CIBERSEGURIDAD

PRÁCTICA 3: Atacando una aplicación Web

AUTORES

CARLOS GARCÍA SANTA

GRUPO 14

INGENIERÍA INFORMÁTICA

ESCUELA POLITÉCNICA SUPERIOR

UNIVERSIDAD AUTÓNOMA DE MADRID

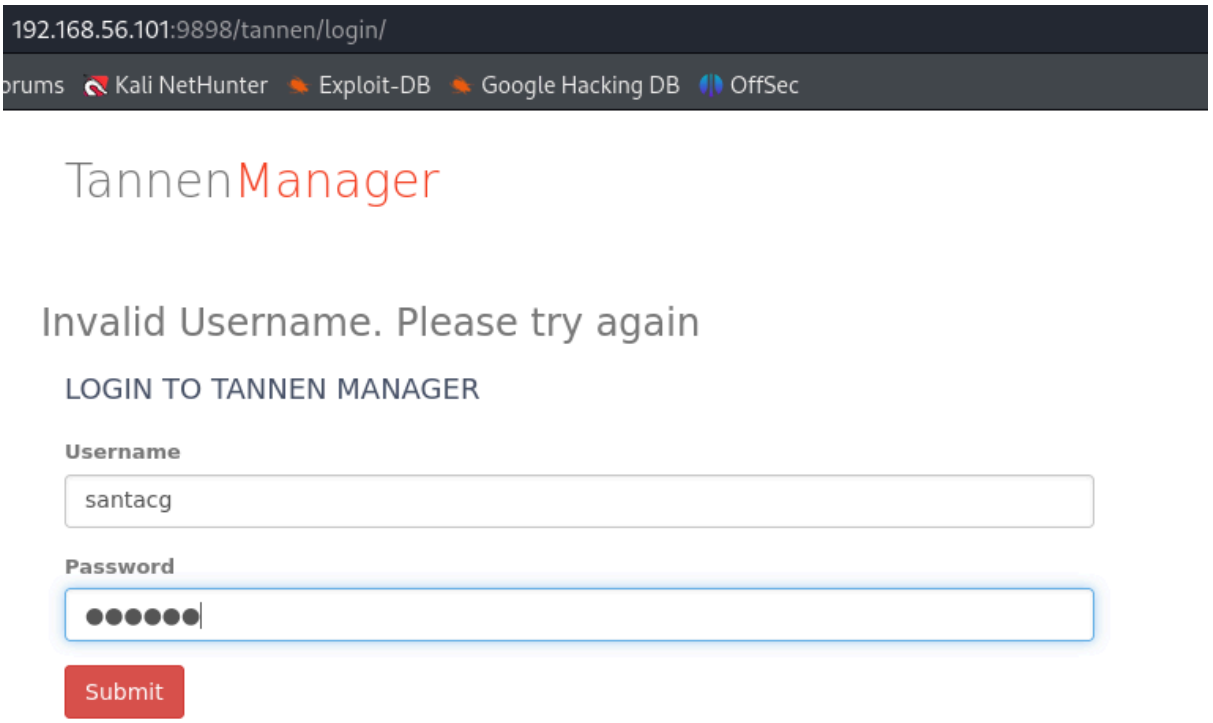
25/04/2025

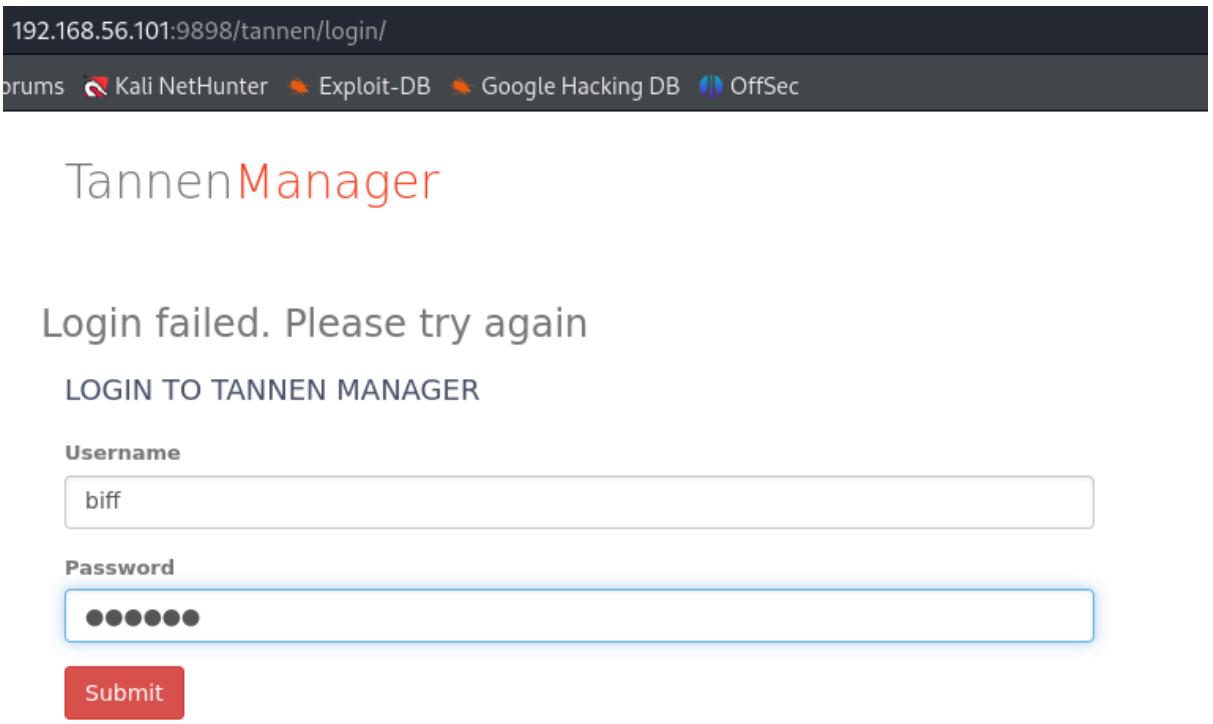
Índice

1. Vulnerabilidad de enumeración de usuarios

Probando la aplicación, se observa rápidamente que, al intentar iniciar sesión, el sistema devuelve mensajes de error distintos en función de si el nombre de usuario existe o no.

Para comprobarlo, se accede al formulario de login en "http://192.168.56.101:9898/tannen/login" y se prueban dos casos: primero se introduce un usuario inexistente "santacg" y después un usuario conocido "biff" con una contraseña incorrecta. En el primer caso, el mensaje mostrado es "Invalid Username. Please try again.", mientras que en el segundo caso es "Login failed. Please try again."





Esto confirma que la aplicación permite la enumeración de usuarios, ya que basta con analizar el mensaje de error para saber si un usuario existe en el sistema. Este tipo de vulnerabilidad facilita ataques como fuerza bruta o phishing.

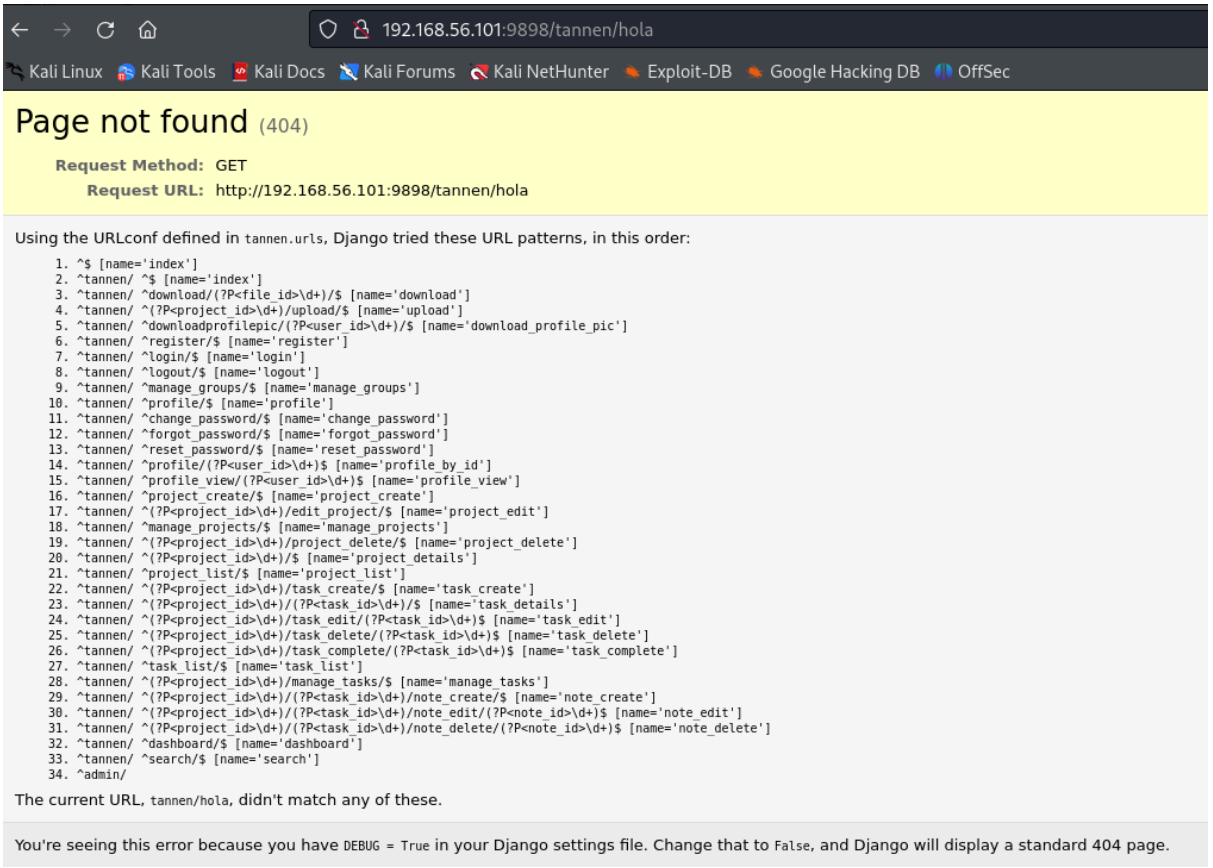
Para solventar este error, se debería unificar el mensaje de error en el login para que no revele información sobre la validez del usuario, mostrando siempre un mensaje genérico como "Invalid username or password."

2. Vulnerabilidad de rutas por debug en Django

Se intenta acceder a una URL inexistente “/tannen/hola”, lo cual genera una página de error del framework Django. Esta respuesta no solamente muestra el típico error 404, sino que además imprime la lista completa de rutas internas configuradas en la aplicación.

Esta exposición permite a cualquier usuario no autenticado descubrir la estructura interna de la aplicación. Además, se observa que la aplicación se encuentra en modo DEBUG=True. El uso de este modo expone innecesariamente detalles técnicos y rutas del sistema que pueden facilitar la exploración o la explotación de vulnerabilidades por parte de atacantes.

Se recomienda desactivar el modo debug y mostrar únicamente mensajes de error genéricos.



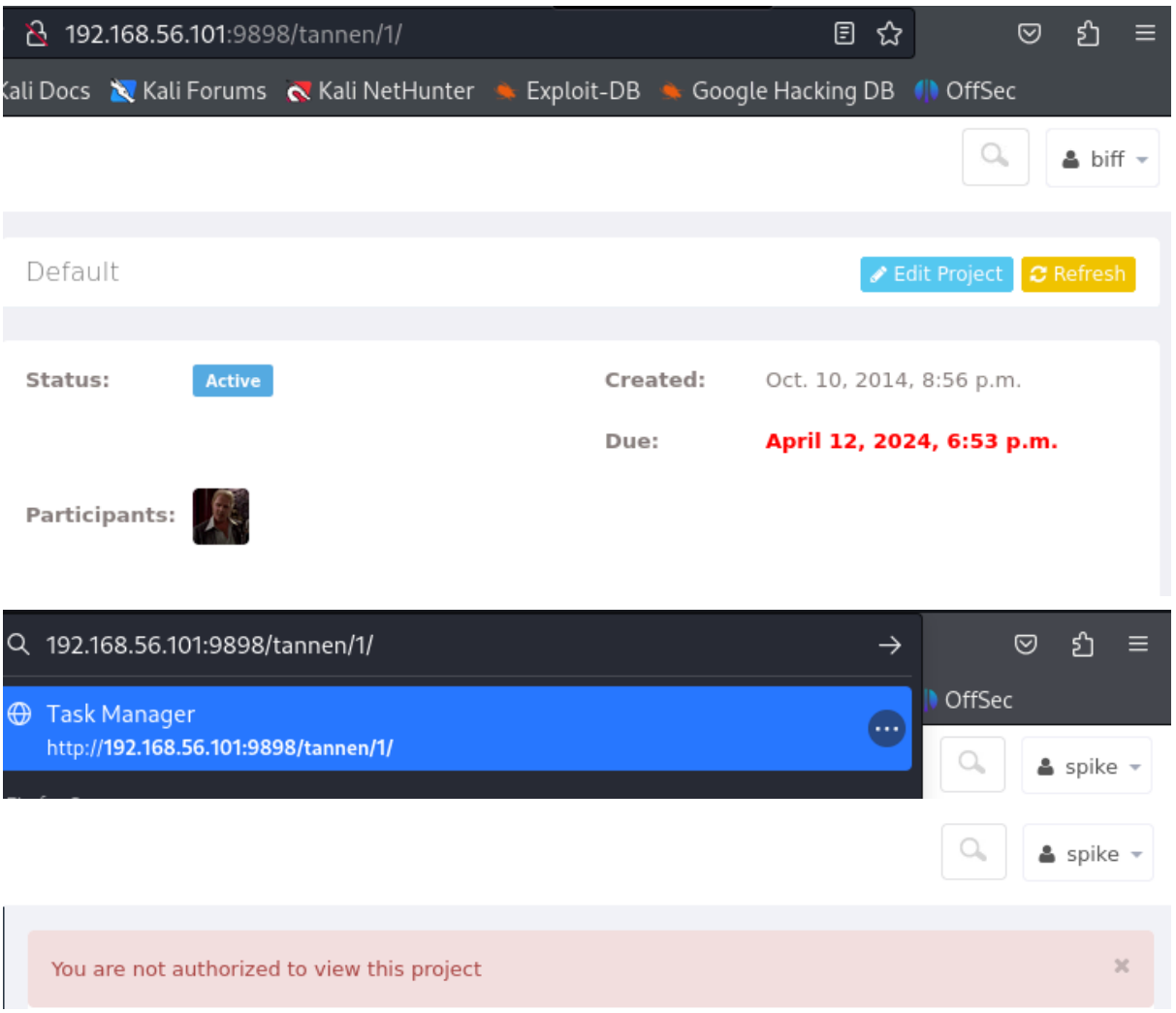
3. Vulnerabilidad de acceso horizontal

Se intenta llevar a cabo una prueba de acceso horizontal no autorizado en la aplicación, que consiste en que un usuario acceda a proyectos que no le pertenecen.

Para ello, se inicia sesión como “biff” y se accede a un proyecto en el “biff” sea el único participante, después obtenemos la url del proyecto: “192.168.56.101:9898/tannen/1/” (el proyecto tiene el id 1). Posteriormente, accedemos a la cuenta “spike” y modificamos manualmente el identificador del proyecto en la URL para intentar acceder al proyecto 1, que pertenece al usuario “biff”.

El sistema responde correctamente mostrando el mensaje "You are not authorized to view this project.", bloqueando el acceso al recurso de otro usuario.

Por tanto, se confirma que la aplicación implementa de manera correcta el control de acceso horizontal, evitando que un usuario pueda visualizar o manipular proyectos ajenos.



4. Vulnerabilidad de inyecciones SQL

Se realiza una prueba para verificar la posible existencia de vulnerabilidades de inyección SQL (SQLi). Se prueban varios puntos críticos de entrada de datos, incluyendo el formulario de login, el formulario de registro de nuevos usuarios y los campos de búsqueda y formularios de creación de proyectos y tareas.

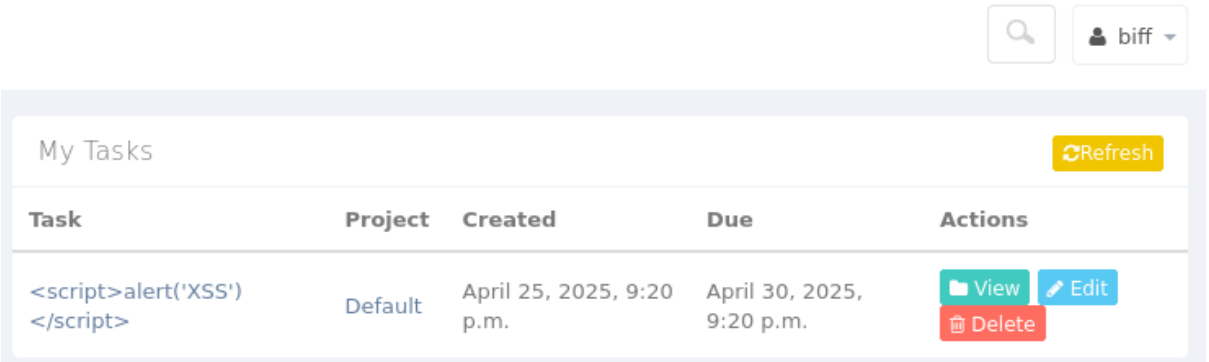
Para ello, se introducen payloads típicos de inyección como ' OR '1'='1' -- tanto en el campo de usuario y contraseña como en los campos de texto disponibles.

En todos los casos, la aplicación responde de forma controlada, mostrando mensajes de error de validación como "Invalid user name" en el login, impidiendo el registro de usuarios con entradas maliciosas y no mostrando resultados anómalos en las búsquedas.

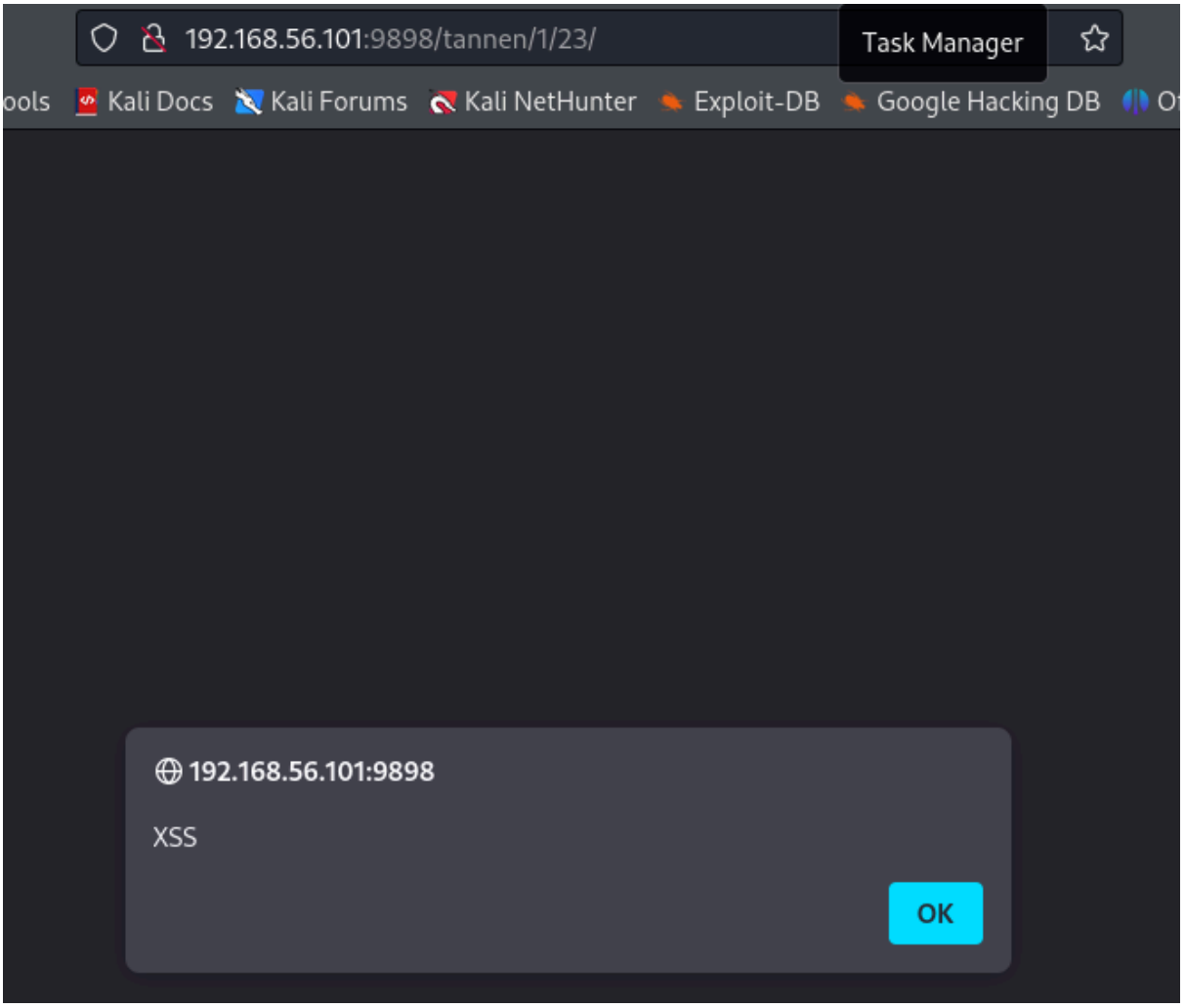
No se observan errores de SQL, comportamientos extraños ni accesos no autorizados, lo que indica que la aplicación implementa correctamente medidas de protección frente a ataques de inyección SQL en los formularios probados.

5. Vulnerabilidad de XSS

En la creación de tareas con el usuario administrador, se detecta una vulnerabilidad de tipo Cross-Site Scripting (XSS) persistente.
Para comprobarlo, se crea una nueva tarea dentro de un proyecto existente, introduciendo el siguiente payload en el campo de descripción "<script>alert('XSS')</script>"



Tras guardar la tarea, al acceder posteriormente a su vista desde el panel, se ejecuta el código JavaScript inyectado, mostrando una ventana emergente "alert" en el navegador. Esto confirma que la aplicación no realiza ninguna validación ni escape de contenido HTML al mostrar los datos de las tareas.



Este tipo de vulnerabilidad permite ejecutar código arbitrario en el navegador de otros usuarios que acceden a la tarea comprometida, lo que podría utilizarse para robar cookies, secuestrar sesiones, realizar ataques CSRF, etc.

Para solucionar esto se debería implementar una sanitización estricta de los datos de entrada en todos los formularios que aceptan texto, así como escapar correctamente los caracteres especiales al renderizar contenido en HTML.

6. Vulnerabilidad de CSRF en formularios

Se detecta que el formulario de recuperación de contraseña "/tannen/forgot_password" no incluye ninguna protección frente a ataques de tipo Cross-Site Request Forgery (CSRF), comprobando el documento HTML asociado al formulario usando "Ctrl U" para ver el código fuente de la página.

A diferencia de otros formularios en la aplicación, como la creación de proyectos o tareas, este formulario carece del campo oculto “csrfmiddlewaretoken”, lo que permite que pueda ser invocado desde páginas externas sin ningún tipo de validación adicional.

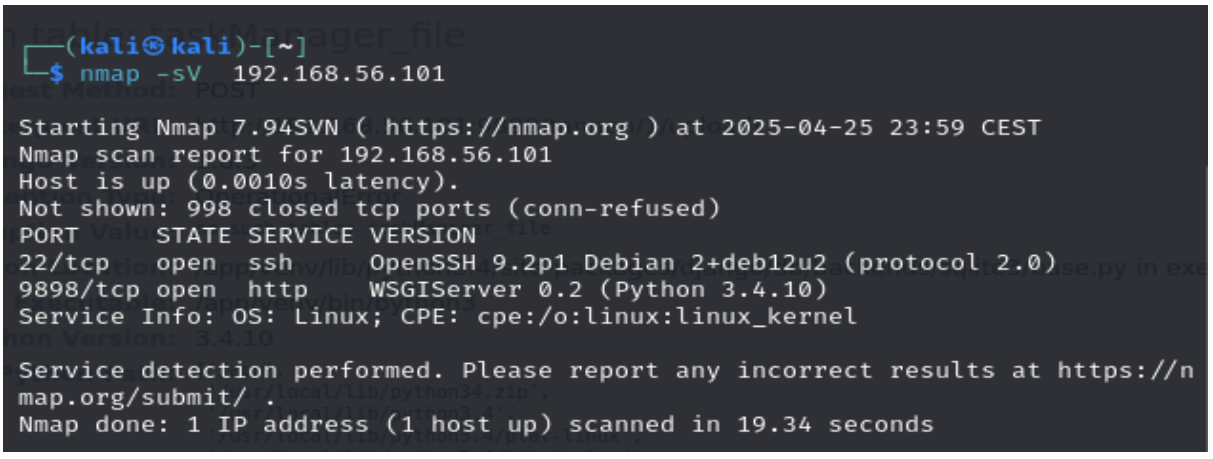
Aunque esta funcionalidad no ejecuta una acción crítica, forma parte del flujo de autenticación y manejo de cuentas, por lo que debería estar protegida. Además, su ausencia podría permitir ataques como envío masivo de peticiones o abuso para enumerar usuarios registrados en el sistema.

Para solucionarlo habría que incluir el token CSRF en el formulario de “forgotten password”, esto además tiene que ocurrir para todos los formularios que hagan peticiones “POST” para asegurar que no pueda ocurrir el problema descrito.

7. Vulnerabilidad de exposición de servicios de red

Mediante un escaneo de puertos utilizando la herramienta nmap, se detectan los servicios accesibles públicamente en la máquina que ejecuta la aplicación.

El escaneo muestra que, además del puerto 9898 correspondiente al servidor web de la aplicación, se encuentra expuesto el puerto 22, correspondiente a un servicio OpenSSH.



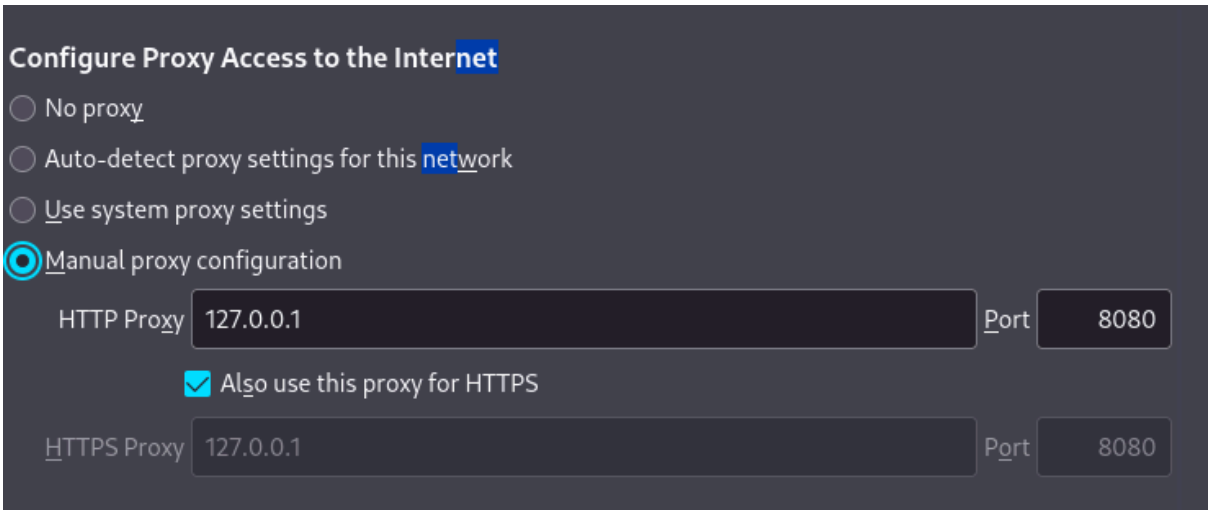
La exposición del servicio SSH supone un riesgo elevado, ya que puede permitir acceso remoto completo a la máquina, si se conocen las credenciales de acceso. Esto facilitaría ataques de fuerza bruta, explotación de vulnerabilidades y movimientos laterales en la red interna.

Se debería restringir el acceso al servicio SSH mediante firewalls o listas blancas de IPs confiables, así como desactivar su exposición en entornos donde no sea necesario.

8. Vulnerabilidad de ataques de fuerza bruta al login

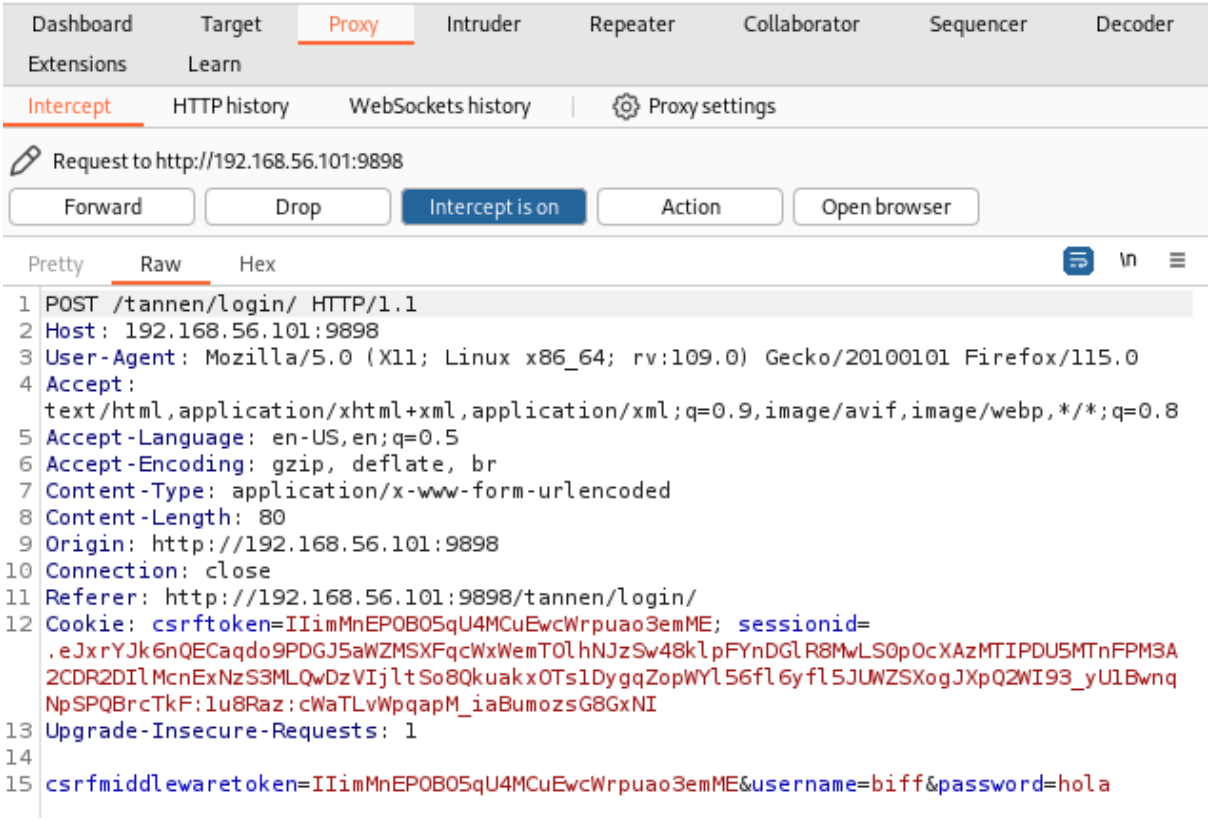
Se realiza un ataque de fuerza bruta sobre el formulario de autenticación de la aplicación utilizando la herramienta Burp Suite Intruder. El objetivo del ataque es comprobar si es posible realizar múltiples intentos de login sin restricciones.

Para llevar a cabo la prueba, se configura Firefox para redirigir todo el tráfico HTTP y HTTPS a través del proxy de Burp Suite, accediendo a la configuración de red en los ajustes del navegador.



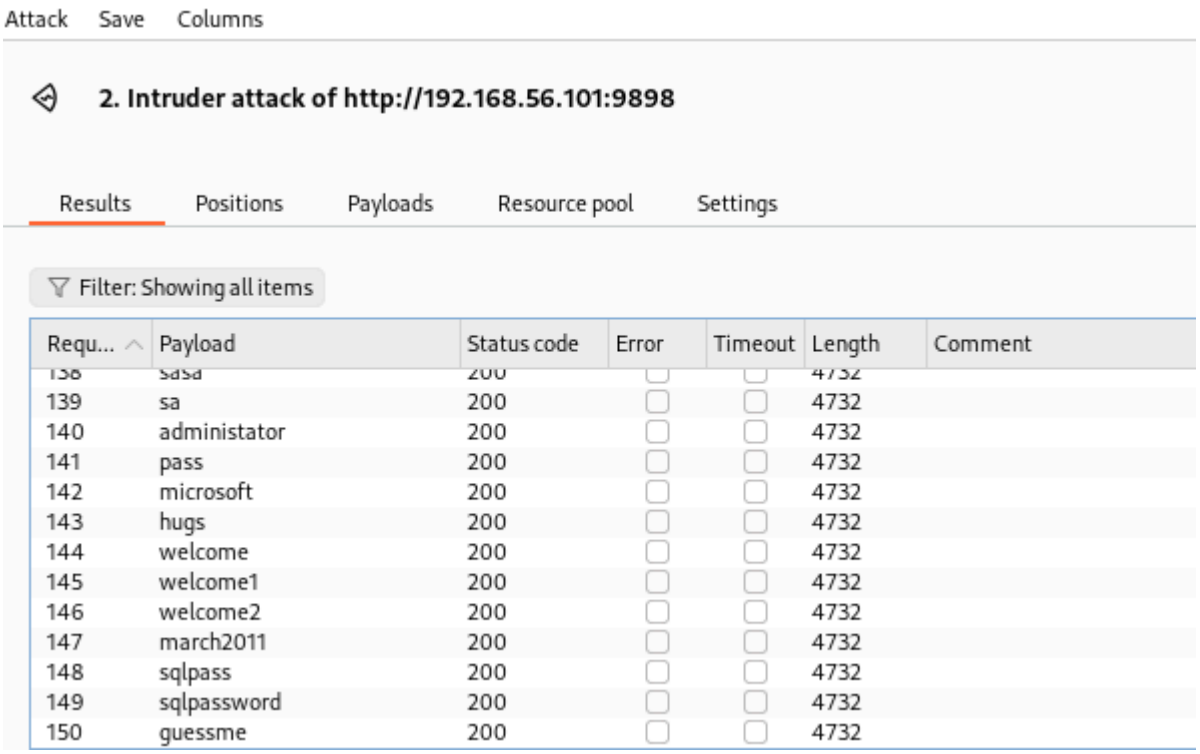
Una vez establecida la conexión, se accede a la página de login de la aplicación disponible en “http://192.168.56.101:9898/tannen/login”, donde se introducen credenciales falsas (“biff” como usuario y “hola” como contraseña).

Burp Suite con la opción Intercept ON activada, captura la petición POST de inicio de sesión, la cual incluye un token CSRF válido.



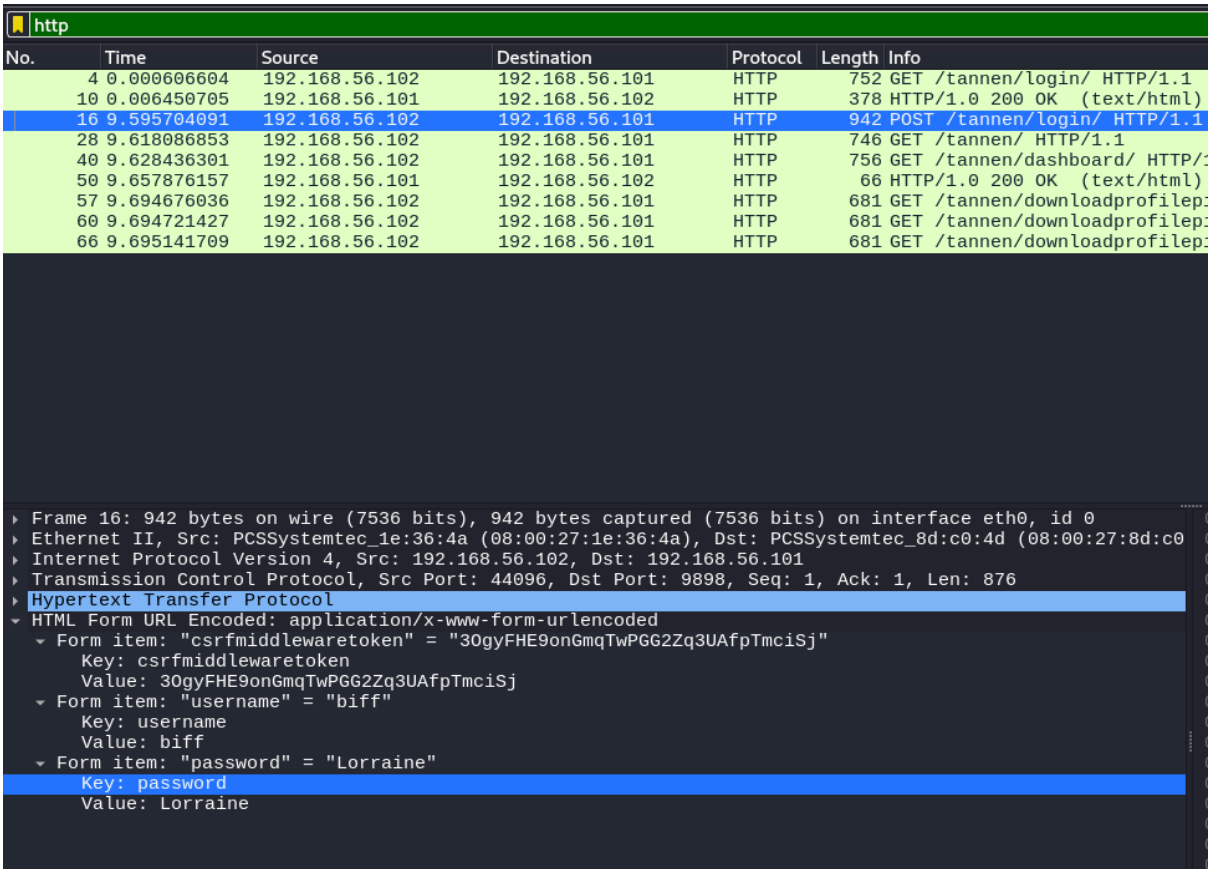
La petición interceptada se envía al módulo Intruder para su manipulación, seleccionando como objetivo el campo de contraseña y sustituyéndola por "§". Se configura un ataque de tipo Sniper, marcando la contraseña para ser modificada en cada intento y cargando una lista reducida de contraseñas comunes, como fasttrack.txt, debido a las limitaciones de la versión Community Edition de Burp Suite que impiden usar listas más grandes como rockyou.txt.

Una vez configurado el ataque, se ejecutan múltiples intentos de login de forma automatizada, respetando el token CSRF y la sesión activa. El ataque se completa con éxito, permitiendo probar diferentes contraseñas de forma automatizada sin que la aplicación aplique medidas de mitigación como bloqueos temporales o retardos entre intentos. Aunque no se logra recuperar la contraseña válida debido al uso de una lista limitada de contraseñas, el hecho de poder realizar el ataque confirma que el login es vulnerable a ataques de fuerza bruta.



Para mitigar esta vulnerabilidad, se recomendaría implementar bloqueos temporales tras varios intentos fallidos de login, añadir mecanismos de CAPTCHA en el formulario tras detectar múltiples fallos, introducir retardos progresivos entre intentos consecutivos y monitorizar los patrones de acceso para identificar y bloquear IPs que realicen intentos múltiples de autenticación.

9. Vulnerabilidad de cifrado HTTP



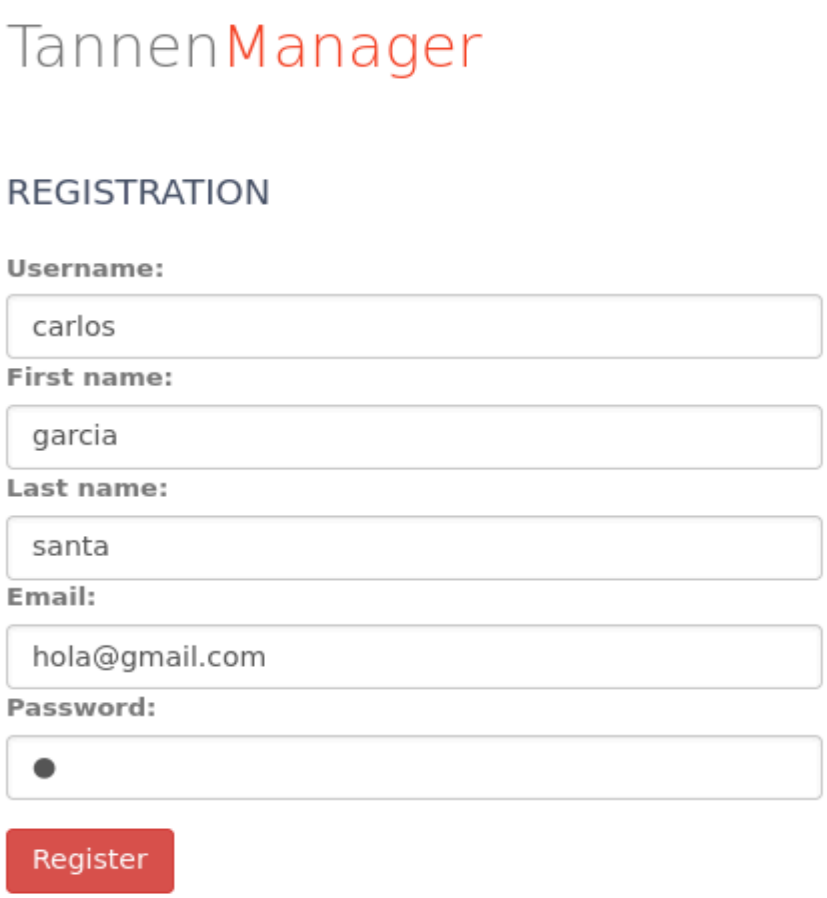
Se comprueba que las comunicaciones entre el cliente y el servidor se realizan utilizando el protocolo HTTP sin cifrado. Para verificarlo, se emplea la herramienta Wireshark, capturando el tráfico de red mientras se realiza un intento de login en la aplicación. Durante el análisis de las capturas, se observa que tanto el nombre de usuario como la contraseña, así como otros parámetros sensibles como tokens CSRF y cookies de sesión, se transmiten en texto claro, sin ningún tipo de protección criptográfica.

Es relevante mencionar que dado que esta aplicación forma parte de la práctica, es posible que el uso de HTTP sea intencionado para facilitar el análisis de vulnerabilidades o el desarrollo de la app. No obstante, en un entorno real de producción, la utilización de HTTP en lugar de HTTPS constituiría una vulnerabilidad crítica. La ausencia de cifrado expone la aplicación a ataques de tipo Man-in-the-Middle (MITM), facilitando la interceptación y robo de credenciales o de información sensible por parte de atacantes que se encuentren en la misma red que el usuario.

Sería necesario en un entorno real implementar el protocolo HTTPS mediante certificados válidos y actualizados, asegurando así la confidencialidad e integridad de los datos transmitidos entre el cliente y el servidor.

10. Vulnerabilidad de complejidad de contraseñas

Se observa que el sistema permite la utilización de contraseñas débiles tanto en el registro de nuevos usuarios como en el cambio de contraseñas existentes. No se aplican restricciones de longitud mínima, ni requisitos de uso de caracteres especiales, mayúsculas o cifras, lo que facilita la elección de claves fácilmente adivinables. En la captura se utiliza un “1” como contraseña, y el sistema permite el registro.



Este comportamiento incrementa notablemente el riesgo de ataques de fuerza bruta y de acceso no autorizado a las cuentas de los usuarios.

Sería necesario implementar una política de contraseñas que requiera una longitud mínima de al menos 8 caracteres y la inclusión de combinaciones de letras mayúsculas, minúsculas, números y símbolos especiales.