**Informe sobre Métodos para Obtener Endpoints y Automatizar Formulario**

**Métodos Utilizados:**

1. Inspección de Red:

Se realizó un monitoreo de la red utilizando herramientas de inspección para capturar las solicitudes realizadas durante el llenado manual del formulario. Aunque se identificaron solicitudes POST, no se obtuvo información detallada sobre los endpoints involucrados. Esta técnica demostró limitaciones debido a la falta de visibilidad completa de los endpoints en el tráfico de red.

2. Análisis de Código Fuente:

Se examinó el código fuente de la página web en busca de referencias a endpoints utilizados en la automatización de formularios. Aunque se identificaron elementos `<form>` y atributos `action`, no se encontraron indicios claros sobre los endpoints requeridos. La falta de comentarios o identificadores explícitos en el código dificultó la identificación de los endpoints.

Sin embargo se obtuvo algunos endpoints de este tipo

[**https://performancemanager8.successfactors.com/ec/ajax/remoting/call/plaincall/hrisHireControllerProxy.getNewPage.dwr**](https://performancemanager8.successfactors.com/ec/ajax/remoting/call/plaincall/hrisHireControllerProxy.getNewPage.dwr)

claramente es un endpoint .dwr (direct web rwmoting) que permite comunicación remota entre el cliente y el servidor en aplicaciones web. El gran desafio es que tiene medidas de seguridad como autenticación y desconocemos la estructura que espera por recibir en el formato tipo JSON.

Los script presentados se crearon con el propósito de obtener la autenticación y que nos permitiera realizar peticiones tipo POST desde la web creada para la automatización, se ejecutaron pruebas pero ninguna con éxito, puede ser por parte de protocolos de autenticación mas severos o que la estructura del mismo no sea compatible.

Por otro lado se desconoce si esta autenticación se envía mediante el header, que en este se puede incluir un token del que también se desconoce.

3. Herramientas de Desarrollo:

Se utilizaron herramientas de desarrollo web para inspeccionar las solicitudes generadas durante la interacción manual con el formulario. Sin embargo, la información sobre los endpoints seguía siendo elusiva. A pesar de capturar las solicitudes POST, la información específica de los endpoints no estaba directamente visible en los encabezados o detalles de las solicitudes.

4. Análisis de la Página Web y API:

Se intentó explorar la documentación del sitio web en busca de información sobre las API utilizadas y los endpoints disponibles. Sin embargo, la falta de documentación detallada limitó el éxito de esta estrategia. La ausencia de información clara sobre endpoints relevantes en la documentación dificultó su identificación.

5. Ingeniería Inversa de Scripts de Página:

Se examinaron los scripts y archivos JavaScript en el código fuente de la página en busca de pistas sobre la construcción de las solicitudes y los endpoints asociados. Lamentablemente, esto tampoco proporcionó resultados concluyentes. La ofuscación o compresión de los scripts dificultó la identificación precisa de los endpoints.

Para demostrar un enfoque de automatización, se proporcionó un script de Python. El script utiliza las bibliotecas `requests`, `BeautifulSoup` y `mysql.connector` para automatizar el inicio de sesión en un sitio web y el llenado de un formulario con datos de una base de datos. El script primero se conecta a la base de datos, recupera los registros de usuarios y utiliza una sesión para iniciar sesión en el sitio web. Luego, analiza el contenido HTML de la página del formulario para identificar los campos necesarios. Finalmente, el script envía solicitudes POST a los endpoints correspondientes, simulando el llenado del formulario.

Primer escenario con BeautifulSoup

import requests

from bs4 import BeautifulSoup

import mysql.connector

# *Configuración de la base de datos y del sitio web*

db\_config = {

    'user': 'root',

    'password': '123456',

    'host': 'localhost',

    'database': 'dbp\_bot\_claro'

}

login\_url = "https://performancemanager8.successfactors.com/login?bplte\_logout=1&company=comunicaci&\_s.crb=qgk6zncK2k2t5O5whCyOPdDx%252bN%252bLtYKNoAeX4UydIq0%253d#/login"

form\_page\_url = "https://performancemanager8.successfactors.com/xi/ui/peopleprofile/pages/newhire.xhtml?&\_s.crb=Cw%2f%2b8spZ9XJlF3ySGzjyPHAGCIo0Vy3h7S8%2fdPwP1Mg%3d"

# *Conexión a la base de datos*

conn = mysql.connector.connect(\*\*db\_config)

cursor = conn.cursor()

# *Recuperar datos de usuarios de la base de datos*

cursor.execute("SELECT nombre, password FROM usuarios")

usuarios = cursor.fetchall()

# *Automatizar el inicio de sesión*

session = requests.Session()

login\_payload = {

    'username': 'EC2454G',

    'password': 'Holanda20\*'

}

session.post(login\_url, data=login\_payload)

# *Obtener el contenido HTML de la página del formulario*

response = session.get(form\_page\_url)

soup = BeautifulSoup(response.content, 'html.parser')

# *Identificar el formulario y los campos*

form = soup.find('form')

nombre\_input = form.find('input', {'name': 'username'})

# *email\_input = form.find('input', {'name': 'email'})*

password\_input = form.find('input', {'name': 'password'})

# *Automatizar el acceso al formulario y envío de datos*

for usuario in usuarios:

    nombre, email, password = usuario

    payload = {

        'nombre': nombre,

        # *'email': email,*

        'password': password

    }

    # *Construir la URL del endpoint basado en la acción del formulario*

    endpoint\_url = form.get('action')

    response = session.post(endpoint\_url, data=payload)

    if response.status\_code == 200:

        print(f"Usuario {nombre} registrado exitosamente.")

    else:

        print(f"Error al registrar usuario {nombre}.")

# *Cerrar conexiones*

cursor.close()

conn.close()

segundo escenario   
  
  
import requests

import mysql.connector

# *Configuración de la base de datos*

db\_config = {

    'user': 'usuario',

    'password': 'contraseña',

    'host': 'localhost',

    'database': 'basededatos'

}

# *Configuración del sitio web*

login\_url = "URL\_DE\_INICIO\_DE\_SESION"

form\_url = "URL\_DEL\_FORMULARIO"

# *Conexión a la base de datos*

conn = mysql.connector.connect(\*\*db\_config)

cursor = conn.cursor()

# *Recuperar datos de usuarios de la base de datos*

cursor.execute("SELECT nombre, email, password FROM usuarios")

usuarios = cursor.fetchall()

# *Automatizar el inicio de sesión*

session = requests.Session()

login\_payload = {

    'username': 'tu\_usuario',

    'password': 'tu\_contraseña'

}

session.post(login\_url, data=login\_payload)

# *Automatizar el acceso al formulario y envío de datos*

for usuario in usuarios:

    nombre, email, password = usuario

    payload = {

        'nombre': nombre,

        'email': email,

        'password': password

    }

    response = session.post(form\_url, data=payload)

    if response.status\_code == 200:

        print(f"Usuario {nombre} registrado exitosamente.")

    else:

        print(f"Error al registrar usuario {nombre}.")

# *Cerrar conexiones*

cursor.close()

conn.close()

Conclucione

la obtención de endpoints relevantes para la interacción con el backend puede ser un desafío. Aunque se intentaron varios métodos, la falta de información clara y explícita en el código fuente y la documentación puede dificultar la identificación precisa de los endpoints y estructura de la petición. La ingeniería inversa de scripts y el monitoreo de red pueden proporcionar algunas pistas, pero es importante tener en cuenta que no siempre garantizan resultados completos.