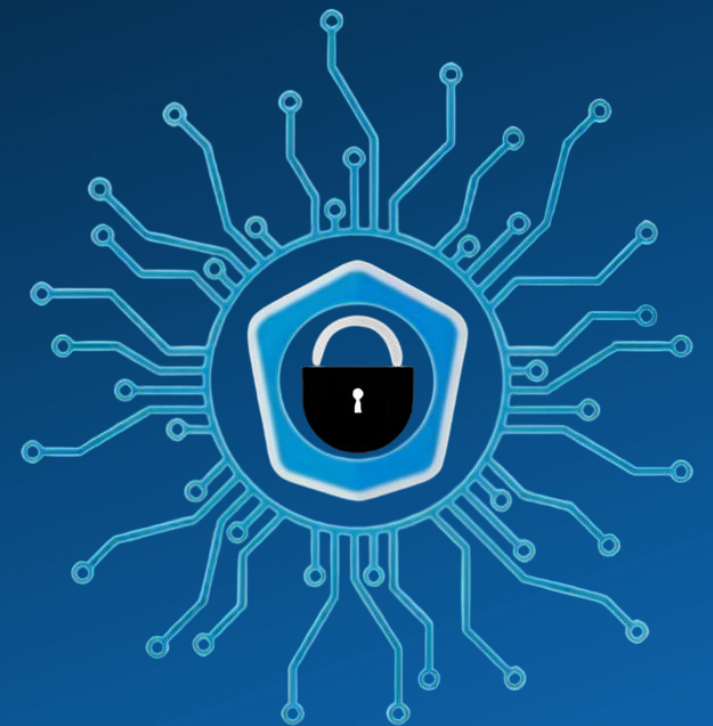


QUESTION CHALLENGE WALKTHROUGH

**WRITTEN BY:
TALEEN SKAFI**



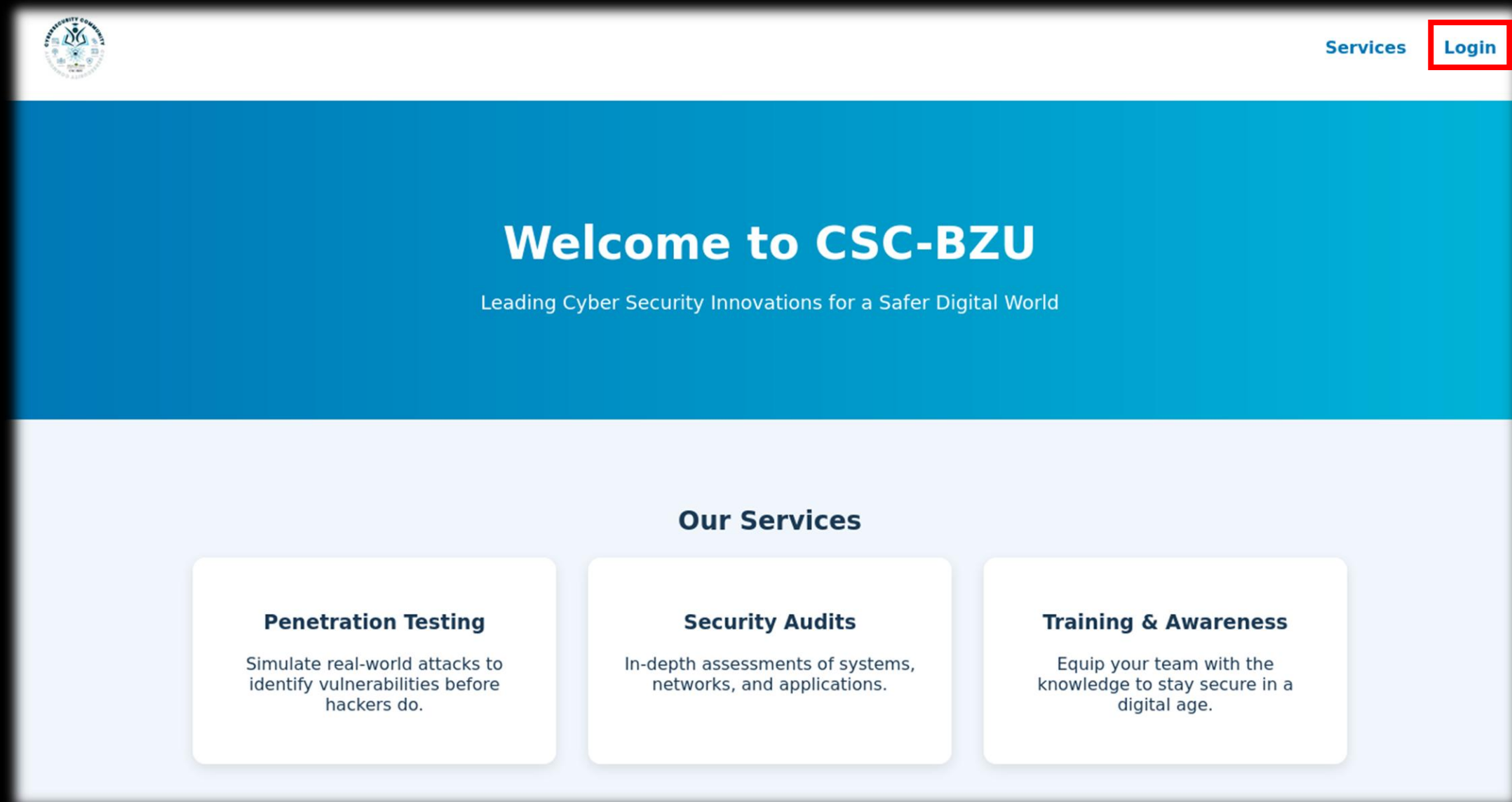
BZU - CSC

Welcome to **Guession**, a web security challenge that dives into the risks of insecure session management and privilege escalation through predictable identifiers. In this scenario, you start with valid credentials for a low-privileged user named Jack. After logging in, you notice something unusual, the URL contains a sid parameter. Curious, you decode it and find a format like session_1001, hinting at a simple, sequential session ID system. With just a small change (switching it to session_1000) , you suddenly gain access to the admin account. No password guessing, no login bypass , just a well-timed observation and a predictable token. This challenge is a reminder of how critical it is to handle sessions securely, because even small implementation flaws can lead to full compromise.





When we open the challenge, we see a web page with some services and a login button. We click on the login button to discover the login portal.



We're now on the login portal, so let's enter the provided credentials for **Jack** to sign in.



The login form is centered on a light blue background. It features the CSC-BZU logo at the top, which is a circular emblem with a stylized 'C' and 'B' and the text 'CYBERSECURITY COMMUNITY' and 'CSC-BZU'. Below the logo is the title 'CSC-BZU Login' in a bold, dark blue font. There are two input fields: the first is for the username, containing the text 'jack', and the second is for the password, represented by ten black dots. Both input fields have a yellow background and a blue border. A red rectangle highlights the 'jack' text in the first field, and another red rectangle highlights the password dots in the second field. Below the input fields is a blue button with the text 'Login' in white.

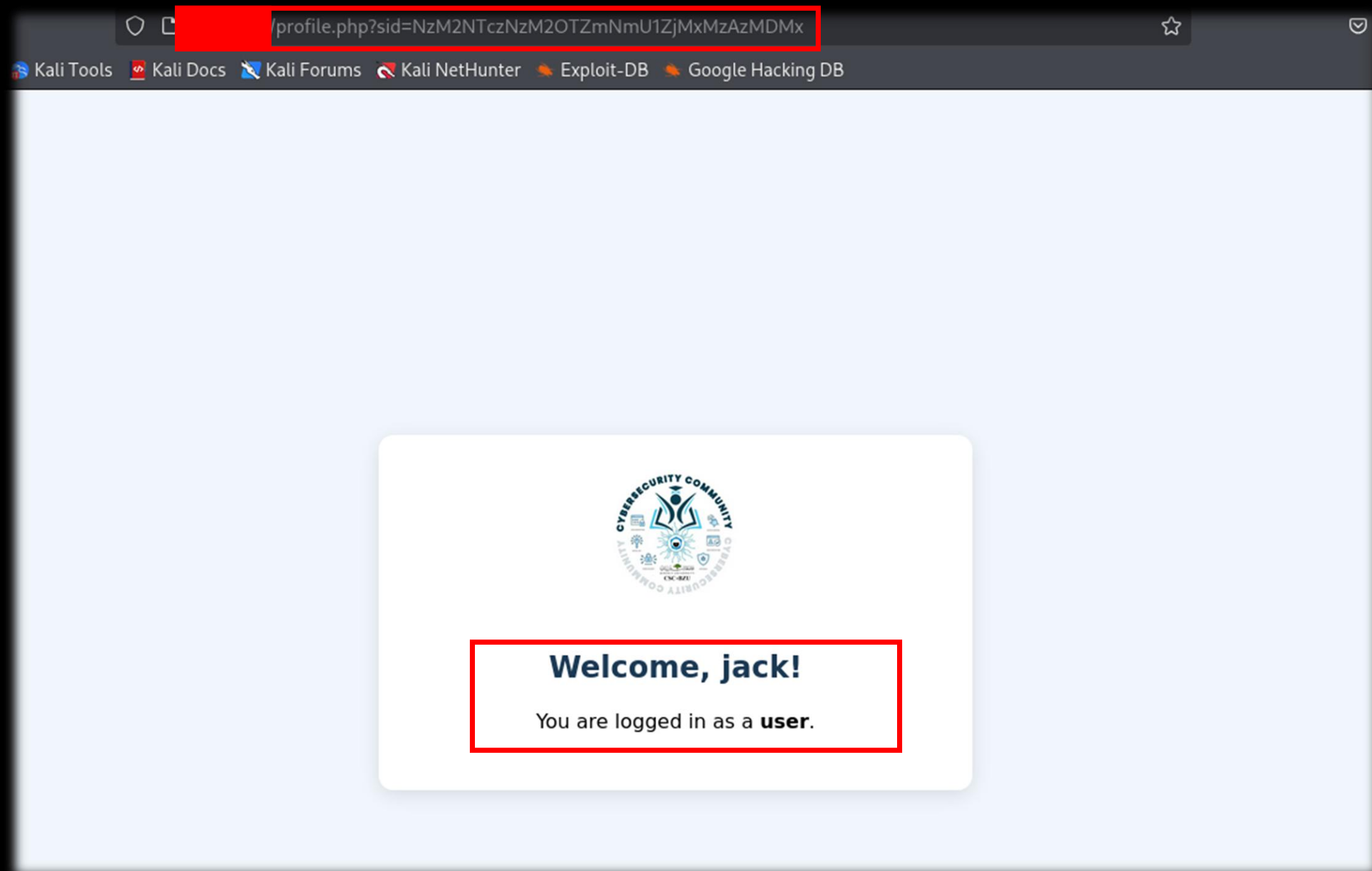
CSC-BZU Login

jack

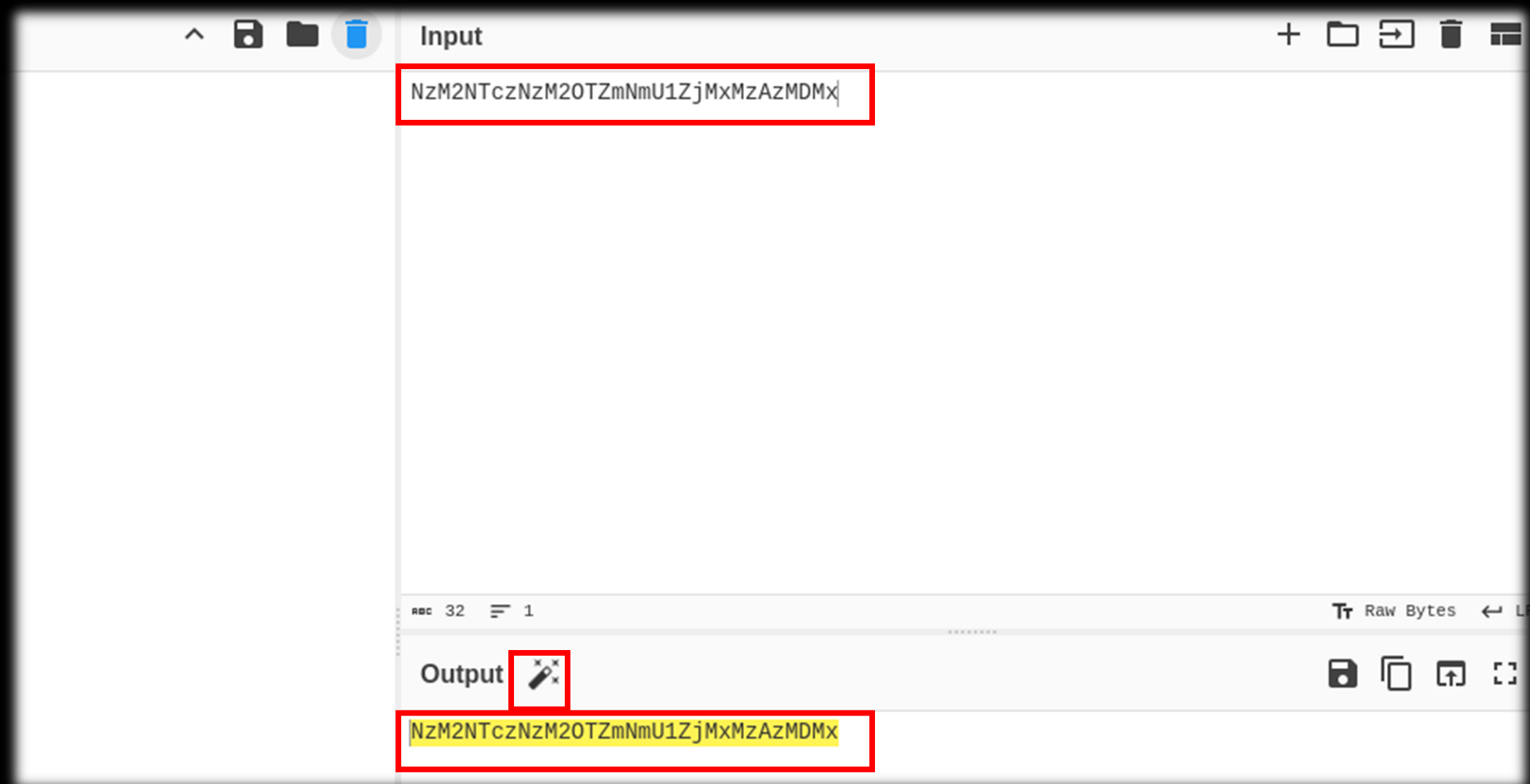
.....

Login

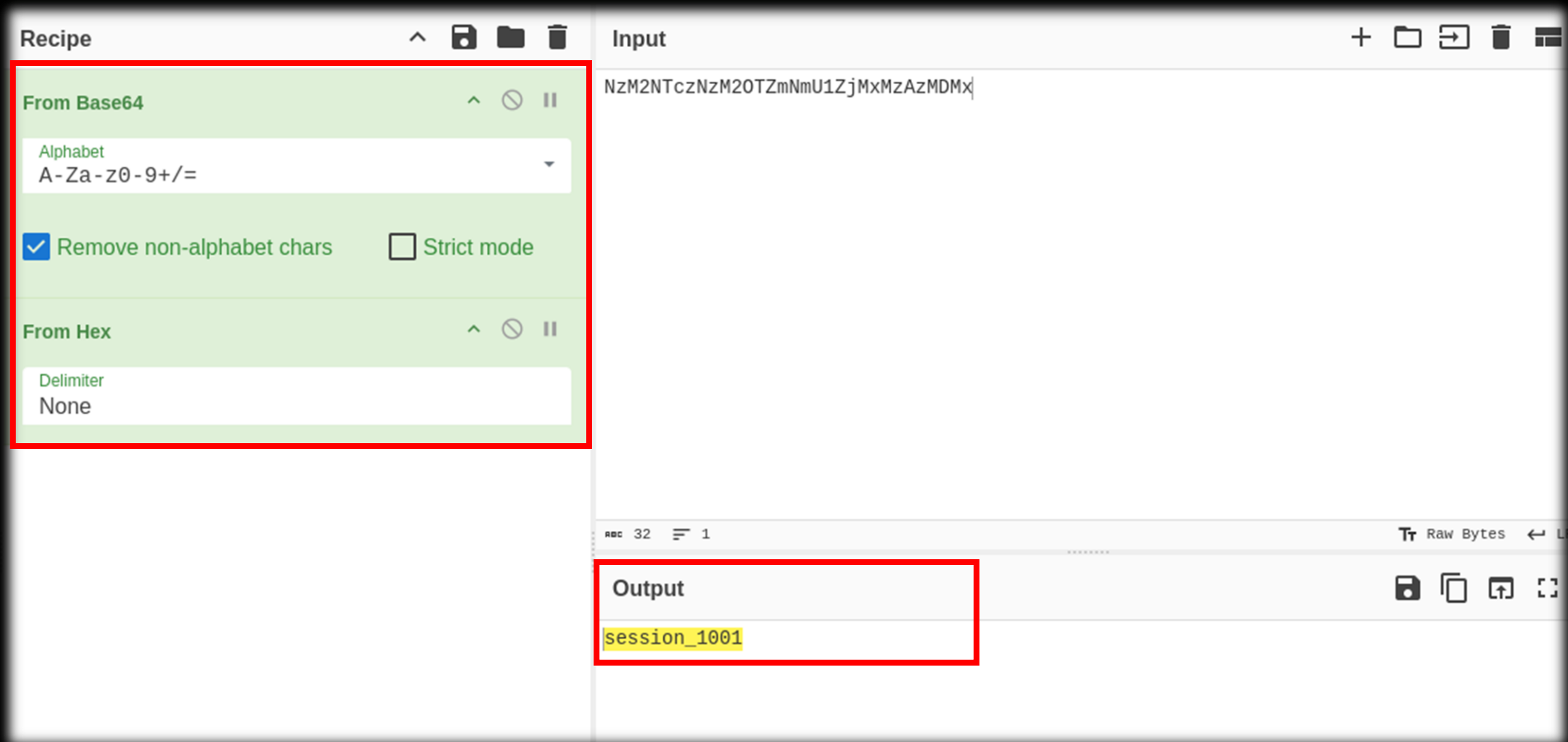
Now that we're logged in as the user **Jack**, we notice a **sid** parameter in the URL. Let's copy it and use **CyberChef** to decode it and see what it contains.



After pasting the value into **CyberChef** and clicking the Magic wand icon, we see that the data is first encoded in **Hex** and then in **Base64**.

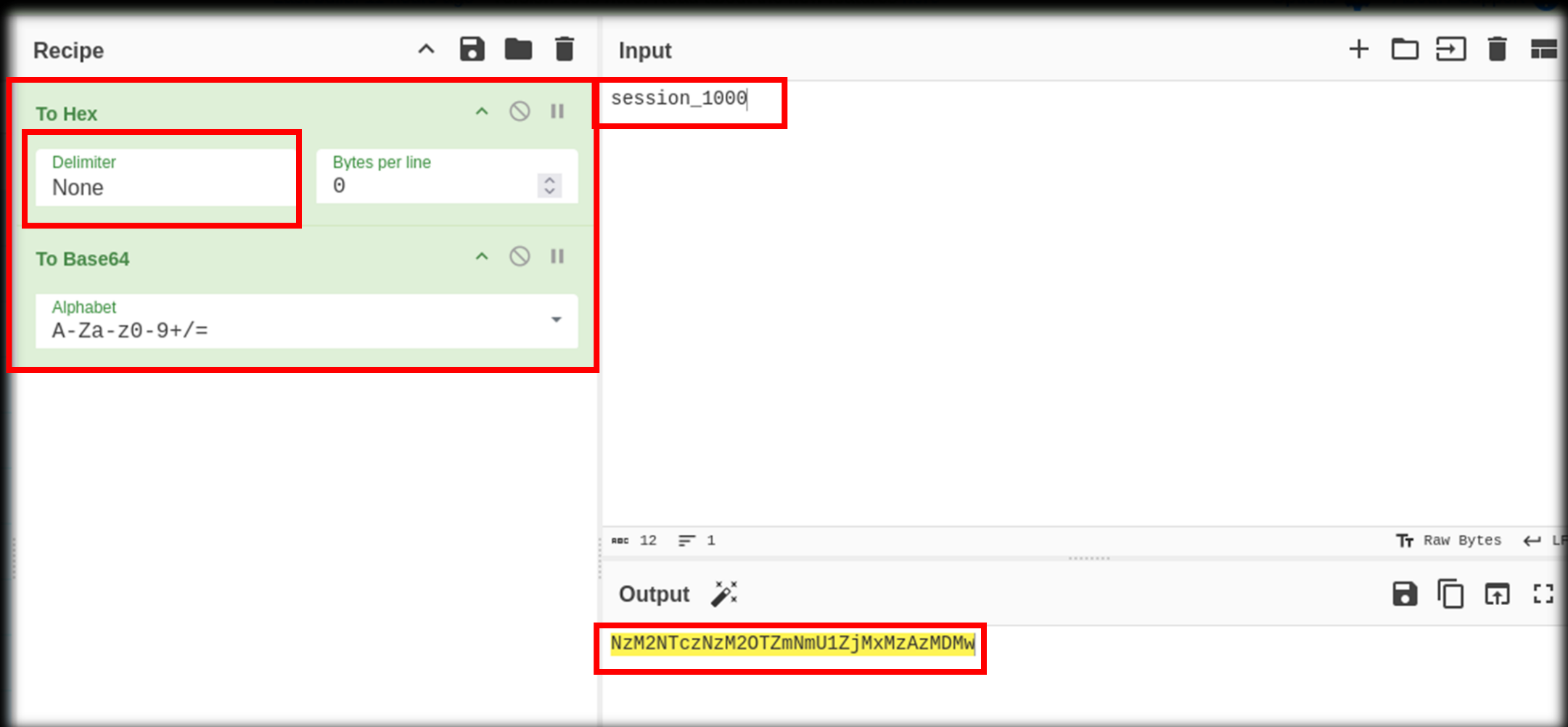


Once the text decoded, it reveals the string: **session_1001**:



The screenshot displays a web-based Base64 decoder interface. The interface is divided into two main sections: 'Recipe' on the left and 'Input' on the right. The 'Recipe' section is highlighted with a red border and contains two active recipes: 'From Base64' and 'From Hex'. The 'From Base64' recipe is selected, showing a dropdown menu for 'Alphabet' set to 'A-Za-z0-9+/' and a checked checkbox for 'Remove non-alphabet chars'. The 'From Hex' recipe is also visible, showing a dropdown for 'Delimiter' set to 'None'. The 'Input' section on the right contains a text area with the Base64 string 'NzM2NTczNzM2OTZmNmU1ZjMxMzAzMDMx'. Below the input area, the 'Output' section is highlighted with a red border and displays the decoded string 'session_1001' in yellow text.

The value session_1001 suggests that the session IDs might be sequential. Let's try changing **1001** to **1000** in the decoded string, re-encode it, and see if it gives us access to another user's page.



The screenshot shows a web application security tool interface with two main panels: "Recipe" and "Input".

Recipe Panel:

- To Hex:** A sub-panel with a "Delimiter" dropdown set to "None" and a "Bytes per line" dropdown set to "0".
- To Base64:** A sub-panel with an "Alphabet" dropdown set to "A-Za-z0-9+/".

Input Panel:

- The input field contains the text "session_1000".

Output Panel:

- The output field displays the Base64-encoded string "NzM2NTczNzM2OTZmNmU1ZjMxMzAzMDMw".

After updating the sid in the URL with the re-encoded **session_1000** value, we refresh the page and it logs us in as the admin. Just like that, we've gained access to the admin panel and captured the flag.

