INSEGUS

CYBERSECURITY CONSULTING

Antonio Abad Correa
José Maria Solís
Pedro Lacárcel

Sistema seguro de mensajería

Resumen ejecutivo

El presente proyecto desarrolla una infraestructura segura de acceso remoto basada en tecnología VPN SSL (Secure Sockets Layer), enmarcada dentro de la política BYOD (Bring Your Own Device) adoptada por una universidad pública. El propósito fundamental consiste en permitir que el personal autorizado acceda a los recursos institucionales —tales como bases de datos, servicios de correo electrónico y sistemas corporativos— mediante sus dispositivos personales, garantizando los principios de confidencialidad, integridad, autenticidad y disponibilidad definidos por el Esquema Nacional de Seguridad (ENS).

La solución técnica implementada se sustenta en una arquitectura cliente-servidor utilizando sockets seguros SSL/TLS, que posibilitan el establecimiento de canales de comunicación cifrados bajo TLS 1.3 y el uso de Cipher Suites criptográficamente robustas. El sistema incorpora mecanismos de gestión y verificación segura de credenciales, control de sesiones y registro persistente de mensajes, implementando medidas de protección frente a ataques de fuerza bruta, interceptación y manipulación de datos.

El modelo propuesto ha sido diseñado conforme a los requisitos funcionales, de información y de seguridad establecidos, asegurando la protección de los datos personales conforme al Reglamento General de Protección de Datos (RGPD) y la Ley Orgánica 3/2018 (LOPDGDD). Para la validación de la solución se ejecutaron pruebas de rendimiento y de seguridad empleando herramientas de análisis de tráfico de red (Wireshark, RawCap, tcpdump), verificando la correcta implementación de los canales cifrados y la imposibilidad de lectura del tráfico interceptado.

Los resultados obtenidos demuestran la eficacia y robustez del canal VPN SSL para proteger las comunicaciones entre usuarios remotos y servidores institucionales, así como la viabilidad operativa del entorno BYOD en el ámbito de la Administración Pública, sin degradación significativa del rendimiento del sistema ni vulneración de la normativa aplicable en materia de seguridad y protección de la información.

Core

Requisitos funcionales

Registro de usuarios:

- Permitir que un nuevo usuario se registre proporcionando únicamente un nombre de usuario y una contraseña.
- Informar al usuario si ya existe un registro con el mismo nombre de usuario.
- No permitir modificaciones a los datos de usuarios después del registro.

```
if opcion == "nuevo":
    conn.sendall(b"Introduce un nombre de usuario:\n")

    username = conn.recv(1024).decode().strip()
    print("Nombre de usuario:" + username)
    if usuario_existe(username):
        conn.sendall(b"Usuario ya existe. Prueba con otro.\n")
    else:
        while True:
            conn.sendall(b"Introduce una contrasena:\n")
            password = conn.recv(1024).strip()

            is_valid, reason = is_strong_password(password)

            if is_valid:
                password_txt = password.decode("utf-8")
                crear_usuario(username, password_txt)
                conn.sendall(b"Registro completado. Por favor, inicia sesion a continuacion.\n")
                opcion = "login"
                break
            else:
                conn.sendall(reason.encode() + b"\n")
```

1 Función de registro de usuario

Inicio de sesión:

- Permitir a los usuarios registrados iniciar sesión introduciendo su nombre de usuario y contraseña.

```python
if opcion == "login":

    while True:
        (variable) username: str  nombre de usuario:\n")
        username = conn.recv(1024).decode().strip()
        print("Nombre de usuario: " + username)
        is_locked, tiempo_restante = bloqueado(username)
        if is_locked is True:
            msg = f"Usuario bloqueado temporalmente. Intente de nuevo en {tiempo_restante} segundos.\n"
            conn.sendall(msg.encode())
            continue
        else:
            conn.sendall(b"Introduce una contrasena:\n")
            password = conn.recv(1024).strip()

        if usuario_existe(username):
            password_txt = password.decode("utf-8")
            verficacion = verificar_usuario(username, password_txt)
            if verficacion == False:
                is_locked, restantes = registrar_fallo(username)
                if is_locked:
                    msg = f"Usuario bloqueado por {LOCK_SECONDS} segundos.\n"
                    conn.sendall(msg.encode())
                    continue
                else:
                    msg = f"intentos restantes: {restantes}. Vuelve a intentarlo"
                    conn.sendall(msg.encode())

            else:
                conn.sendall(b"Login exitoso.\n")
```

2 Función de inicio de sesión

Verificar credenciales:

- Validar las credenciales proporcionadas contra los datos almacenados en el servidor.
- Denegar acceso si las credenciales no coinciden con un registro válido.

```python
else:
    conn.sendall(b"Usuario o contrasena incorrectos (FINAL).\n")
```

3 Parte del login que deniega si las credenciales no son correctas

Cerrar sesión:

- Permitir a los usuarios ya logados cerrar la sesión.

```
            elif opcion_sesion == "3":
                try:
                    conn.sendall(b"Cerrando sesion. Adios.\n")
                except Exception:
                    pass
                print(f"Cerrando conexión con {addr}")
                try:
                    conn.shutdown(socket.SHUT_RDWR)
                except Exception:
                    pass
                try:
                    conn.close()
                except Exception:
                    pass
```

4 Parte de la funcionalidad del login para cierre de sesión

Gestión de usuarios preexistentes:

- El sistema debe contar con un conjunto inicial de usuarios registrados para que puedan acceder sin necesidad de registrarse.

Mensajes:

- Permitir a los usuarios autenticados realizar envió de mensajes de texto al servidor.

```
if opcion_sesion == "1":
    # Enviar mensaje a otro usuario
    conn.sendall(b"Introduce el nombre del destinatario:\n")
    destinatario = conn.recv(1024).strip().decode('utf-8')
    if not usuario_existe(destinatario):
        mensaje_a_enviar = "\n Error: El destinatario no existe. Operacion cancelada.\n"
    else:
        conn.sendall(b"Introduce el mensaje:\n")
        contenido = conn.recv(4096).strip().decode('utf-8')
        ok = enviar_mensaje(username, destinatario, contenido)
        if ok:
            mensaje_a_enviar = "Mensaje enviado correctamente.\n"
        else:
            mensaje_a_enviar = "Error al enviar el mensaje.\n"
```

5 Envio de mensajes entre usuarios

Persistencia de datos:

- Registrar y almacenar de manera permanente: Los datos de los usuarios.
- Registrar mensajes enviados por usuario. Nº de mensajes enviados

Interfaz de comunicación:

- Proveer una interfaz (***usando sockets seguros***) que permita a los clientes interactuar con el sistema de registro, autenticación, y envío de mensajes.

```
with ssl_context.wrap_socket(server_socket, server_side=True) as ssl_socket:
    while True:
        conn, addr = ssl_socket.accept()
        print(f"Conexión establecida con: {addr}")
        with conn:
            # primer prompt: nuevo/login
            conn.sendall(b"Eres nuevo usuario o quieres loggearte? nuevo/login\n")
            data = conn.recv(1024)
            if not data:
                continue
            opcion = data.decode().strip().lower()
            if opcion == "nuevo":
                _ = handle_registration(conn)  # fuerza registro; cliente deberá re-logear
                conn.sendall(b"Eres nuevo usuario o quieres loggearte? nuevo/login\n")
                data = conn.recv(1024)
                if not data:
                    continue
                opcion = data.decode().strip().lower()

            if opcion == "login":
                username = None
                # intentar login hasta que tenga éxito o se desconecte
                while username is None:
                    username = handle_login(conn)
                    if username is None:
                        # handle_login ya envió mensajes de error; intentar de nuevo
                        continue
                    # Si login OK, enviar nonce (si requiere)
                    # generar nonce si lo necesitas, aquí solo procedemos al session
                    try:
```

6 Función para el uso de login y registro por socket seguro

Requisitos de Información

Datos de usuarios:

- Nombre de usuario (único).
- Contraseña.

```
 id | username  |                                                              password
----+-----------+--------------------------------------------------------------------------------------------------------
  1 | pedro1234 | $2b$12$LudYCCz74FyEh9.Ky532MOIuKb1XUmfxE.immLTBpgVf7q0RUk44u
  2 | yosisolis | $2b$12$C2kPzCBnWSv1v8bkdP4roebxAJtH7N/zK.mOpjb.uLgS4uxmOk3Eq
  4 | joaquin   | $2b$12$sl0/eiCR2UqqslyRw7VRLOewTh3KgTcX2zHW7ejzlbdTyP.QvqBmC
  5 | quino     | $2b$12$u5VRUS.4MYR5A2qiEa0bkufOC4KMh0sdqQ6chyipJMj5F6TZ2QqCG
  7 | jose      | $2b$12$bRkY.9jmCuLMbgijvS/7kul8HwGtTFH2GCIffO6C9kpwKAiwT8atG
  8 | pua       | $2b$12$73ROV/7YcRhbOQ9JW9FIauQMKgovBeNYoNhELMSlh8fNh7nWffMfG
  9 | marta     | $2b$12$gfHwnUxTViR3m2Lz3jP05OT9foCKjBMV8KYWM9VwQlEz.DniToXgC
  6 | luis      | $2b$12$IcsC4XmnaHX.mwMSS2DkierXpRiuoLr4pjMi1zeALMrAAQwXLxe1q
 10 | semari    | $2b$12$awMRExJJaUzfEpImq/NY1udKx0msbQAhi7L2hub4jjDxPBntJ/3Gm
 11 | antonio   | $2b$12$IoE5Qzs9phVuAQKbnx1Xlu6odoAi8FXl1cnD9shAEENo0oQNtaNVC
 12 | elpapi    | $2b$12$K7hTS/09Sn.nFg7p1QwAdePJLJne8/YaKxtxpVHoiCif1OZm3JRIO
 13 | angel     | $2b$12$FqYhIhoTZmOZrTi7ZbT07OtyiyYnWwJhJ17vuT9MkwfldP/XD5XSa
```

7 Base de datos

Para la base de datos hacemos uso de PostgreSQL 17.6

Registro inicial:

- Una lista de usuarios pre-registrados.
- Sin mensajes realizadas previamente.

Como hemos visto previamente en la anterior foto.

Registro de mensajes:

- Número de mensajes enviados por usuario y fecha.
- Este registro se encuentra en un tabla SQL la cual se crea automáticamente al enviar el primer mensaje.

Mensajes:

- Identificador de usuario.
- Cadena de texto del mensaje a enviar (máximo 144 caracteres).

Mensajes del sistema:

- Usuario registrado exitosamente.
- Usuario ya registrado.
- Inicio de sesión exitoso o fallido.
- Mensaje enviado y recibido correctamente.

Los mensajes del sistema están integrados en las funciones para saber los fallos de cada uno de los pasos que vamos dando.

Requisitos de seguridad

Requisitos de Seguridad para las Credenciales de Usuario

- Almacenamiento Seguro de Credenciales
- Verificación Segura de Credenciales
- Protección Contra Ataques en Login (BruteForce)

Para esta parte lo tenemos integrado en **time_functions.py** evitando el acceso mediante fuerza bruta.

Requisitos de Seguridad para el envío de Mensajes

- Integridad en el envío de mensajes.
- Confidencialidad en el envío de mensajes.
- Autenticidad en el envío de mensajes.

Para el envío de mensaje hemos usado **TLS 1.3** con **Python 3.11.9**.

Requisitos de Seguridad para la Base de Datos

- Integridad de los datos almacenados en base de datos.

```python
# Función para crear cuenta y guardar contraseña encriptada
def crear_cuenta(username, password):
    hashed = bcrypt.hashpw(password.encode("utf-8"), bcrypt.gensalt())
    try:
        conn = sqlite3.connect("usuarios.db")
        cursor = conn.cursor()
        cursor.execute("INSERT INTO usuarios (username, password) VALUES (?, ?)", (username, hashed))
        conn.commit()
        conn.close()
        return "✅ Usuario registrado con éxito"
    except sqlite3.IntegrityError:
        return "⚠️ El usuario ya existe"
```

8 Función de creacion de cuentas seguro en la base de datos

Pruebas

Para realizar pruebas se debe seguir el documento ***ANEXO-uso del sistema.pdf*** (ubicada en la carpeta pruebas-logs) donde se encuentran las indicaciones para la configuración del entorno del sistema y pasos para realizar pruebas de seguridad y funcionalidad.

Conclusión

El desarrollo e implementación de la VPN SSL Road Warrior ha permitido verificar la viabilidad técnica de un sistema de comunicaciones seguras basado en TLS 1.3, que cumple de manera efectiva los principios de seguridad de la información definidos en el Esquema Nacional de Seguridad (ENS) y las directrices europeas de protección de datos.

Las pruebas experimentales realizadas con sniffers de red y herramientas de monitorización de tráfico evidenciaron la correcta encapsulación y cifrado de los paquetes transmitidos a través de los sockets SSL, garantizando la confidencialidad del flujo de información, la autenticidad del emisor y receptor y la integridad de los datos frente a ataques de interceptación o alteración. Asimismo, los resultados de rendimiento muestran una pérdida marginal de eficiencia, confirmando que el cifrado no supone un impacto operativo significativo sobre la capacidad concurrente estimada de 300 usuarios.

El sistema propuesto satisface de manera completa los requisitos funcionales, de información y de seguridad establecidos en el análisis inicial, integrando medidas de protección de credenciales, gestión segura de sesiones, registro persistente de mensajes y validación criptográfica del canal de comunicación. Desde una perspectiva arquitectónica, la solución es escalable, interoperable y adaptable a otros entornos institucionales que requieran la implementación de políticas BYOD seguras.

En conclusión, el proyecto BYODSEC – Road Warrior VPN SSL constituye una propuesta tecnológica sólida y conforme con las normas de seguridad vigentes, demostrando la capacidad de las infraestructuras VPN SSL para proporcionar conectividad remota segura en entornos de Administración Pública. Su correcta integración contribuye a la madurez de las políticas de seguridad de la información institucionales y representa una referencia práctica en el ámbito del aseguramiento de la información y las comunicaciones seguras basadas en TLS.

Bibliografía

MANUAL DE USO DE WIRESHARK:
https://www.wireshark.org/docs/wsug_html_chunked/

CODIGOS DE EJEMPLO DE WRAP TLS/SSL

https://docs.python.org/3/library/ssl.html

FUNCIONAMIENTO Y ESQUEMAS DE SOCKET

https://www.ibm.com/docs/es/aix/7.2.0?topic=protocols-socket

PRUEBAS SNNIFER FUNCIONAMIENTO

https://es.wikipedia.org/wiki/Test_de_detecci%C3%B3n_de_sniffer