Universidad de Sevilla

Grado en Ingeniería Informática – Ingeniería de Computadores

Seguridad en Sistemas Informáticos y en Internet

PRÁCTICA 1

Grupo 2

Alumnos: Silvia Castillo Ruiz, Amara Innocent Millán y Víctor Ramos Lara

Índice

1. Resumen Ejecutivo

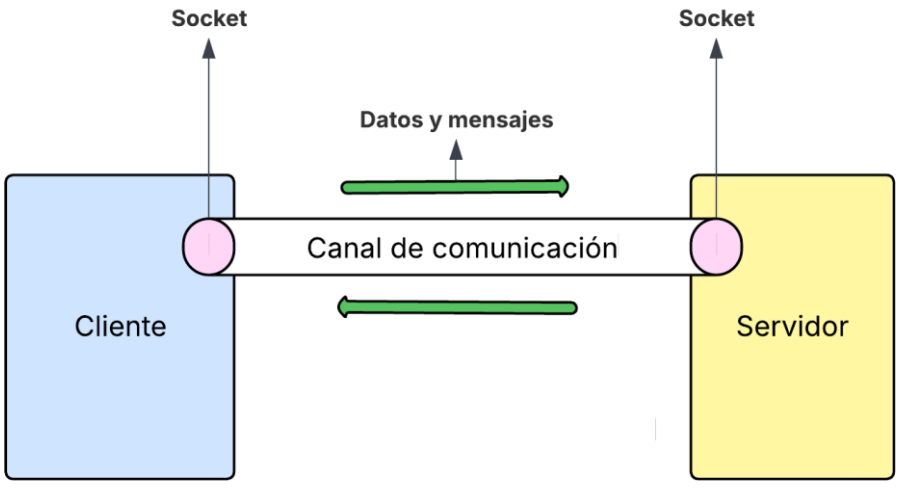
En el presente proyecto hemos desarrollado una solución orientada a garantizar la integridad en el almacenamiento y transmisión de datos para una entidad financiera que ofrece servicios de transferencia a través de una arquitectura cliente-servidor basada en sockets. El sistema implementado permite el registro, autenticación y gestión de usuarios mediante credenciales, así como la realización de transacciones en un formato definido, preservando en todo momento la seguridad de la información.

Para dar cumplimiento a las políticas de seguridad planteadas por la entidad, hemos diseñado mecanismos que aseguran tanto la integridad de las credenciales almacenadas como la integridad de las comunicaciones en un entorno de red pública. Con este objetivo, el proyecto incorpora técnicas de protección frente a ataques comunes, tales como Man-in-the-Middle, Replay, derivación de claves y canales laterales, haciendo uso de MAC, NONCE, tamaños de clave adecuados y secure-comparator.

En conclusión, el trabajo desarrollado proporciona un sistema seguro y eficiente que cumple con los objetivos de preservar la integridad de la información tanto en el almacenamiento como en la transmisión, sentando las bases para un servicio financiero confiable en entornos distribuidos.

2. Core

2.1 Arquitectura



En el desarrollo de nuestro trabajo hemos implementado una arquitectura cliente-servidor sustentada en el uso de sockets como medio de comunicación entre ambas partes. Este enfoque nos permitió establecer un canal de comunicación bidireccional, en el cual el cliente inicia la conexión con el servidor y ambos pueden intercambiar mensajes y datos de manera ordenada y confiable. Cada extremo de la arquitectura dispone de un socket que actúa como interfaz de enlace, facilitando la transmisión y recepción de la información.

Asimismo, el modelo cliente-servidor aporta una clara separación de roles: el cliente se encarga de solicitar los servicios, mientras que el servidor centraliza el procesamiento y la gestión de las respuestas. De esta manera, logramos una comunicación eficiente, escalable y coherente con los objetivos planteados en nuestro proyecto.

2.2 Decisiones

Para asegurar la integridad de los mensajes transmitidos, utilizamos la función hmac de Python para generar códigos de autenticación MAC altamente robustos y confiables. Esta implementación garantiza que cualquier alteración no autorizada de los mensajes sea fácilmente detectable durante la comunicación entre cliente y servidor. Además, los mensajes seguros incorporan un mecanismo NONCE, que consiste en un número único utilizado una sola vez, con el propósito de contrarrestar ataques de repetición.

Diseñamos el cliente en dos variantes, una versión básica que no incluye protección criptográfica, y otra versión segura, que integra tanto el cálculo de MAC como el uso de NONCEs. Esto permitió comparar y evaluar la eficiencia y robustez de los mecanismos de protección implementados.

Para poner a prueba la seguridad del sistema, desarrollamos un interceptador tipo Man-in-the-Middle, capaz de modificar mensajes en la comunicación sin protección, mientras que en el protocolo seguro dicha manipulación es detectada y bloqueada gracias a la verificación criptográfica. Asimismo, también desarrollamos mecanismos para prevenir ataques replay, los cuales consisten en la interceptación y repetición de mensajes válidos por parte de actores maliciosos con el objetivo de ejecutar acciones fraudulentas. Para mitigar este tipo de ataque, incorporamos el uso de NONCEs, que son números

únicos y de un solo uso, asegurando que cualquier mensaje repetido sea descartado automáticamente por el servidor.

Por otro lado, también contemplamos ataques por fuerza bruta y diccionario destinados a filtrar o descubrir contraseñas. Si bien el sistema actual emplea hashes robustos para el almacenamiento seguro de las credenciales, implementamos medidas adicionales como limitación de intentos y cierre de conexión después de múltiples intentos fallidos.

Además, empleamos librerías modernas para mejorar la visualización en consola, como es el caso de rich para incorporar colores, y el manejo eficiente de hilos y concurrencia, garantizando un código bien organizado y de fácil mantenimiento. Para el correcto funcionamiento de rich fue necesario instalar la librería utilizando el comando pip install rich.

Finalmente, la persistencia de los datos sensibles está asegurada mediante una base de datos MySQL, empleando el programa HeidiSQL, que almacena de forma estructurada los usuarios, contraseñas cifradas y las transacciones registradas, manteniendo un equilibrio óptimo entre seguridad, rendimiento y funcionalidad.

3. Pruebas

3.1 Validación funcional básica

El sistema permite registrar usuarios con datos completos y validar sus credenciales para asegurar un acceso controlado. Los usuarios pueden iniciar y cerrar sesión, manteniendo activas las sesiones mientras interactúan con el sistema. Asimismo, el sistema facilita la realización de transacciones, verificando la validez de los datos enviados y proporcionando confirmaciones claras desde el servidor. También, se implementa una visualización segura del número de cuenta vinculada al usuario, garantizando que esta información solo sea accesible durante sesiones autenticadas. Finalmente también se implementó la opción de eliminar usuario en caso de que se quisiera eliminar una cuenta y todos sus datos almacenados, pudiéndose solo eliminar la cuenta en la que se ha iniciado sesión y solo tras haber introducido el usuario y contraseña asociados a dicha cuenta.

Conexión cliente-servidor:

```
PS C:\Users\silvi\OneDrive\Escritorio\Estudio Sevilla\Sevilla
(US)\SSII\Practica1_SSII\Practica1_SSII> & C:/Users/silvi/App
PS C:\Users\silvi\OneDrive\Escritorio\Estudio Sevilla\Sevilla
(US)\SSII\Practica1_SSII\Practica1_SSII> & C:/Users/silvi/App
Data/Local/Programs/Python/Python312/python.exe "c:/Users/sil
vi/OneDrive/Escritorio/Estudio Sevilla/Sevilla(US)/SSII/Pract
ica1_SSII/Practica1_SSII/serversocket.py"
Conexión exitosa a la base de datos
✓ 3 usuarios preexistentes cargados con número de cuenta.
El servidor está esperando conexiones...
Conexión desde ('172.20.10.3', 1871) establecida.
Respuesta al cliente: Usuario registrado con número de
cuenta.
Respuesta al cliente: Inicio de sesión exitoso
Respuesta al cliente: [('ES0000000154',)]
Respuesta al cliente: ✅ 10.0 € enviados a la cuenta
ES0000000004 perteneciente a Silvia Castillo
Cliente cerró la conexión
Servidor cerrado
```

Conexión cliente-servidor seguro:

```
PS C:\Users\silvi\OneDrive\Escritorio\Estudio Sevilla\Sevilla
(US)\SSII\Practica1_SSII\Practica1_SSII> python serversocket_
seguro.py
🔐 Conexión segura a la base de datos
✓ Usuarios preexistentes cargados con seguridad.
🔐 Servidor seguro esperando conexiones...
🔐 Conexión segura desde ('172.20.10.3', 19910)
Respuesta: Usuario registrado con seguridad.
Respuesta: Inicio de sesión exitoso
Respuesta: ('ES0000000190',)
Respuesta: ✅ 22.0 € enviados a la cuenta ES0000000154
perteneciente a Luis Pardo
Cliente cerró la conexión
🔌 Servidor cerrado tras desconexión del cliente.
Servidor apagado.
PS C:\Users\silvi\OneDrive\Escritorio\Estudio Sevilla\Sevilla
(US)\SSII\Practica1_SSII\Practica1_SSII>
```

3.2 Ataque Man-in-the-Middle

Los ataques Man-in-the-Middle se caracterizan por la intervención de un tercero malicioso que intercepta y modifica la comunicación entre dos partes sin que estas lo detecten. En el canal inseguro del sistema, este tipo de ataque pudo realizarse con éxito,

permitiendo alterar los mensajes sin ser detectado. Sin embargo, en el protocolo seguro, la verificación mediante MAC implementada en el servidor detectó y rechazó cualquier intento de modificación.

Prueba ataque Man-in-the-Middle cliente-servidor inseguros:

```
PS C:\Users\silvi\OneDrive\Escritorio\Estudio Sevilla\Sevilla(US)\SSII\Practica1_SSII\Practica1_SSII> python mitm.py
Usuario maligno escuchando en 127.0.0.1:9000
Cliente conectado desde ('127.0.0.1', 8575)
Interceptado: 2,lolalola,hola1234
Interceptado: Inicio de sesión exitoso
Interceptado: 4,lolalola,ES0000000003,20.0
¿Quieres modificar los campos de esta transacción? (s/n):s
Campos actuales:
1. Usuario origen: lolalola
2. Cuenta destino: ES0000000003
3. Cantidad: 20.0
Nueva cuenta destino (enter para no cambiar):
Nueva cantidad (enter para no cambiar): 200
Mensaje modificado a: 4,lolalola,ES0000000003,200
Interceptado: ✅ 200.0 € enviados a la cuenta ES0000000003 perteneciente a Victor Ramos
```

Prueba ataque Man-in-the-Middle cliente-servidor seguros:

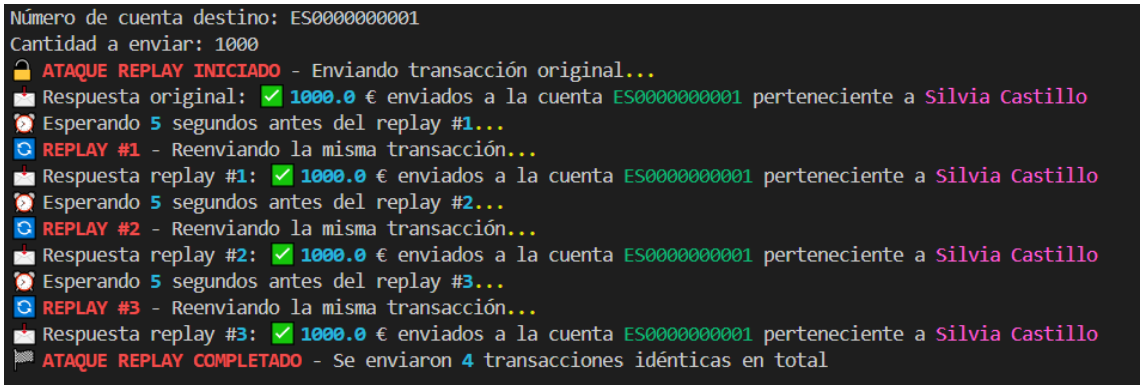
```
PS C:\Users\silvi\OneDrive\Escritorio\Estudio Sevilla\Sevilla(US)\SSII\Practica1_SSII\Practica1_SSII> python mit
m.py
Usuario maligno escuchando en 127.0.0.1:9000
Cliente conectado desde ('127.0.0.1', 27596)
Interceptado: 2,silcasrubi,carolaR45
Interceptado: Inicio de sesión exitoso
Interceptado:
4s,silcasrubi,ES0000000002,10.0,n51205,c243025d778e5735e0eaf34a74ff12730aa48472bcf4cfb19daec02eefe5cb09
¿Quieres modificar los campos de esta transacción segura? (s/n):s
Campos actuales:
1. Usuario origen: silcasrubi
2. Cuenta destino: ES0000000002
3. Cantidad: 10.0
4. Nonce: n51205
5. MAC: c243025d778e5735e0eaf34a74ff12730aa48472bcf4cfb19daec02eefe5cb09
Nuevo usuario origen (enter para no cambiar):
Nueva cuenta destino (enter para no cambiar):
Nueva cantidad (enter para no cambiar): 100
Nuevo nonce (enter para no cambiar):
Mensaje modificado y MAC recalculada a:
4s,silcasrubi,ES0000000002,100,n51205,3f0aadfff6c33e76c4f8a1d397631370a9c8d270b01d0a02207112a1dce653b7
Interceptado: ❌ Ataque detectado: MAC no válido.
```

3.3 Ataques Replay

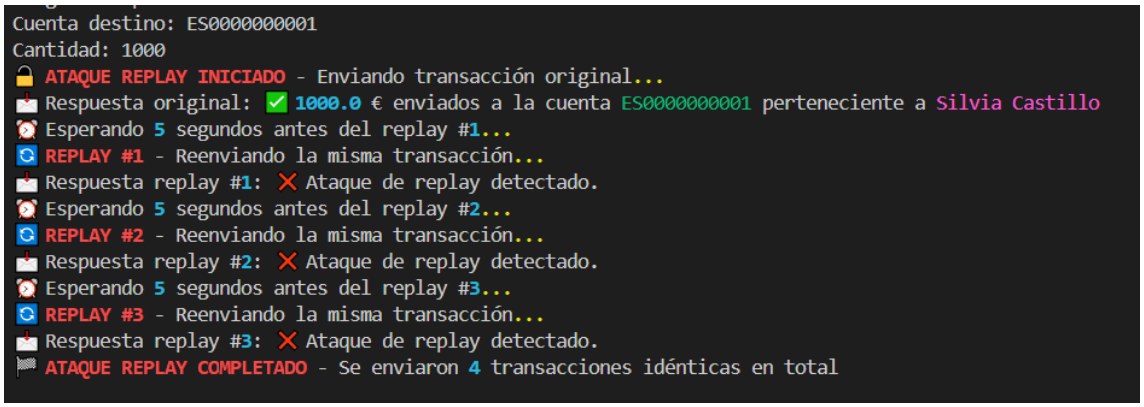
Los ataques de replay consisten en interceptar transmisiones de datos para luego retransmitirlas fraudulentamente. Esto puede permitir repetir acciones autorizadas, realizar transacciones duplicadas o acceder múltiples veces a un sistema. Para eliminar este riesgo, nuestro sistema utiliza NONCEs, que son valores únicos para cada mensaje,

garantizando que ningún mensaje se acepte más de una vez. Gracias a esto, cualquier intento de retransmisión es detectado y bloqueado, manteniendo la seguridad y la integridad de las comunicaciones.

Prueba ataque Replay cliente-servidor inseguros:



Prueba ataque Replay cliente-servidor seguros:



3.4 Ataques de fuerza bruta y diccionario

Los ataques de fuerza bruta y por diccionario consisten en intentar descubrir contraseñas probando múltiples combinaciones o palabras comunes sistemáticamente. Aunque el sistema utiliza hashes robustos para almacenar contraseñas, estos ataques pueden ser efectivos si no se aplican medidas adicionales. Por ello implementamos mecanismos de limitación de intentos y bloqueos, para dificultar estos ataques automatizados y proteger el acceso al sistema.

Prueba ataque por diccionario-servidor inseguro:

```
PS C:\Users\amara\OneDrive\Escritorio\SSII\Practica1_SSII> python ataques.py
Conectado al servidor

Opciones:
1. Login normal
2. Ataque por Diccionario
3. Ataque por Fuerza Bruta
4. Salir
Elige opción: 2
Usuario: rolerAmari
Probando contraseña: 1234 -> fallo
Probando contraseña: password -> fallo
Contraseña encontrada para rolerAmari: pepita
```

Prueba ataque por diccionario-servidor seguro:

```
PS C:\Users\amara\OneDrive\Escritorio\SSII\Practica1_SSII> python ataques.py
Conectado al servidor

Opciones:
1. Login normal
2. Ataque por Diccionario
3. Ataque por Fuerza Bruta
4. Salir
Elige opción: 2
Usuario: silcasrubi
Probando contraseña: 1234 -> fallo
Probando contraseña: password -> fallo
Probando contraseña: pepita -> fallo
Probando contraseña: admin -> fallo
Probando contraseña: contraseña123 -> fallo
Probando contraseña: qwerty -> fallo
Traceback (most recent call last):
  File "C:\Users\amara\OneDrive\Escritorio\SSII\Practica1_SSII\ataques.py", line 83, in <module>
    main()
  File "C:\Users\amara\OneDrive\Escritorio\SSII\Practica1_SSII\ataques.py", line 68, in main
    ataque_diccionario(client_socket, usuario, diccionario)
  File "C:\Users\amara\OneDrive\Escritorio\SSII\Practica1_SSII\ataques.py", line 20, in ataque_diccionario
    respuesta = client_socket.recv(1024).decode()
ConnectionResetError: [WinError 10054] Se ha forzado la interrupción de una conexión existente por el host remoto
```

Prueba por Ataque Fuerza Bruta cliente-servidor inseguro:

```
PS C:\Users\amara\OneDrive\Escritorio\SSII\Practica1_SSII> python ataques.py
Conectado al servidor

Opciones:
1. Login normal
2. Ataque por Diccionario
3. Ataque por Fuerza Bruta
4. Salir
Elige opción: 3
Usuario: rolerAmari
Longitud mínima de la contraseña: 5
Longitud máxima de la contraseña: 9
Fuerza bruta para usuario 'rolerAmari' desde longitud 5 a 9 iniciada...
Probada contraseña: aaaaa -> fallo
Probada contraseña: aaaab -> fallo
Probada contraseña: aaaac -> fallo
Probada contraseña: aaaad -> fallo
Probada contraseña: aaaae -> fallo
Probada contraseña: aaaaf -> fallo
Probada contraseña: aaaag -> fallo
Probada contraseña: aaaah -> fallo
Probada contraseña: aaaai -> fallo
Probada contraseña: aaaaj -> fallo
Probada contraseña: aaaak -> fallo
Probada contraseña: aaaal -> fallo
Probada contraseña: aaaam -> fallo
Probada contraseña: aaaan -> fallo
Probada contraseña: aaaao -> fallo
Probada contraseña: aaaap -> fallo
Probada contraseña: aaaaq -> fallo
```

Prueba por Ataque Fuerza Bruta cliente-servidor seguro:

```
PS C:\Users\amara\OneDrive\Escritorio\SSII\Practica1_SSII> python ataques.py
Conectado al servidor

Opciones:
1. Login normal
2. Ataque por Diccionario
3. Ataque por Fuerza Bruta
4. Salir
Elige opción: 3
Usuario: silcasrubi
Longitud mínima de la contraseña: 5
Longitud máxima de la contraseña: 9
Fuerza bruta para usuario 'silcasrubi' desde longitud 5 a 9 iniciada...
Probada contraseña: aaaaa -> fallo
Probada contraseña: aaaab -> fallo
Probada contraseña: aaaac -> fallo
Probada contraseña: aaaad -> fallo
Probada contraseña: aaaae -> fallo
Probada contraseña: aaaaf -> fallo
Traceback (most recent call last):
  File "C:\Users\amara\OneDrive\Escritorio\SSII\Practica1_SSII\ataques.py", line 83, in <module>
    main()
  File "C:\Users\amara\OneDrive\Escritorio\SSII\Practica1_SSII\ataques.py", line 73, in main
    ataque_fuerza_bruta(client_socket, usuario, longitud_min, longitud_max)
  File "C:\Users\amara\OneDrive\Escritorio\SSII\Practica1_SSII\ataques.py", line 36, in ataque_fuerza_bruta
    if login(client_socket, usuario, pwd):
  File "C:\Users\amara\OneDrive\Escritorio\SSII\Practica1_SSII\ataques.py", line 12, in login
    respuesta = client_socket.recv(1024).decode()
ConnectionResetError: [WinError 10054] Se ha forzado la interrupción de una conexión existente por el host remoto
```

4. Conclusiones

Este proyecto ha desarrollado un sistema seguro y eficiente para la gestión de transferencias financieras en una arquitectura cliente-servidor basada en sockets. Se implementaron mecanismos criptográficos como códigos MAC y NONCEs, que garantizan la integridad y autenticidad de las transmisiones, protegiendo el sistema contra ataques comunes como Man-in-the-Middle, replay, fuerza bruta y diccionario. Además, la gestión segura de sesiones y el almacenamiento cifrado en base de datos proporcionan una protección robusta para la información sensible. Las pruebas realizadas muestran la capacidad del sistema para detectar manipulaciones, evitar retransmisiones fraudulentas y limitar accesos no autorizados.

5. Referencias

- OpenAI. ChatGPT [modelo de lenguaje]. https://chat.openai.com/
- Perplexity AI. Perplexity Pro [motor de respuestas con citas]. https://www.perplexity.ai/
- GitHub. GitHub Copilot [asistente de programación]. https://github.com/features/copilot