urllib.error

Manejamos errores de URL, por ejemplo vamos a buscar un error en Google.es (no vamos a encontrar ninguno claramente):

```
# Importa las clases HTTPError y URLError del módulo urllib.error.
from urllib.error import HTTPError, URLError
# Importa la función urlopen del módulo urllib.request.
from urllib.request import urlopen

# Intentar realizar una solicitud a la página de inicio de Google
try:
    response = urlopen('https://www.google.com')
# Capturar excepciones específicas para manejar errores HTTP y de URL
except HTTPError as e:
    # Imprimir el error HTTP
    print('Error HTTP:', e.code, e.reason)
# Guardar el error en error.txt
    with open('error.txt', 'w', encoding='utf-8') as file_error:
        file_error.write(f'Error HTTP: {e.code}, {e.reason}\n')
except URLError as e:
    # Imprimir el error de URL
    print('Error de URL:', e.reason)
# Guardar el error en error.txt
    with open('error.txt', 'w', encoding='utf-8') as file_error:
        file_error.write(f'Error de URL: {e.reason}\n')
```

Ahora vamos a buscar errores en una página inventada por ejemplo:

```
GNU nano 7.2 urlerror.py *
# Importa las clases HTTPError y URLError del módulo urllib.error.
from urllib.error import HTTPError, URLError
# Importa la función urlopen del módulo urllib.request.
from urllib.request import urlopen

# Intentar realizar una solicitud a la página de inicio de Google
try:
    response = urlopen('https://www.pruebapaginanoexiste.com')
# Capturar excepciones específicas para manejar errores HTTP y de URL
except HTTPError as e:
    # Imprimir el error HTTP
    print('Error HTTP:', e.code, e.reason)
# Guardar el error en error.txt
    with open('error.txt', 'w', encoding='utf-8') as file_error:
        file_error.write(f'Error HTTP: {e.code}, {e.reason}\n')
except URLError as e:
    # Imprimir el error de URL
    print('Error de URL:', e.reason)
# Guardar el error en error.txt
    with open('error.txt', 'w', encoding='utf-8') as file_error:
        file_error.write(f'Error de URL: {e.reason}\n')
```

Y ahora vemos el resultado del error:

```
(kali㉿kali)-[~/Desktop/urllib/error]
$ sudo python3 urlerror.py
Error de URL: [Errno -2] Name or service not known
```

urllib.robotparser

El archivo robots.txt es utilizado por los propietarios de sitios web para comunicar a los motores de búsqueda y otros agentes web qué partes de su sitio web pueden o no ser rastreadas. Analizamos el archivo robots.txt:

```
GNU nano 7.2 urlrobot.py
from urllib.robotparser import RobotFileParser

# Crear una instancia de RobotFileParser
rp = RobotFileParser()

# Establecer la URL del archivo robots.txt a analizar
rp.set_url("http://www.musi-cal.com/robots.txt")

# Leer y analizar el archivo robots.txt
rp.read()

# Obtener la tasa de solicitudes permitidas para el agente "*"
rrate = rp.request_rate("*")

# Imprimir la información sobre la tasa de solicitudes permitidas
if rrate:
    print(f"Tasa de solicitudes permitidas: {rrate.requests} solicitudes cada {rrate.seconds} segundos")
else:
    print("No hay información de tasa de solicitudes para el agente '*')

# Obtener el retraso de rastreo (crawl delay) para el agente "*"
crawl_delay = rp.crawl_delay("*")
print(f"Retraso de rastreo (crawl delay) para el agente '*': {crawl_delay} segundos")

# Verificar si el agente '*' puede acceder a la URL especificada
can_fetch_search = rp.can_fetch("*", "http://www.musi-cal.com/cgi-bin/search?city=San+Francisco")
print(f"¿El agente '*' puede acceder a la búsqueda? {can_fetch_search}")

# Verificar si el agente '*' puede acceder a la página principal
can_fetch_home = rp.can_fetch("*", "http://www.musi-cal.com/")
print(f"¿El agente '*' puede acceder a la página principal? {can_fetch_home}")
```

Aquí podemos ver el resultado:

```
(kali㉿kali)-[~/Desktop/urllib/robot]
$ sudo python3 urlrobot.py
No hay información de tasa de solicitudes para el agente '*'
Retraso de rastreo (crawl delay) para el agente '*': None segundos
¿El agente '*' puede acceder a la búsqueda? True
¿El agente '*' puede acceder a la página principal? True
```

Si la función `can_fetch(useragent, url)` devuelve `True`, significa que, según las reglas establecidas en el archivo `robots.txt`, el agente de usuario especificado (por ejemplo, un motor de búsqueda) tiene permiso para rastrear la URL proporcionada. En otras palabras, la página puede ser rastreada por ese agente de usuario.

Si la función devuelve `False`, significa que el acceso está prohibido y el agente de usuario especificado no tiene permiso para rastrear la URL. En este caso, la página no debería ser rastreada por ese agente según las reglas del archivo `robots.txt`.