Actividad 2 – Deserialización Insegura

Auditoría Informática

Ingeniería en Desarrollo de Software

Tutor: Jessica Hernández Romero.

Alumno: Carlos Alberto Fuentes Mendoza

Fecha: 08-octubre-2023

Índice

Introducción

En esta segunda etapa vamos a conocer otra vulnerabilidad muy explotada en las aplicaciones web que es la deserialización. Para entender qué es la deserialización insegura, primero es necesario comprender qué quiere decir el verbo serializar en informática.

La serialización es un proceso que permite enviarle información sobre un código a un software. El término se utiliza para referirse a una sintaxis que permite almacenar comandos de programación, con el fin de comunicárselos a un software para que los ejecute. El problema radica en que algunas páginas web interpretan mensajes serializados de forma indiscriminada y esto le abre la puerta a la ejecución de código malicioso dentro de la aplicación.

Este fallo de ciberseguridad tan común se produce cuando, en términos de código, la aplicación tiene activada la función de deserializar las entradas que ingresa un usuario. Esto significa que, por medio de las entradas de texto de la página, un usuario podría enviar un objeto serializado (es decir, un código almacenado) y la aplicación web lo ejecutaría. Esto podría resultar en el robo o la manipulación de datos. De hecho, podría llevar en una ejecución remota de código: una de las instancias más graves de un ciberataque.

La deserialización insegura permite a un atacante manipular objetos serializados para pasar datos dañinos al código de la aplicación, e incluso, reemplazar un objeto serializado por un objeto de una clase distinta. Es decir, los objetos que estén disponibles para el sitio web serán deserializados e instanciados, independientemente de la clase que se esperaba. De hecho, es por ello que esta vulnerabilidad también se conoce como inyección de objetos. Muchos ataques de deserialización finalizan antes de que se complete la deserialización. Esto significa que el proceso en sí mismo puede iniciar un ataque, sin importar si la funcionalidad de la aplicación interactúa o no directamente con el objeto malicioso.

Descripción

Una empresa de software solicita realizar varias pruebas de seguridad en páginas web que no cuentan con los candados de seguridad.

Para esta segunda etapa, se pide realizar una prueba de deserialización insegura en una página específica. Esta debe ser mediante las cookies. Para lograrlo, se deberá de utilizar el programa Burp Suite Community Edition. El objetivo de esta prueba es que se inicie sesión como un usuario normal y luego pasar a modo administrador a través de las cookies.

Para esta actividad 2, con la ayuda de la plataforma PortSwigger, aprenderé a realizar el ataque a una página proporcionada por ellos. En ella, se deberá iniciar sesión con las credenciales que se proporcionan, las cuales son para usuarios normales; no obstante, a través de las cookies, se deberá entrar al modo administrador.

Cabe mencionar que este laboratorio utiliza un mecanismo de sesión basado en serialización. Por ende, es vulnerable a la escalada de privilegios. En consecuencia, hay que editar el objeto serializado en la cookie de sesión para aprovechar esta vulnerabilidad y obtener privilegios administrativos. Finalmente, el objetivo es eliminar la cuenta de Carlos.

Hay que iniciar sesión en la propia cuenta con las siguientes credenciales:

- Usuario: wiener

- Contraseña: peter

Justificación

Un atacante puede abusar del proceso de deserialización si se deja inseguro. Un atacante puede inyectar objetos serializados hostiles a una aplicación web, donde la computadora de la víctima inicializaría la deserialización de los datos hostiles. El atacante puede entonces cambiar el ángulo de ataque, haciendo que la deserialización insegura sea el punto de entrada inicial a la computadora de la víctima.

Prevenir ataques de deserialización es de suma importancia debido a los riesgos significativos que estos ataques pueden representar para la seguridad de una aplicación o sistema. A continuación, se mencionan algunas razones clave por las que es crucial prevenir los ataques de deserialización:

- Protección de datos confidenciales: Los ataques de deserialización pueden permitir a un atacante manipular y acceder a datos confidenciales almacenados en la aplicación o sistema. Esto podría incluir información financiera, datos personales de usuarios, credenciales de acceso y otros datos sensibles.

- Integridad de la aplicación: Un atacante que logra realizar una deserialización no autorizada podría comprometer la integridad de la aplicación o sistema, lo que podría resultar en la ejecución de código malicioso, la corrupción de datos o la alteración de la funcionalidad de la aplicación.

- Prevención de ejecución de código malicioso: Algunos ataques de deserialización pueden permitir que un atacante ejecute código malicioso en el servidor o en el sistema cliente. Esto podría llevar a una variedad de problemas de seguridad, incluidos ataques de inyección de código y escaladas de privilegios.

- Protección de la reputación: Los ataques de deserialización exitosos pueden dañar la reputación de

una organización, ya que pueden resultar en la divulgación pública de datos sensibles o en la explotación de vulnerabilidades de seguridad.

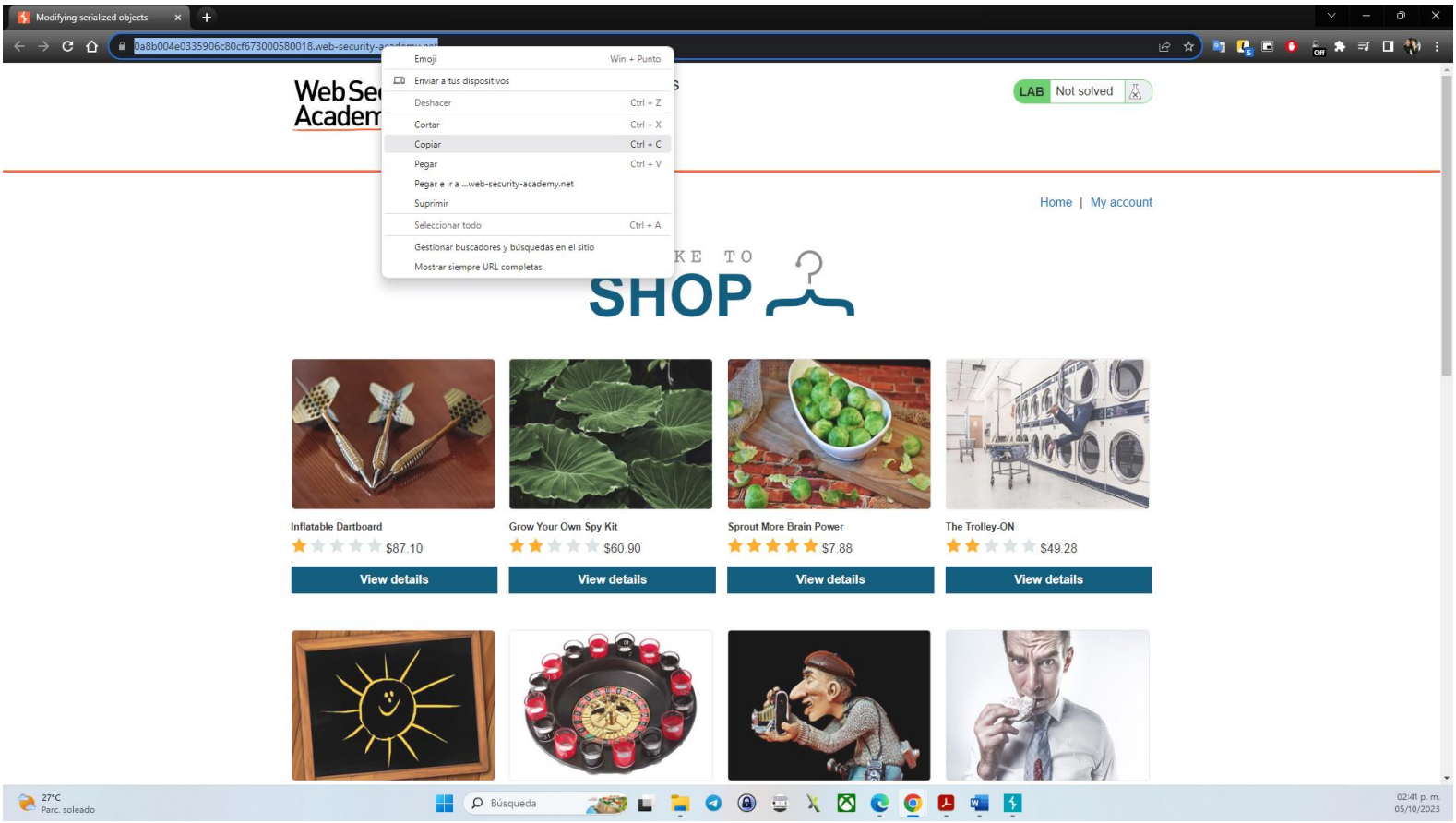
- Cumplimiento de regulaciones y leyes: En muchos lugares, existen regulaciones y leyes que requieren la protección de datos confidenciales y la implementación de medidas de seguridad adecuadas. La falta de prevención de ataques de deserialización puede resultar en incumplimiento de estas regulaciones y en posibles sanciones legales.

- Continuidad del negocio: Un ataque exitoso de deserialización puede interrumpir las operaciones normales de una aplicación o sistema, lo que puede tener un impacto negativo en la continuidad del negocio.

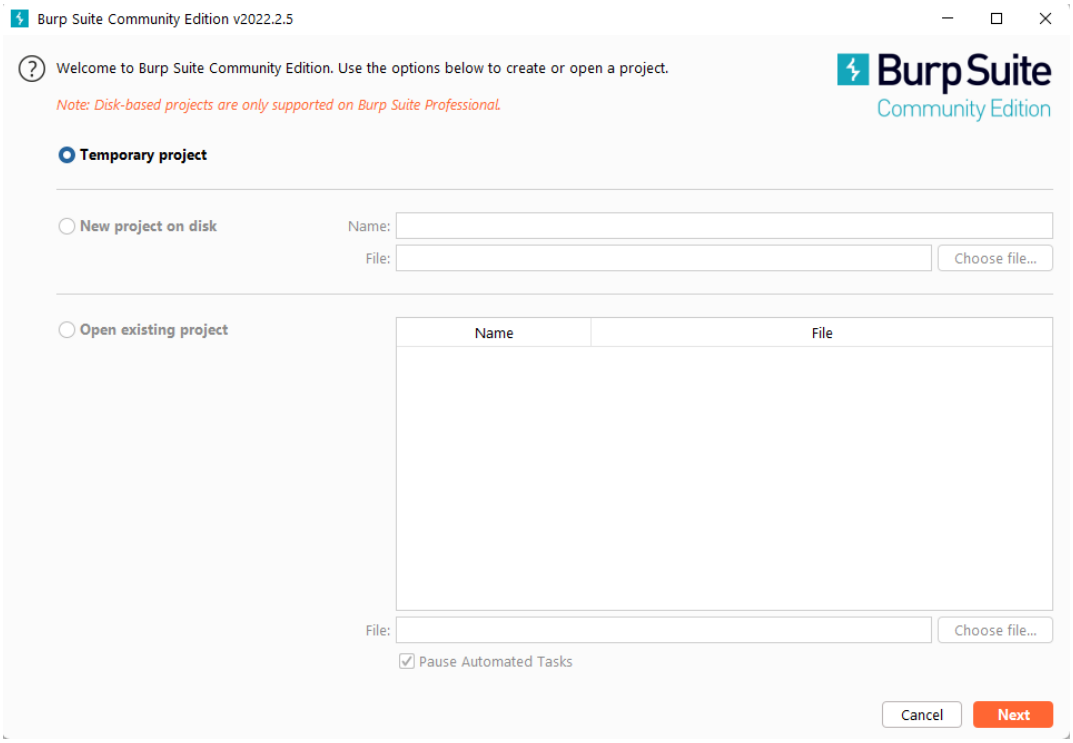
En resumen, prevenir ataques de deserialización es esencial para proteger la confidencialidad, integridad y disponibilidad de los datos y servicios de una aplicación o sistema. La implementación de medidas de seguridad adecuadas, como la validación de datos de entrada y la limitación de privilegios durante la deserialización, es fundamental para mitigar los riesgos asociados con estos ataques y mantener un entorno de desarrollo seguro.

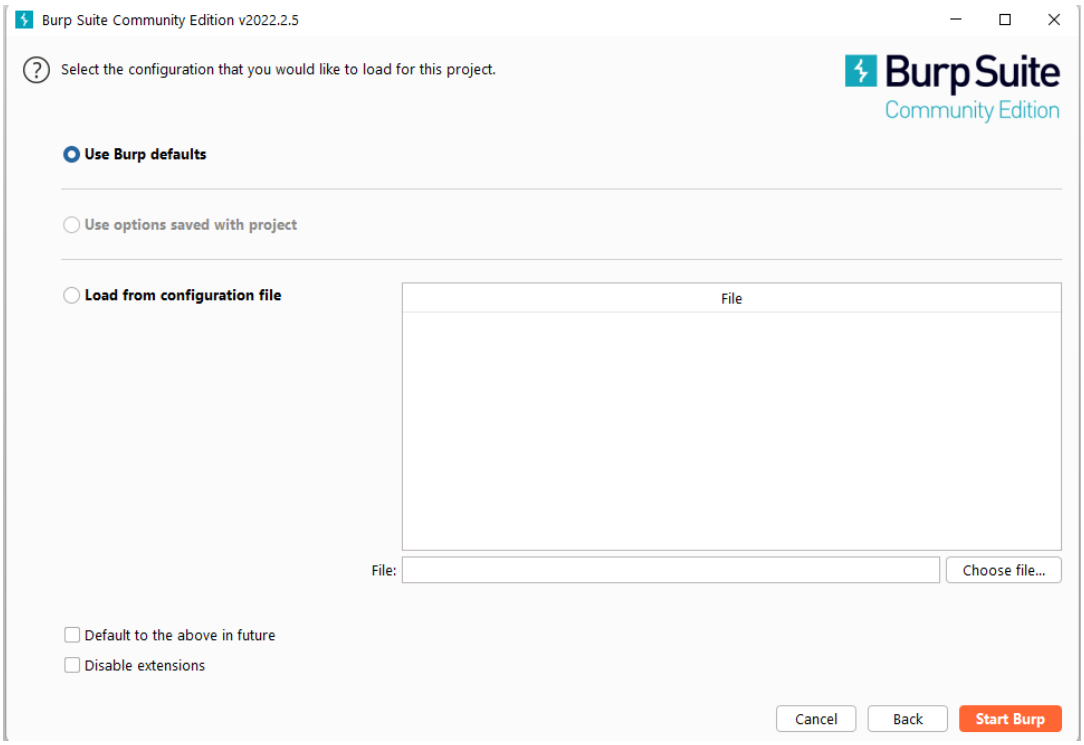
Ataques del sitio

Para comenzar este ataque, nos dirigimos al laboratorio de la pagina y copiamos la dirección url del sitio web.

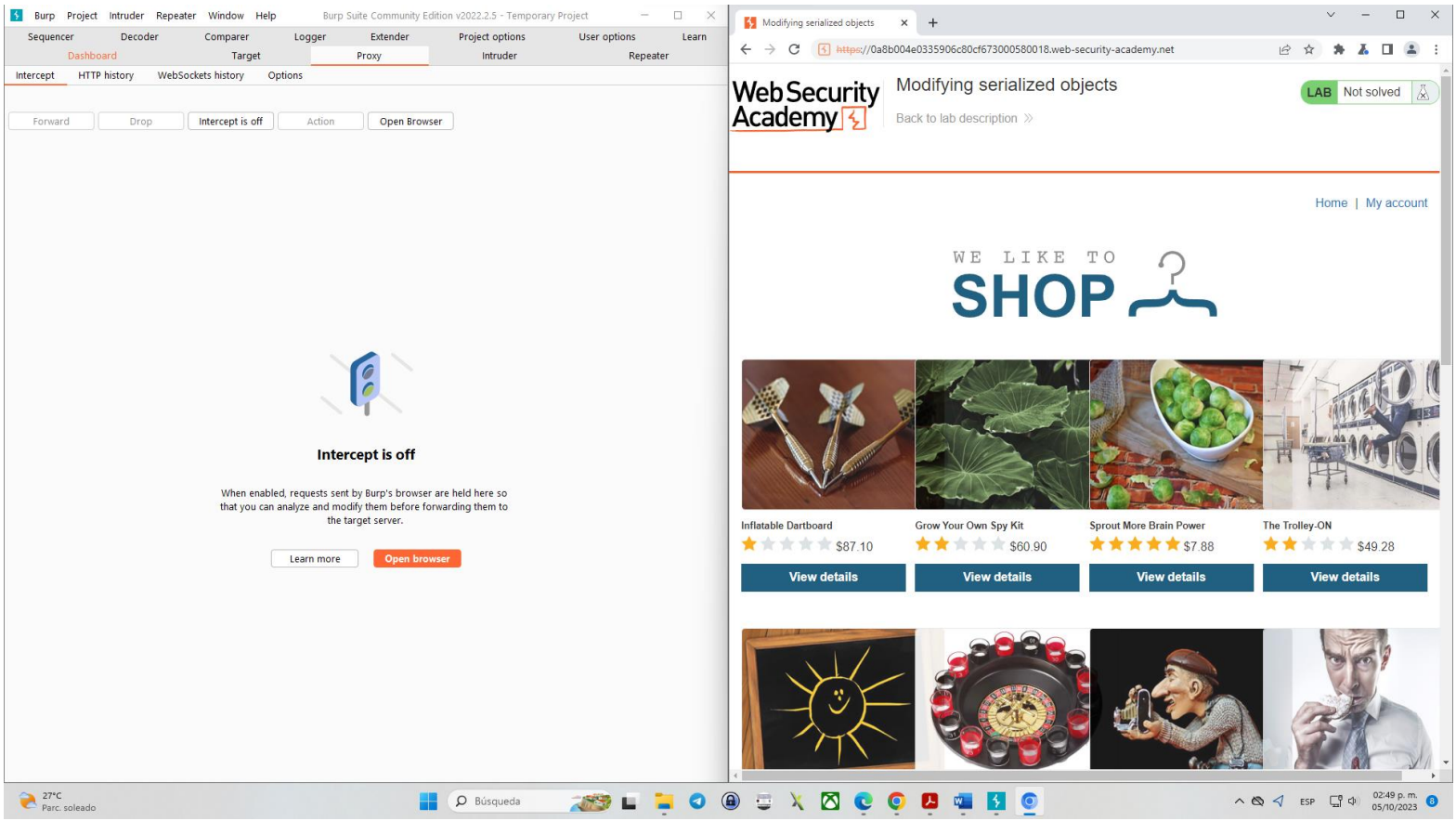


Enseguida pegamos el enlace de la página en la aplicación Burp Suite, para esto, se crea un proyecto temporal con las opciones default.





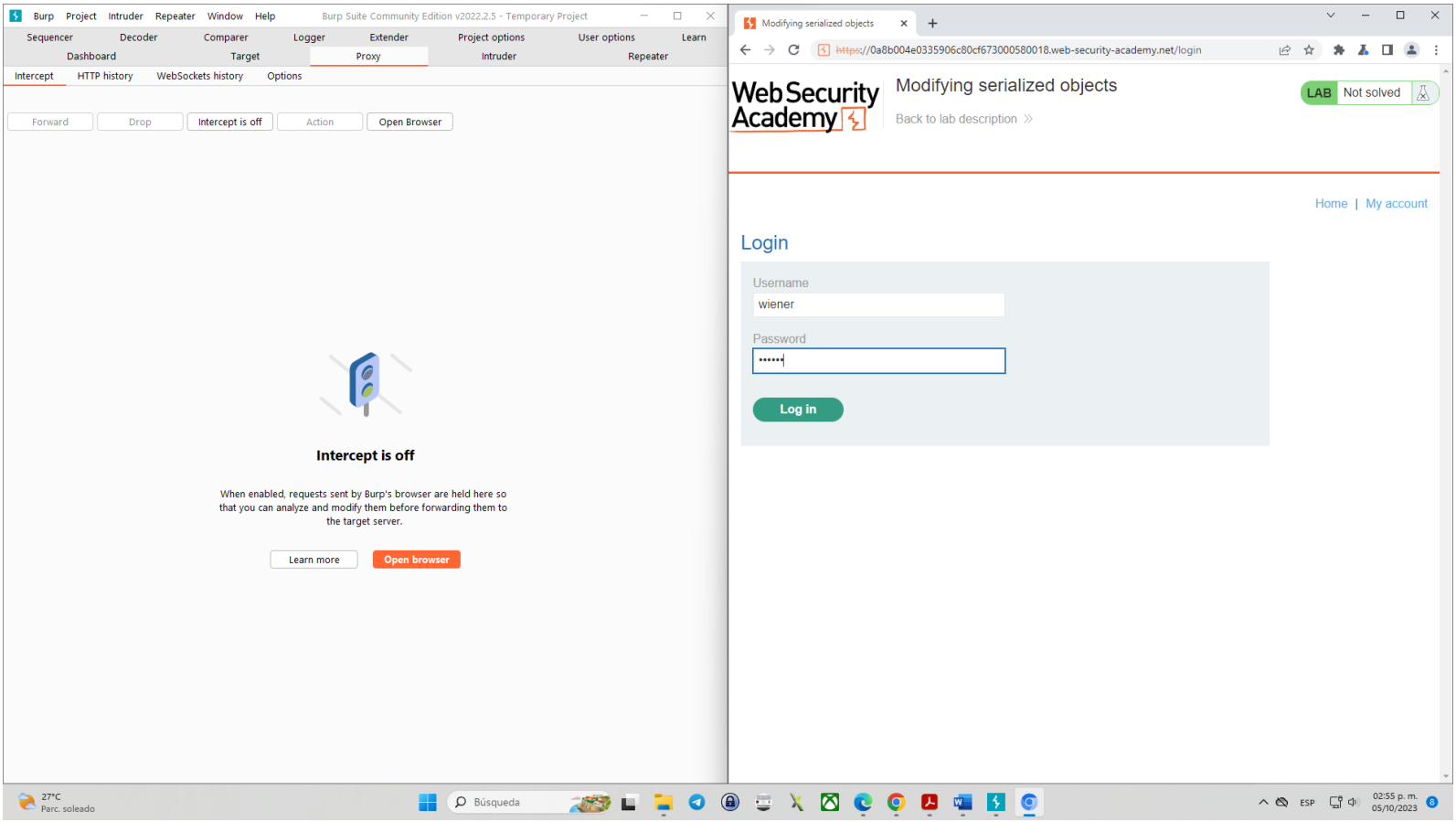
Una vez abierto el panel de trabajo de la aplicación, nos dirigimos a la sección proxy, en donde dando clic en el botón open browser, se abrirá el navegador de la aplicación y pegaremos la url que copiamos.



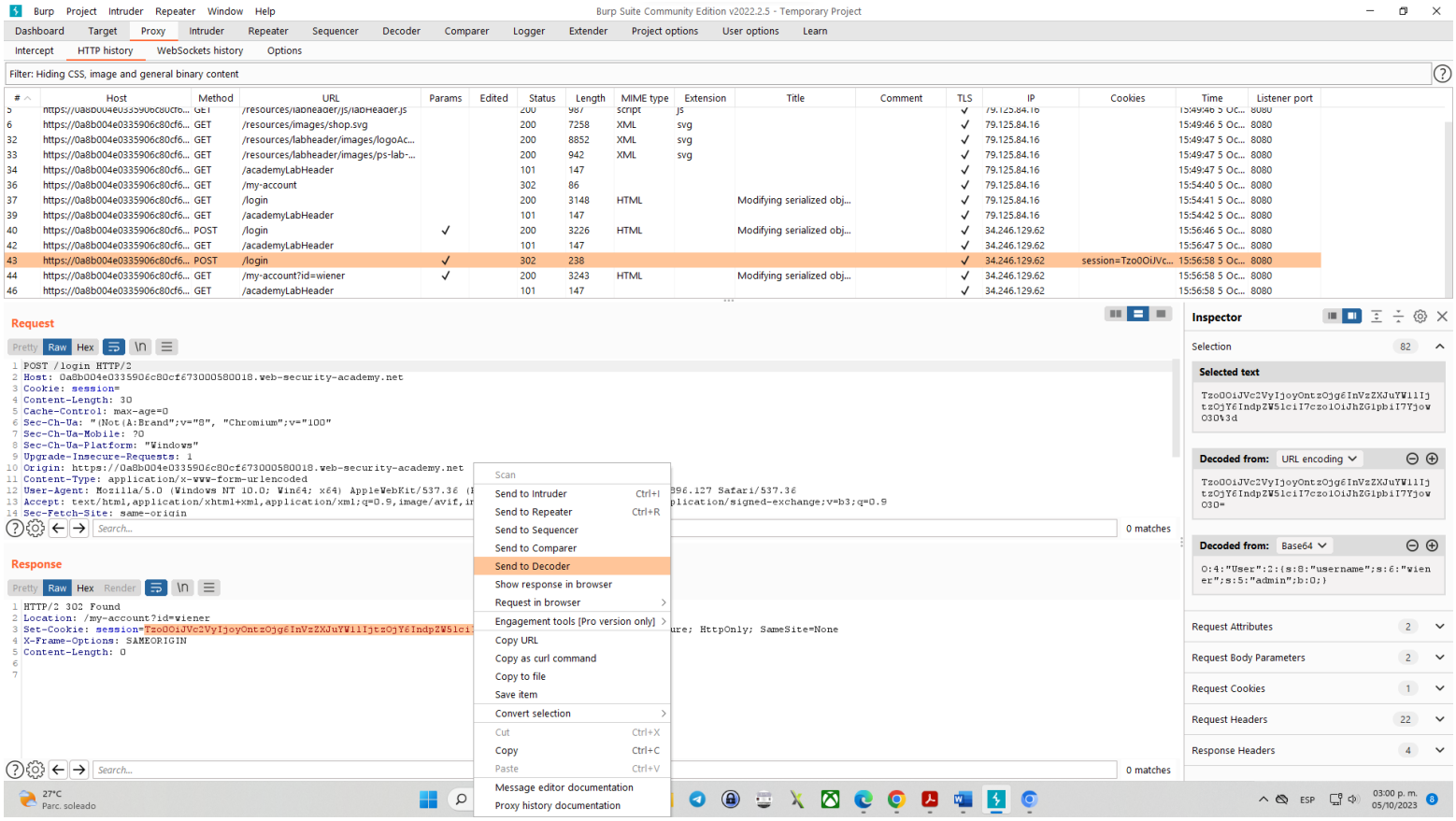
Procedemos entrando al sitio web con las credenciales proporcionadas en el PDF de la actividad.

Usuario: wiener

Contraseña: peter

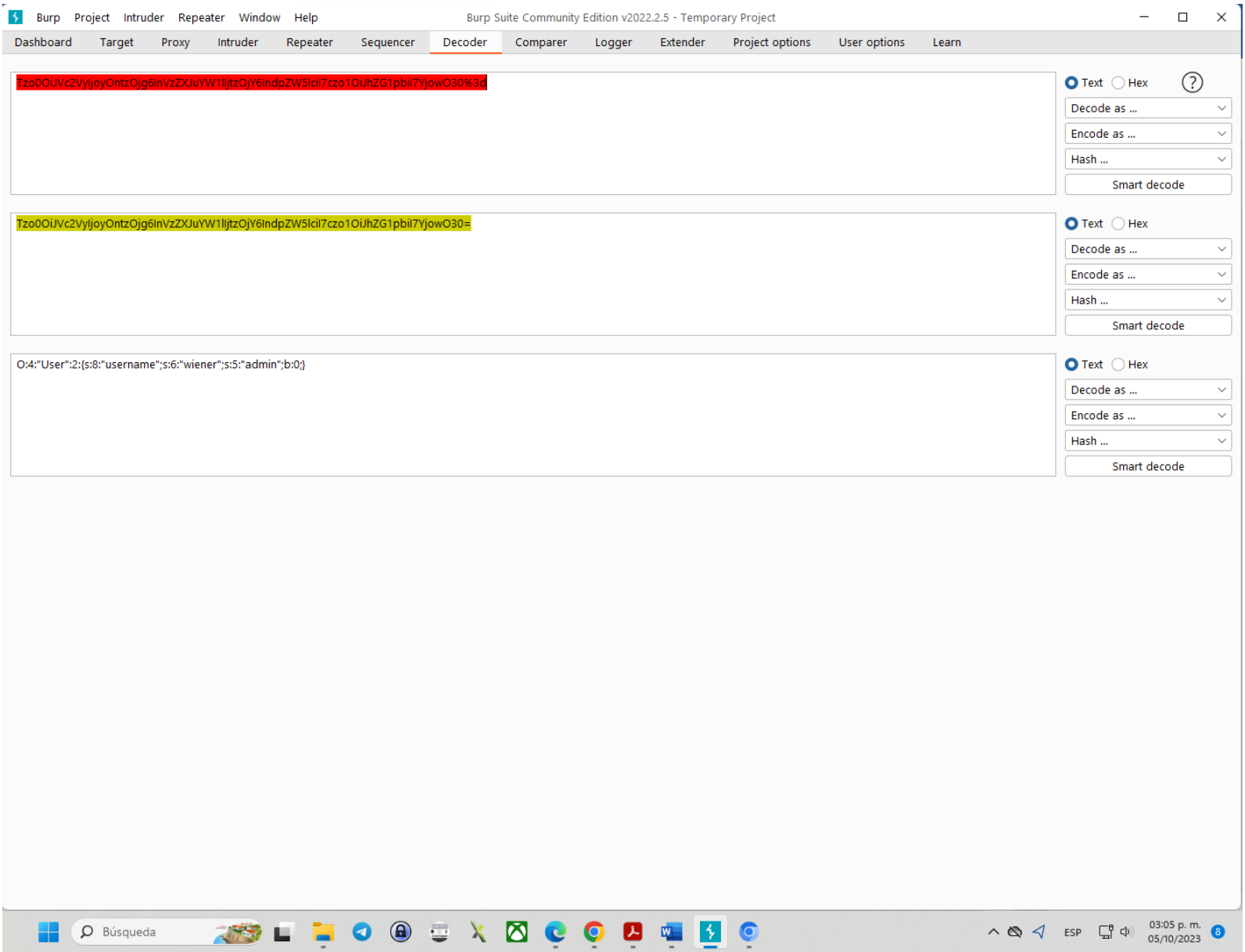


Nos dirigimos nuevamente a la sección Proxy en la pestaña HTTP history, en donde buscamos en el encabezado URL, la dirección /login con el método POST, aquí seleccionamos la Cookie session (las letras que aparecen en rojo) y enviamos a Decoder.



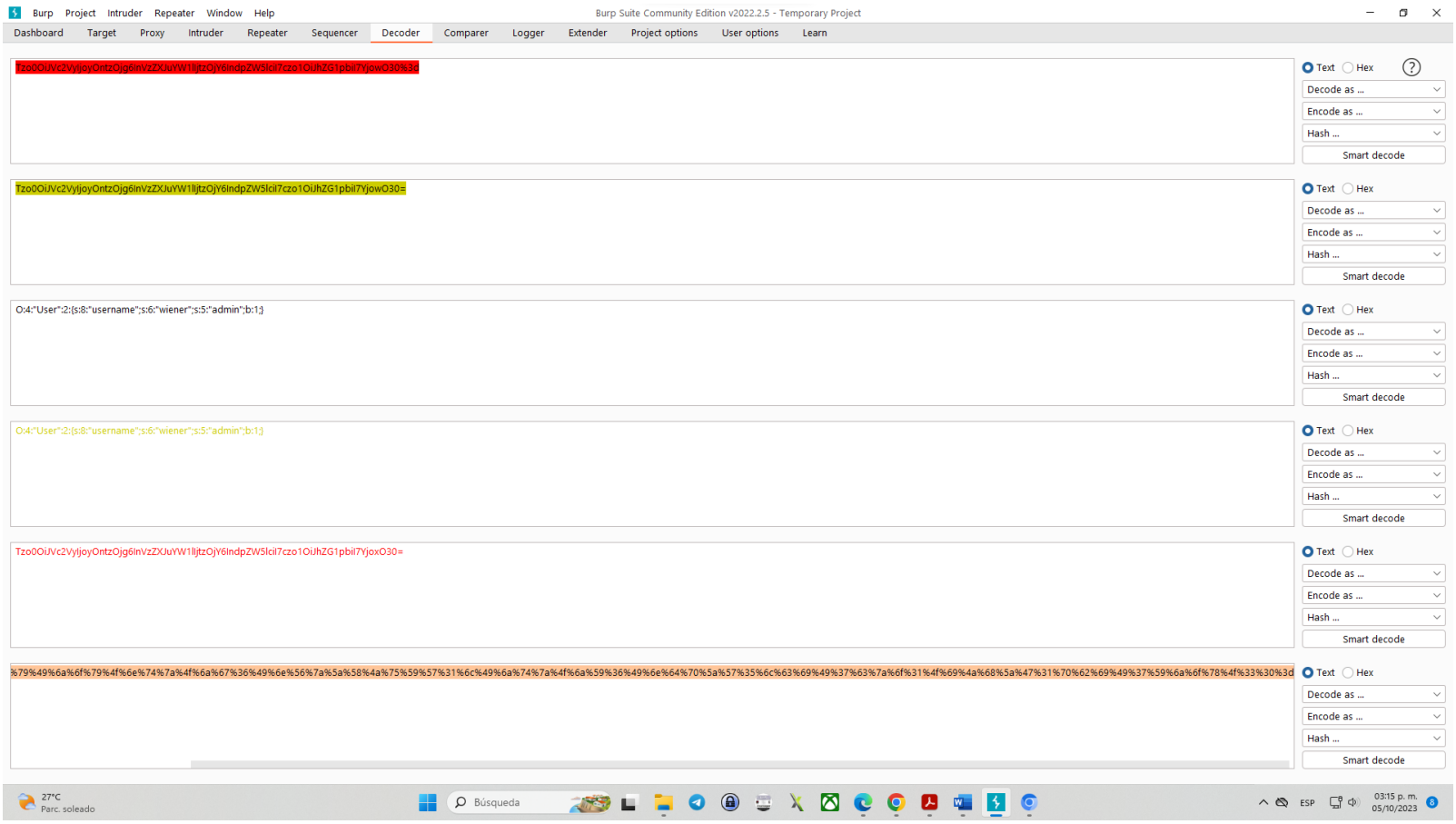
En la siguiente captura se observa la pantalla decoder, en esta sección decodificaremos la cookie, para esto, en la primera opción elegimos decodificar a url y en la segunda a decodificar a base 64.

En la tercera opción, ya tenemos la información serializada, esto lo comprobamos ya que al final de la línea aparece “b:0;” el cual el cero significa que este usuario no tiene privilegios de administrador.



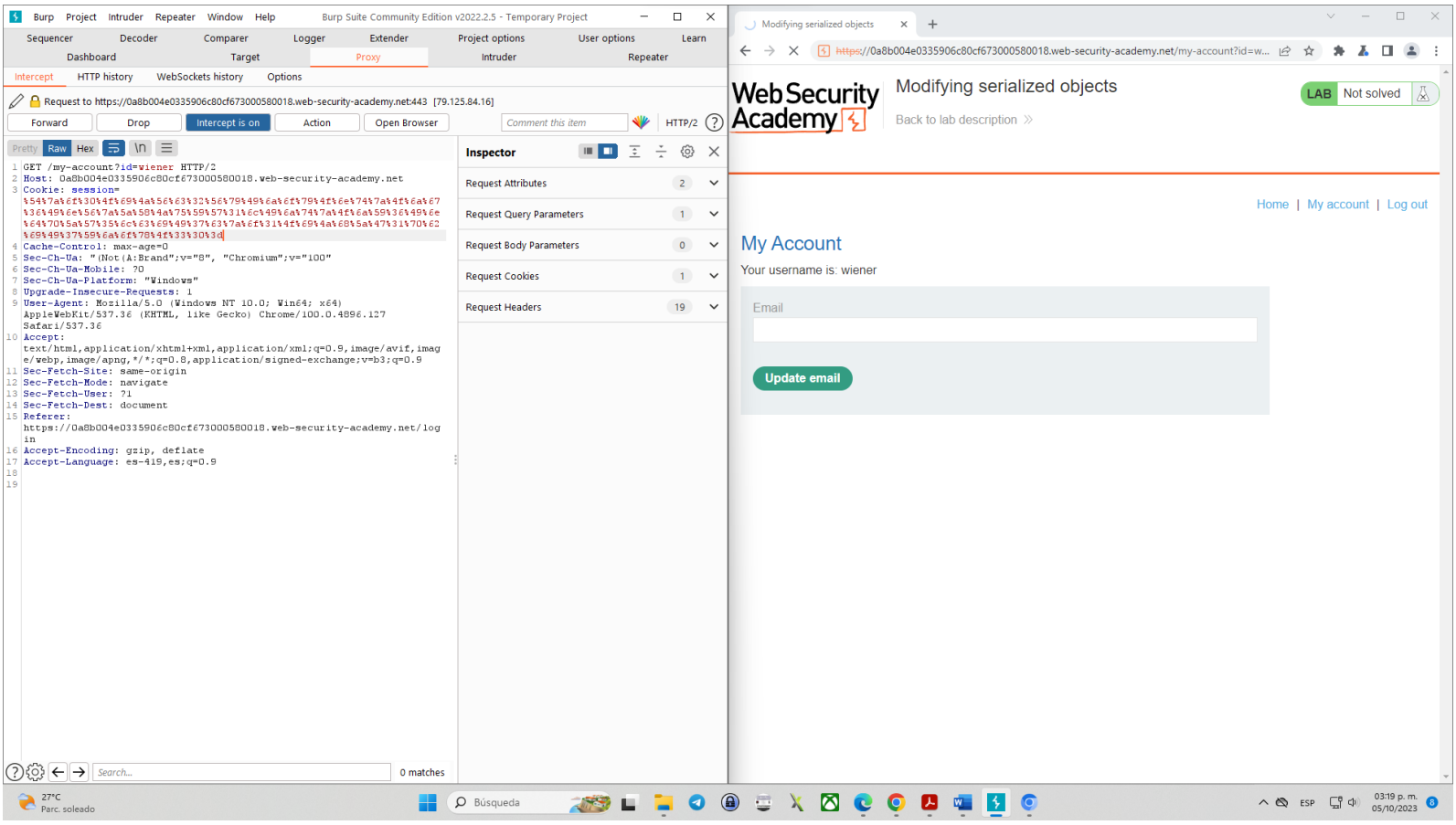
En la siguiente captura se puede observar que se dan privilegios de administrador cambiando el “0” al final de la información serializada por el “1”, para posteriormente codificarla a base 64 y por último a url.

Seleccionamos y copiamos toda la información del último renglón ya que es la que tiene permisos de administrador.

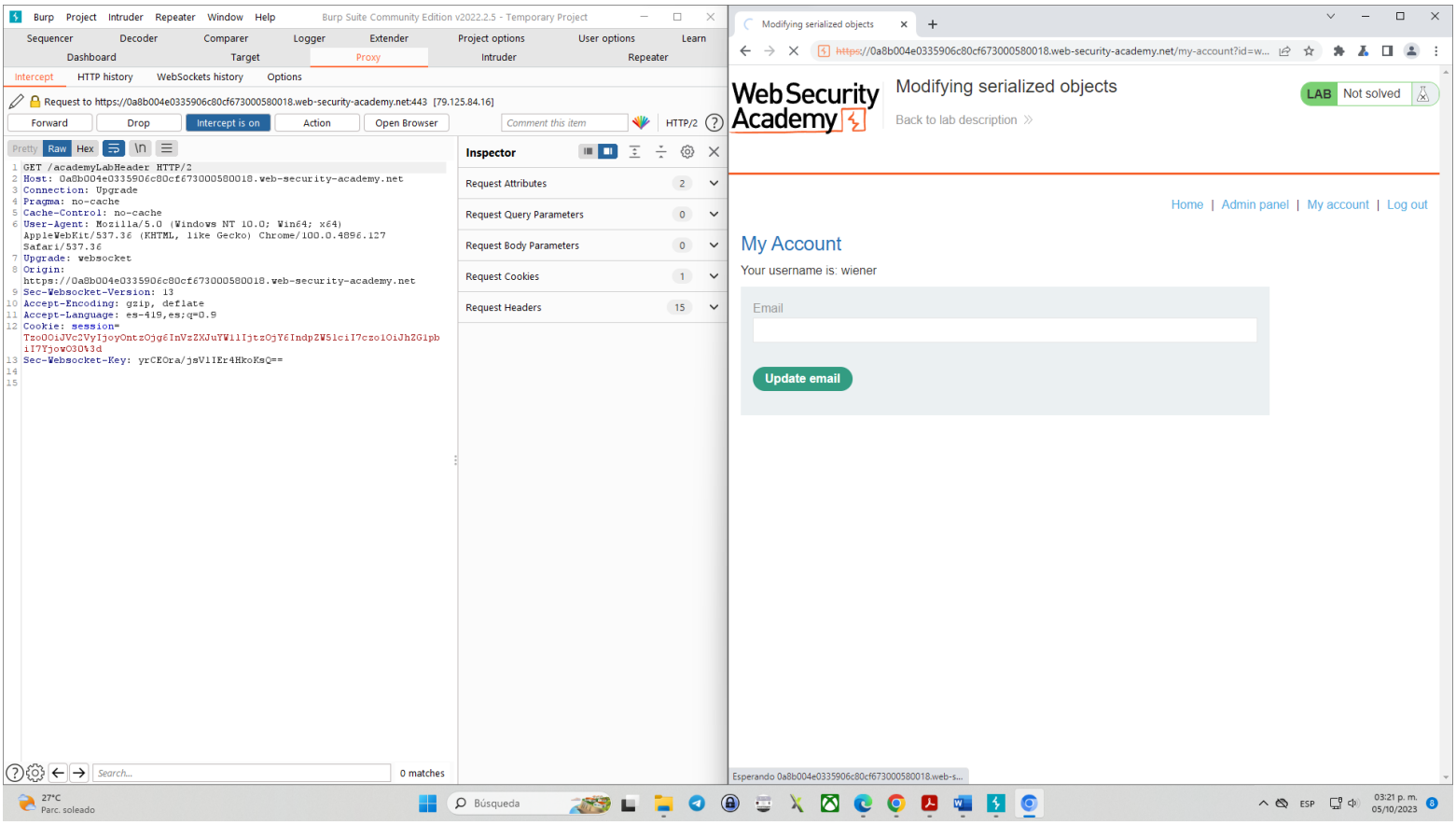


Procedemos a pasar nuevamente a la opción de proxy y encendemos el interceptor, posteriormente actualizamos el navegador para que nos dé nuevamente la información de la sesión del usuario.

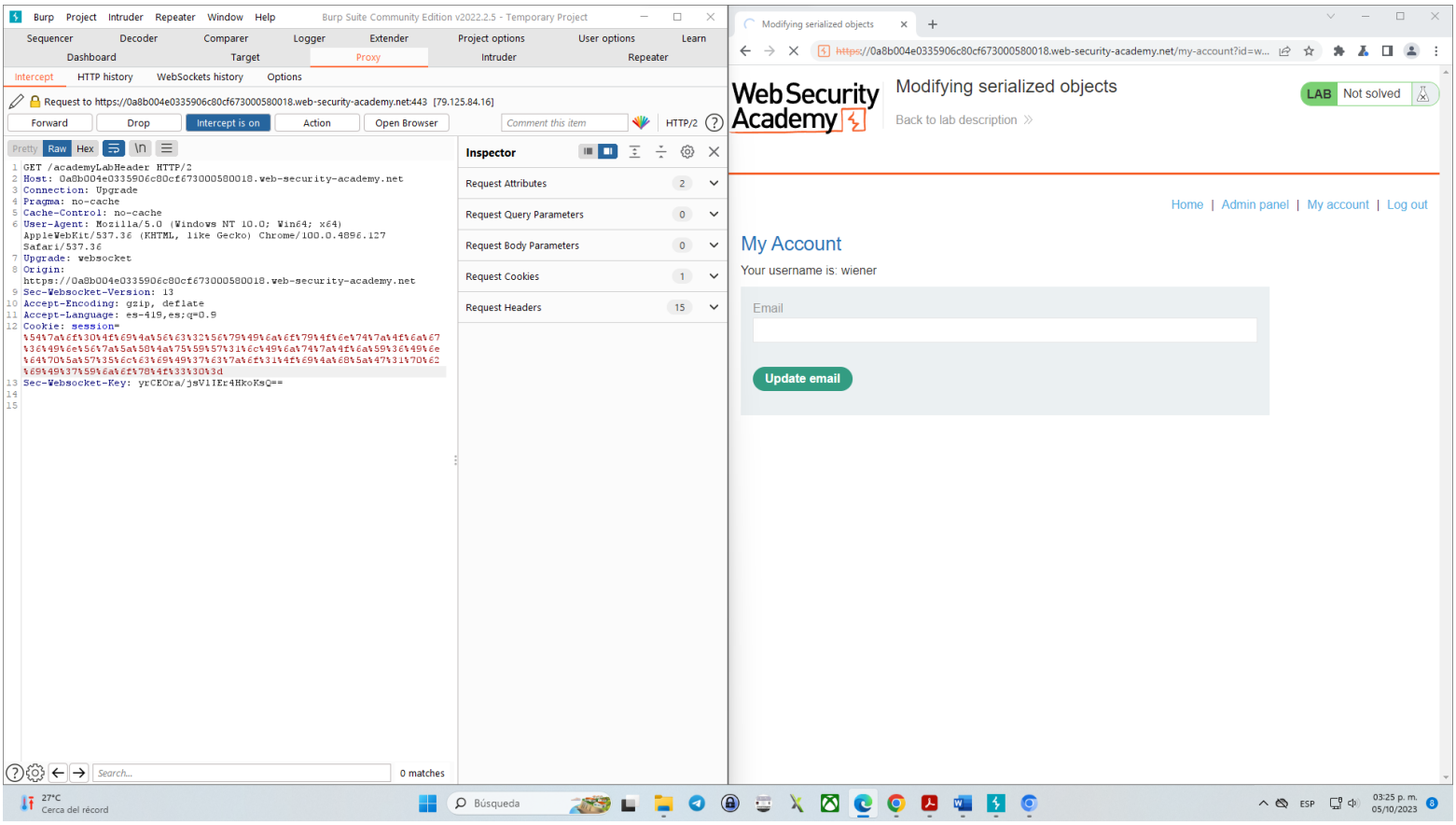
Continuamos cambiando la información de sesión del usuario (las letras en rojo) por las que codificamos con permisos de administrador.



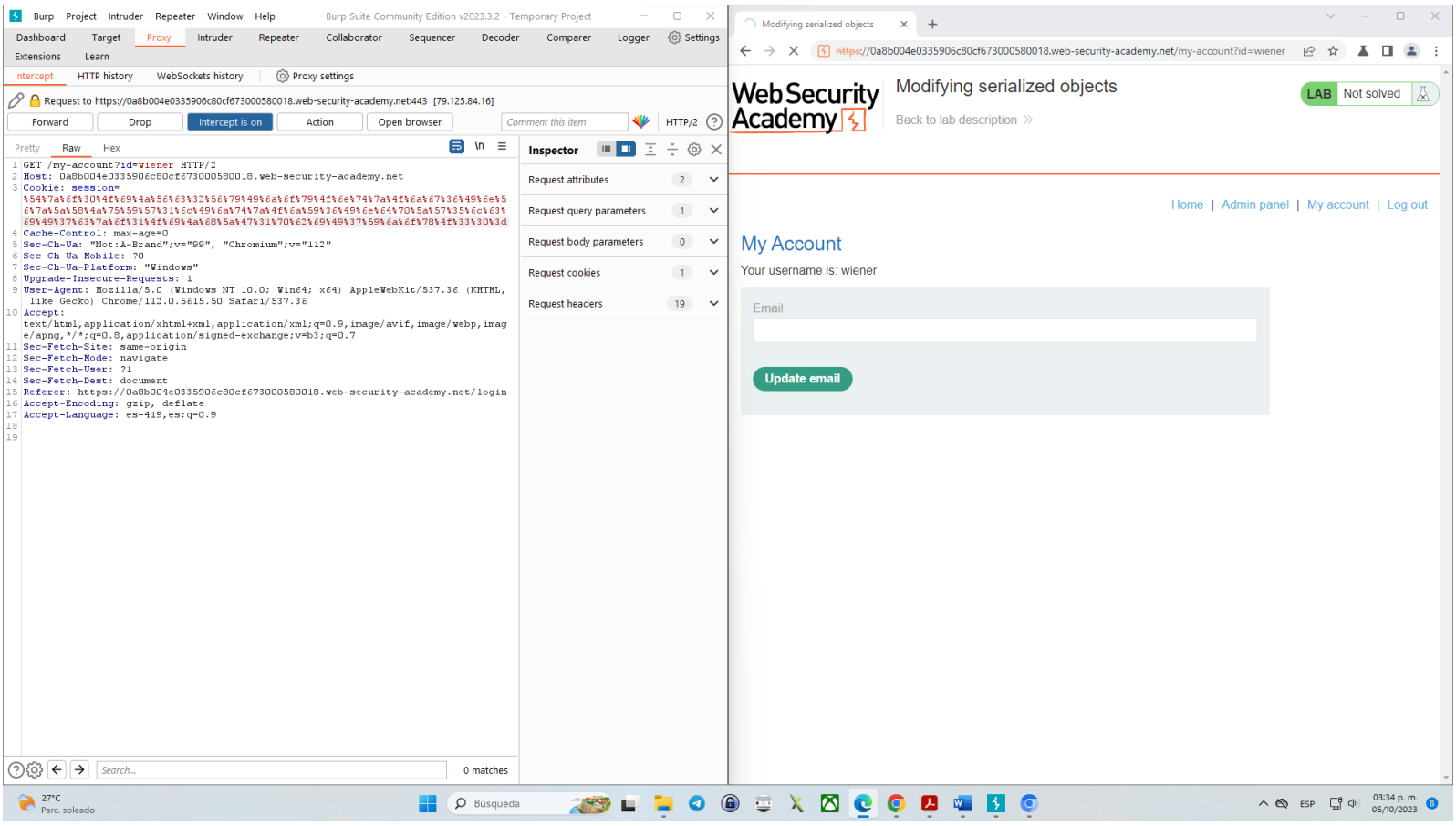
Como se puede observar en la siguiente captura, en las opciones de usuario del navegador aparece una nueva opción con el título “Admin panel”



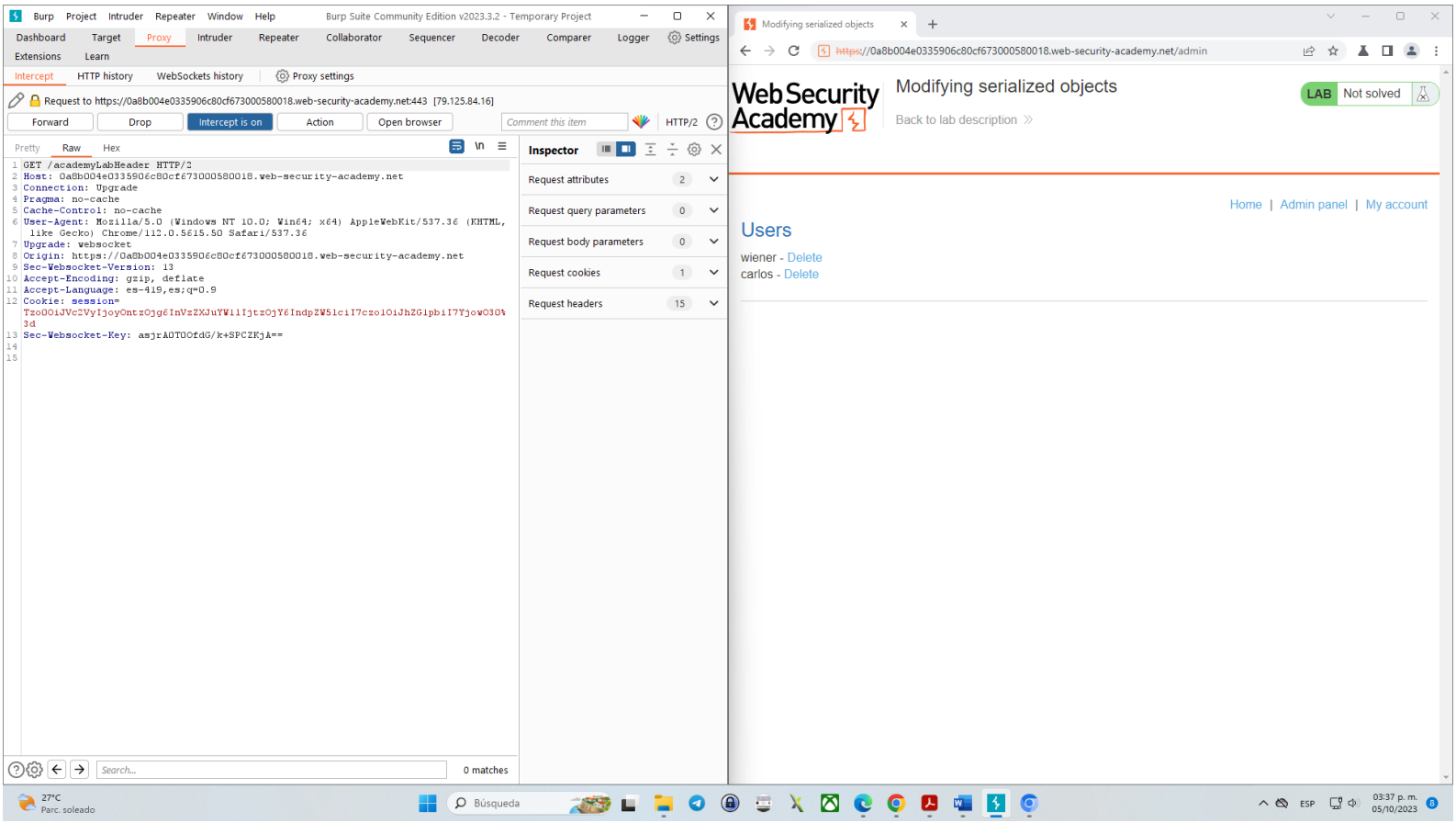
Procedemos a dar clic en la nueva opción y cambiamos nuevamente la cookie session(letras rojas) con la que tenemos con permisos de administrador y damos clic en el botón forward.



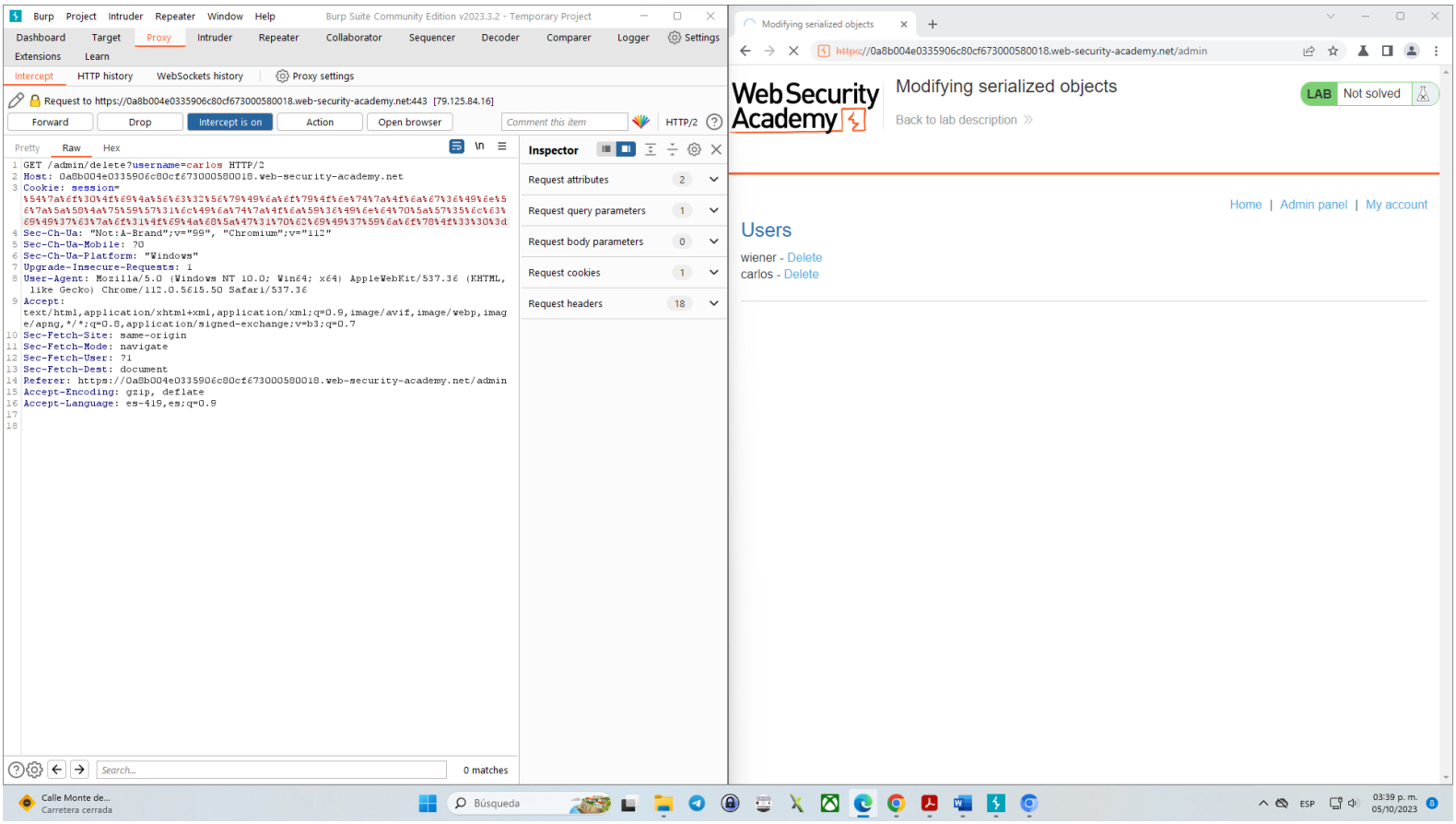
Procedemos a cambiar nuevamente la cookie session(letras rojas) con la que tenemos con permisos de administrador y damos clic en el botón forward y de igual forma damos clic en el botón forward.



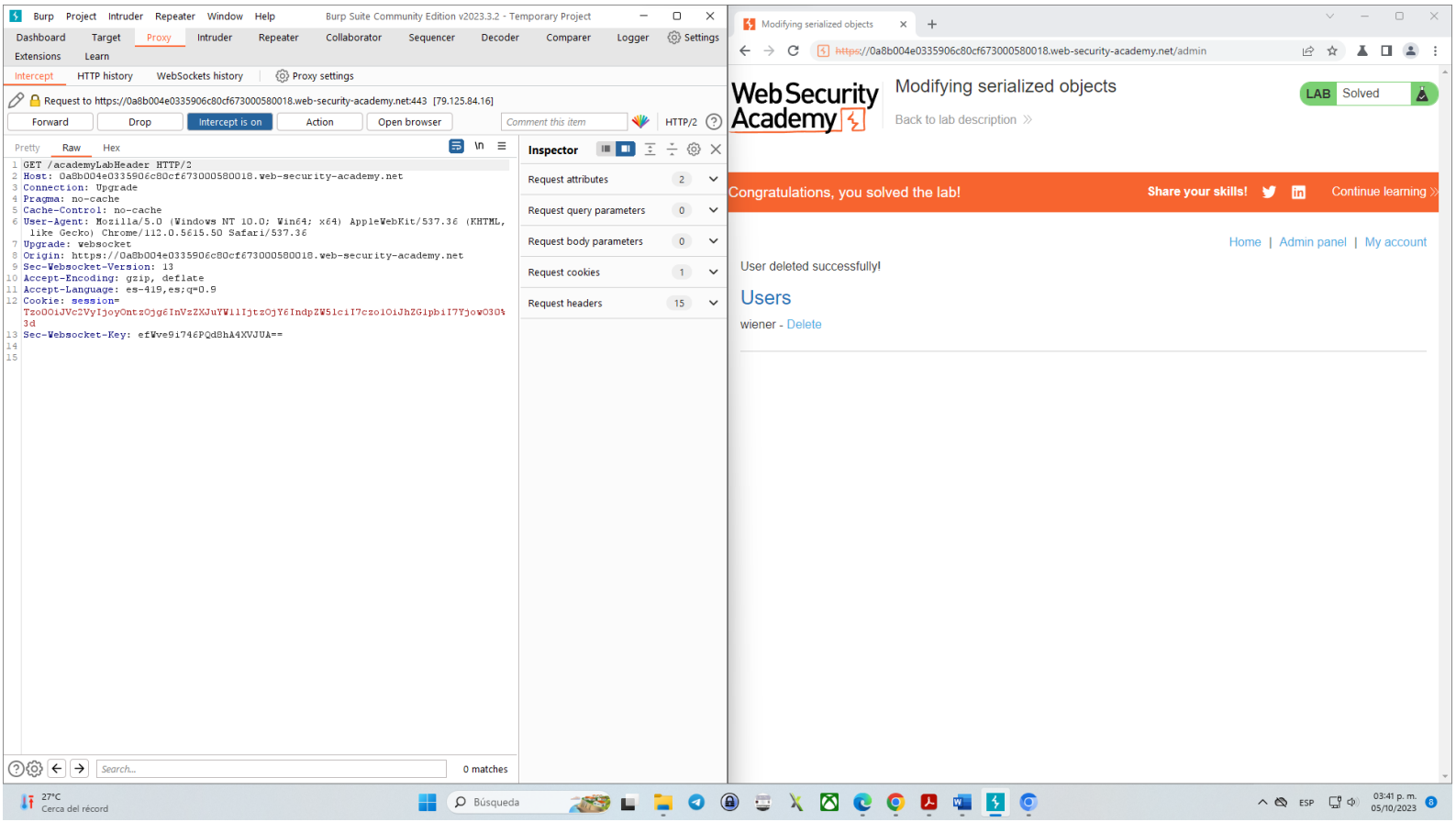
En la siguiente captura se observa como en el navegador, ahora aparece una interfaz que solo se tiene acceso con permisos de administrador, en la cual se pueden muestran los usuarios registrados.



A continuación, eliminaremos el usuario Carlos, para esto, damos clic en la opción Delete y enseguida pegaremos nuevamente la cookie session con los permisos de administrador y damos clic en forward.



Como podemos observar en la siguiente pantalla, el usuario Carlos se ha eliminado perfectamente.



A continuación, se comparte el link de acceso a la actividad en GitHub

https://github.com/charlyfu/Auditoria_informatica

Conclusión

En esta actividad conocí algo que jamás en mi vida había leído, ni escuchado, que es la deserialización, aprendí que es fallo de ciberseguridad de los más explotados en aplicaciones web y de igual forma aparece en el ranking de OWASP Top 10. También aprendí que esta vulnerabilidad ocurre durante el momento en el que se transforman los datos serializados a un objeto, y en este momento un atacante puede abusar de la lógica de la aplicación, y realizar ataques de denegación de servicio (DoS), omitir autenticaciones o incluso ejecutar código de malicioso de forma remota. Ahora entiendo que evitar estos riesgos es de suma importancia para una empresa principalmente, ya que podría poner en peligro a los usuarios de la plataforma y eventualmente generar una brecha costosa para el negocio. Creo firmemente que esta vulnerabilidad deben conocerlo todos los auditores de seguridad al igual que todos los desarrolladores web.

Por otra parte, también aprendí como aprovechar esta vulnerabilidad en el laboratorio que nos proporcionaron para realizar esta actividad, realmente quede sorprendido como, con ayuda de la aplicación Burp Suite Community, fue muy fácil obtener privilegios de administrador por medio de la serialización y deserialización de las cookies, tan solo fue decodificarla como url y luego base 64 para luego colocar el “1” que entiendo que en la BD debería de ser 0 para usuario normal y 1 para administrador, y con esto tener todos los privilegios de administrador para poder eliminar cuantas de usuario.

Referencias

Alex. (2021, 29 diciembre). ¿Qué es la deserialización insegura? Definición de. Krypton Solid. https://kryptonsolid.com/que-es-la-deserializacion-insegura-definicion-de-krypton-solid/

Vaquero, B. A. (2023, 22 julio). Deserialization PHAR: Guía sobre este tipo de vulnerabilidad. LimpiatuWeb. https://limpiatuweb.com/blog/deserialization-phar/

Cantelli, F. (2022, 27 octubre). Qué es Insecure Deserialization y cómo prevenirla. Hackmetrix Blog. https://blog.hackmetrix.com/insecure-deserialization/

KeepCoding, R. (2023c, abril 10). ¿Qué es la deserialización insegura? KeepCoding Bootcamps. https://keepcoding.io/blog/que-es-la-deserializacion-insegura/