Simulación de un ataque Man-in-the-Middle (MitM) en Python

Ejercicio: Simulación de un ataque Man-in-the-Middle (MitM)

En este ejercicio, vamos a simular un ataque Man-in-the-Middle (MitM) donde un proceso intercepta y modifica la comunicación entre dos otros procesos. Utilizaremos la comunicación bidireccional entre procesos en Python para realizar esta simulación.

```
import subprocess

def mitm_ataque():
    # Primer proceso: simula un cliente enviando datos
    proceso1 = subprocess.Popen(["echo", "Mensaje secreto del cliente"], stdout=subprocess.PIPE)
    salida1 = proceso1.communicate()[0]

    # Proceso intermedio: el atacante intercepta y modifica el mensaje
    proceso_intermedio = subprocess.Popen(["sed", "s/secreto/modificado/"], stdin=subprocess.PIPE, stdout=subprocess.PIPE)
    salida_intermedia = proceso_intermedio.communicate(input=salida1)[0]

    # Segundo proceso: simula un servidor recibiendo el mensaje
```

```
            proceso2   =   subprocess.Popen(["cat"],
stdin=subprocess.PIPE, stdout=subprocess.PIPE)
    salida2 = proceso2.communicate(input=salida_intermedia)[0]

    print("Mensaje interceptado y modificado:")
    print(salida2.decode())

mitm_ataque()
```

Explicación

1. **Primer proceso: simula un cliente enviando datos**

```
proceso1 = subprocess.Popen(["echo", "Mensaje secreto del
cliente"], stdout=subprocess.PIPE)
salida1 = proceso1.communicate()[0]
```

Este primer proceso simula un cliente enviando un mensaje secreto. El comando echo genera el mensaje «Mensaje secreto del cliente» y lo redirige a la salida estándar.

2. **Proceso intermedio: el atacante intercepta y modifica el mensaje**

```
proceso_intermedio    =    subprocess.Popen(["sed",
"s/secreto/modificado/"],      stdin=subprocess.PIPE,
stdout=subprocess.PIPE)
salida_intermedia                                    =
proceso_intermedio.communicate(input=salida1)[0]
```

El proceso intermedio simula al atacante que intercepta el mensaje del cliente. Utiliza el comando sed para buscar y reemplazar la palabra «secreto» con «modificado». La entrada estándar de este proceso es la salida del primer proceso.

3. **Segundo proceso: simula un servidor recibiendo el mensaje**

```
proceso2 = subprocess.Popen(["cat"], stdin=subprocess.PIPE,
```

```
stdout=subprocess.PIPE)
salida2 = proceso2.communicate(input=salida_intermedia)[0]
```

El segundo proceso simula un servidor que recibe el mensaje modificado por el atacante. Utiliza el comando cat para mostrar el mensaje recibido.

4. **Mostrar el mensaje interceptado y modificado**

```
print("Mensaje interceptado y modificado:")
print(salida2.decode())
```

Finalmente, el programa imprime el mensaje que ha sido interceptado y modificado, mostrando el resultado final del ataque MitM.

Resumen

Este ejercicio demuestra cómo un atacante puede interceptar y modificar la comunicación entre dos procesos. Utilizando la comunicación bidireccional entre procesos en Python, hemos simulado un ataque Man-in-the-Middle en el que un mensaje secreto enviado por un cliente es interceptado y modificado por un atacante antes de ser recibido por un servidor.