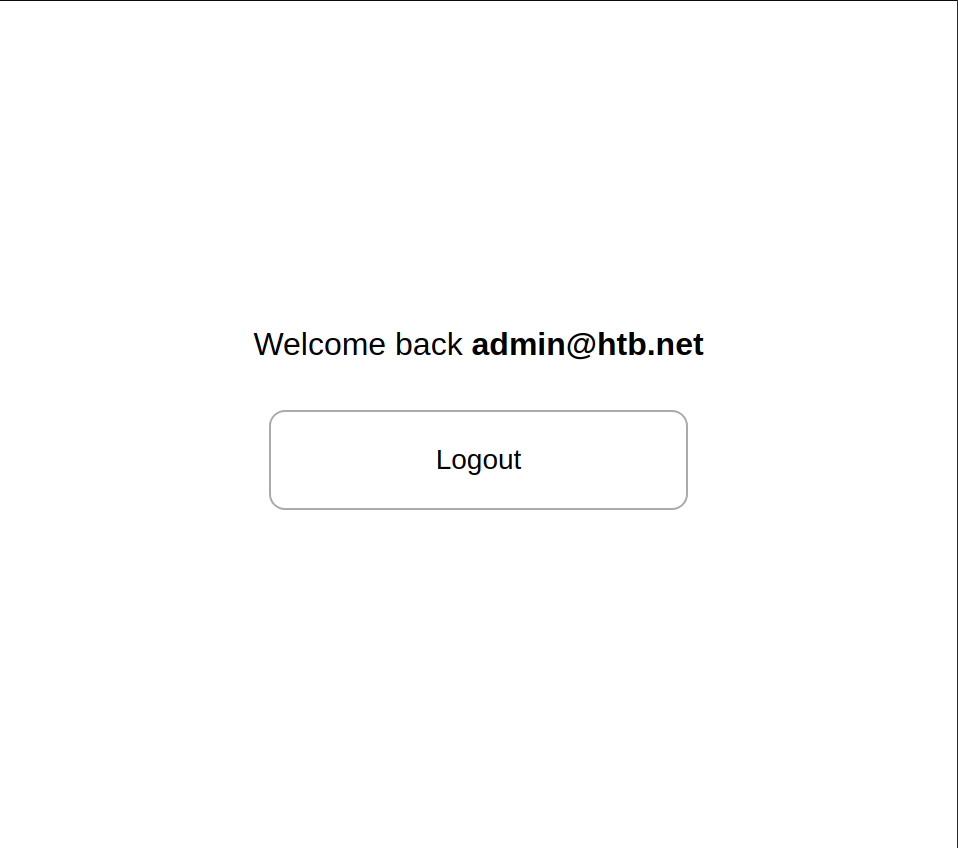
**No Signature Verification**

If no signature verification happens, the server cannot identify the user, nor can it determine whether or not the request was manipulated (changed) on the way. This can lead to you being able to update your roles, change users, etc.

**Capturing SAML Payload**

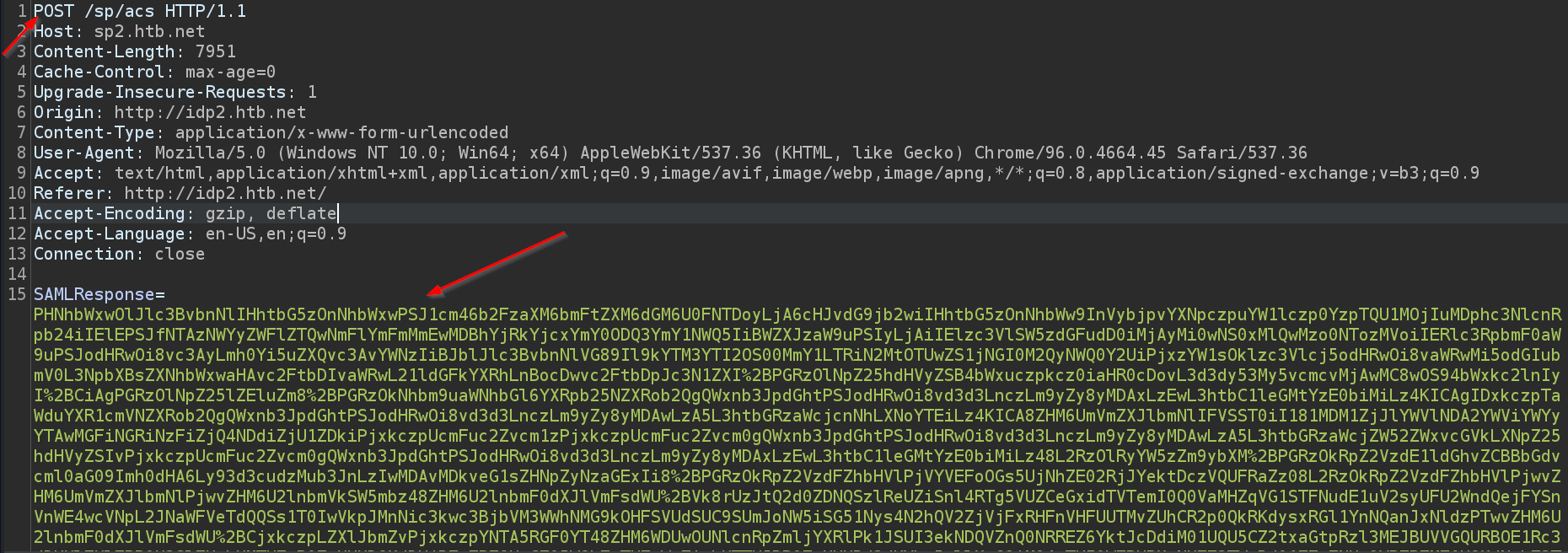
Open Burp so we can intercept the HTTP Traffic, navigate to http://sp2.htb.net and log in with the credentials provided at the start.



We need to log out and then back in to perform the attack. Log out and start the Interceptor.

**Note:** If you do not have the SAMLRaider extension for some reason, you can solve this lab manually. Decode the values and re-encode them. We'll be doing this manually so you learn more. The reason we had to use the SAMLRaider extension for the lab above was due to having to use the Private Key.

Firstly, URL decode the response from the /sp/acs endpoint. You'll see the POST Request in Burp.



Then, base64 decode the SAMLResponse value, and change the username SAMLAttribute to hackme, as follows:

Code: xml

<saml:Attribute Name="username" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"><saml:AttributeValue xsi:type="xs:string">hackme</saml:AttributeValue>

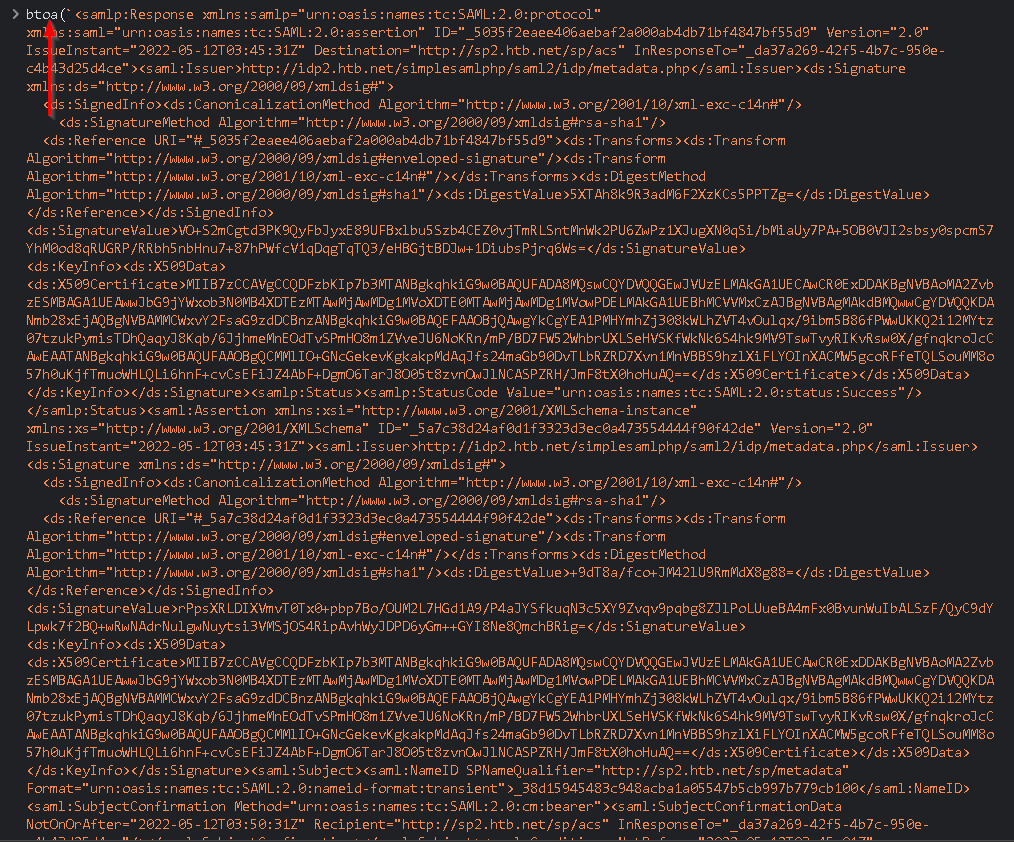
**Missing Signature Verification**

Now, if no signature checks are happening, we should be able to re-encode this and pass it back to the server in the hope we become the hackme user and get returned the flag. If there's a signature check, this will fail, and to bypass it, we'd require the Private key.

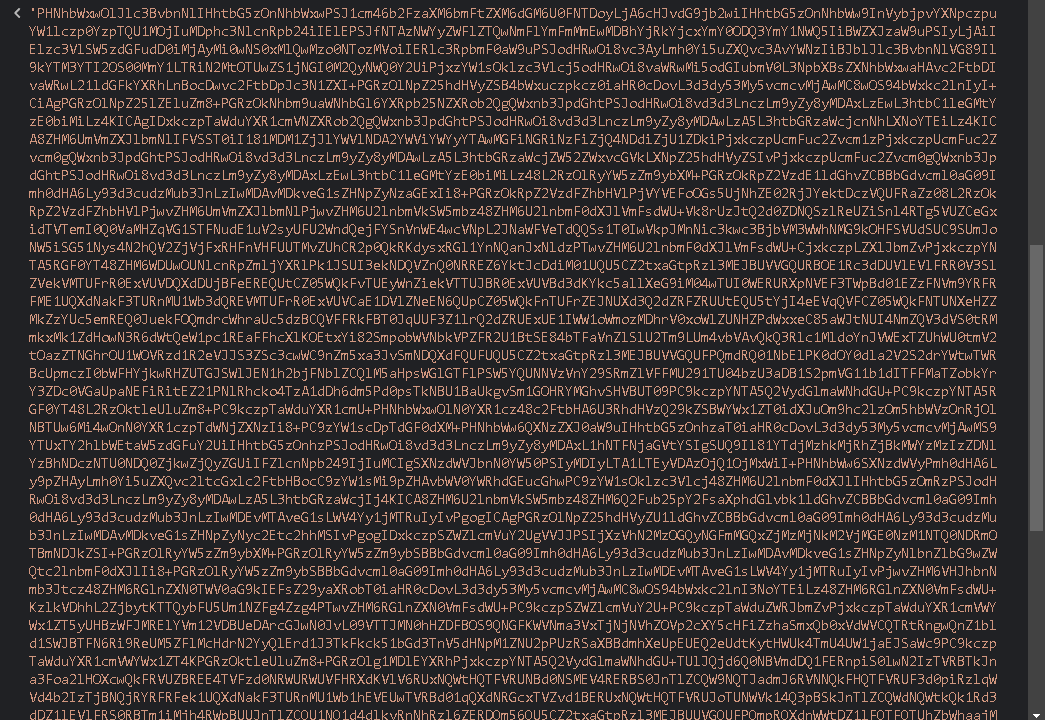
We'll encode using the browser console this time. Hit F12, or left click -> inspect element -> Console. We first need to Base64 encode these values. We'll be using btoa() to do so, btoa() allows you to parse a string which gets converted to Base64.

Syntax:btoa(string)

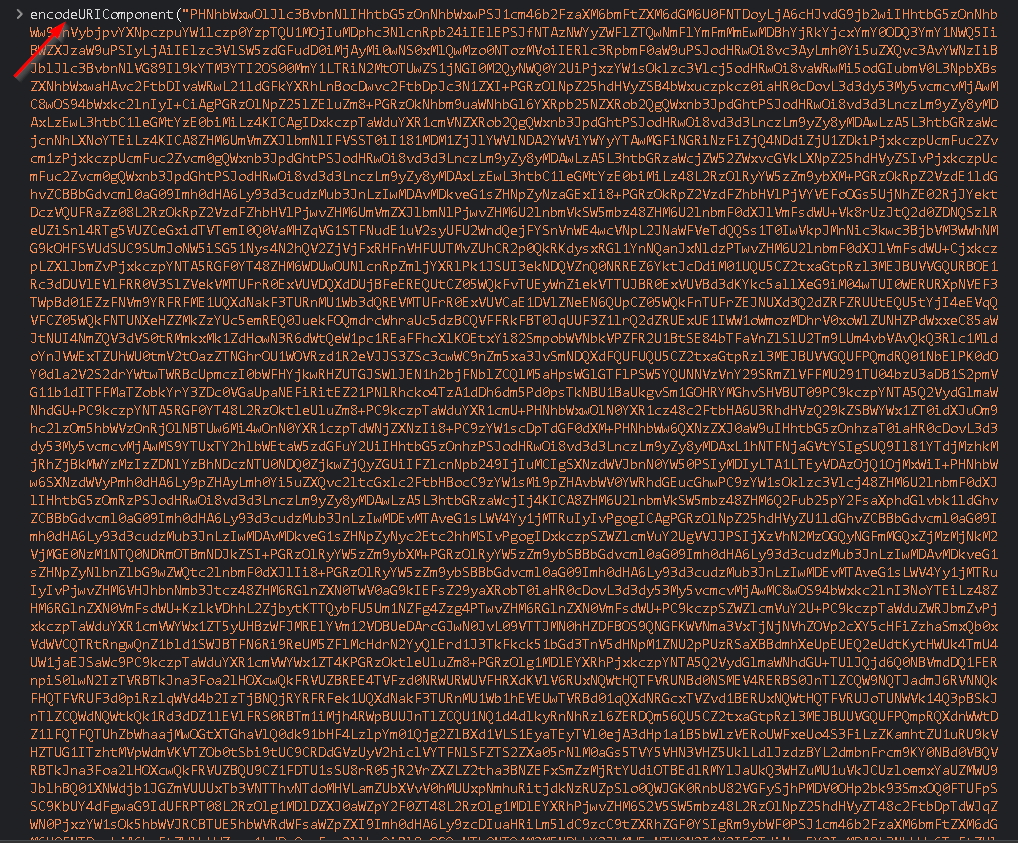
**Note:** Due to the string we're parsing already having double quotes, we'll get an error. We can use backticks (`).



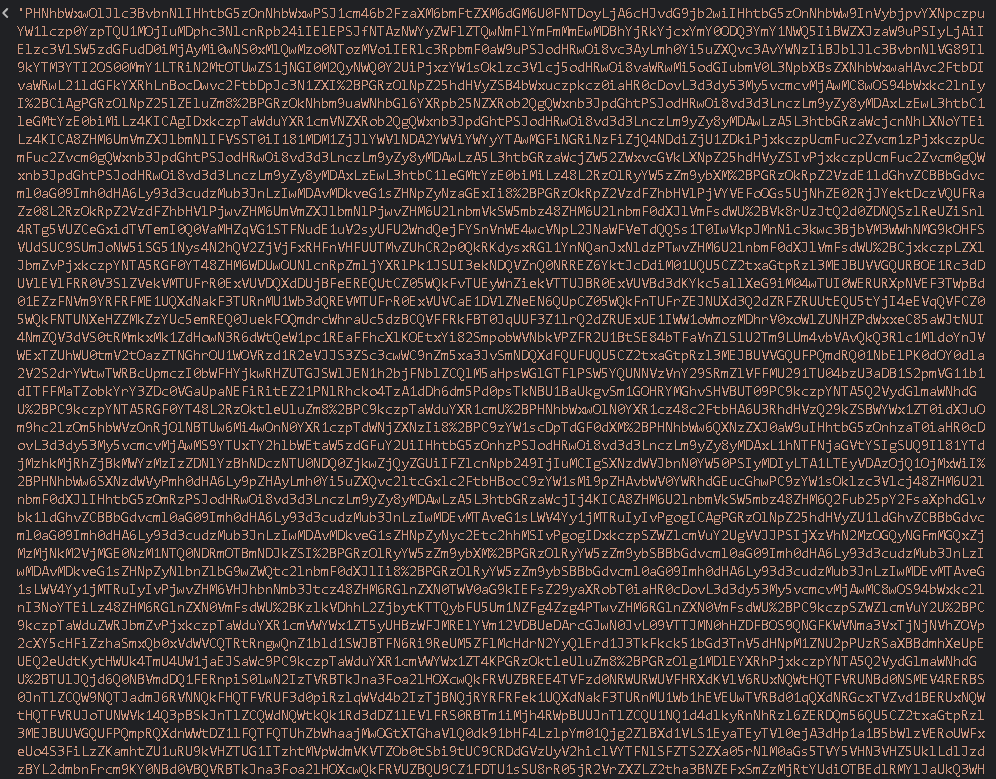
Output:



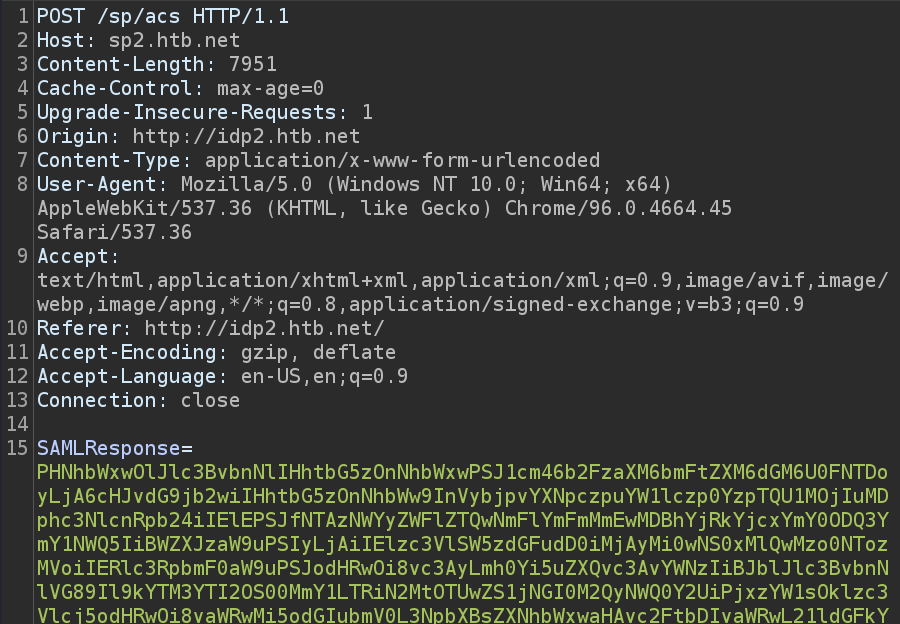
Copy the whole string, excluding the quotes, and now it's time to URL encode the values. We'll use the JS Method encodeURIComponent() for that. Copy the Base64 encoded value and parse it into the method as a string, e.g., encodeURIComponent("string").



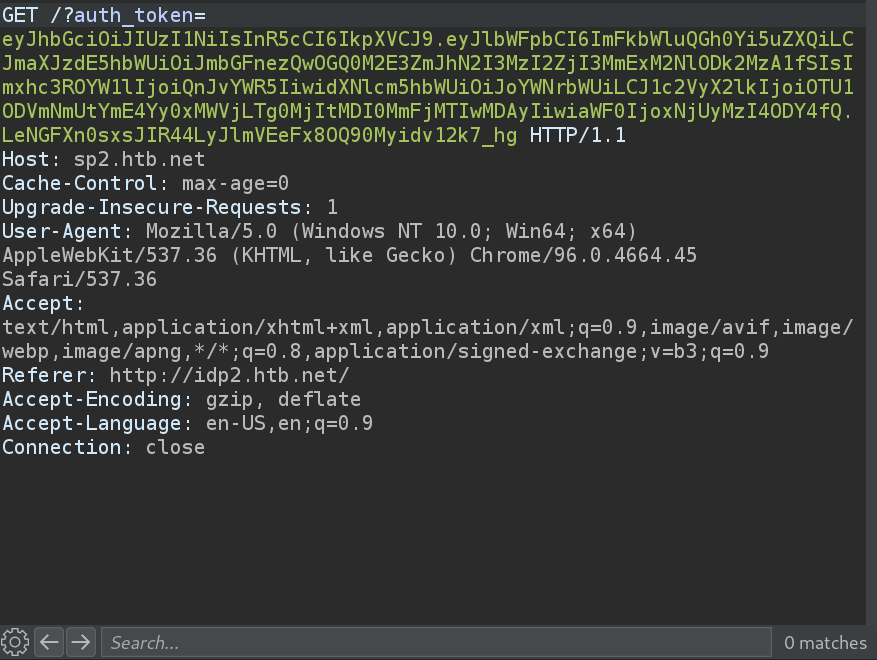
Output:



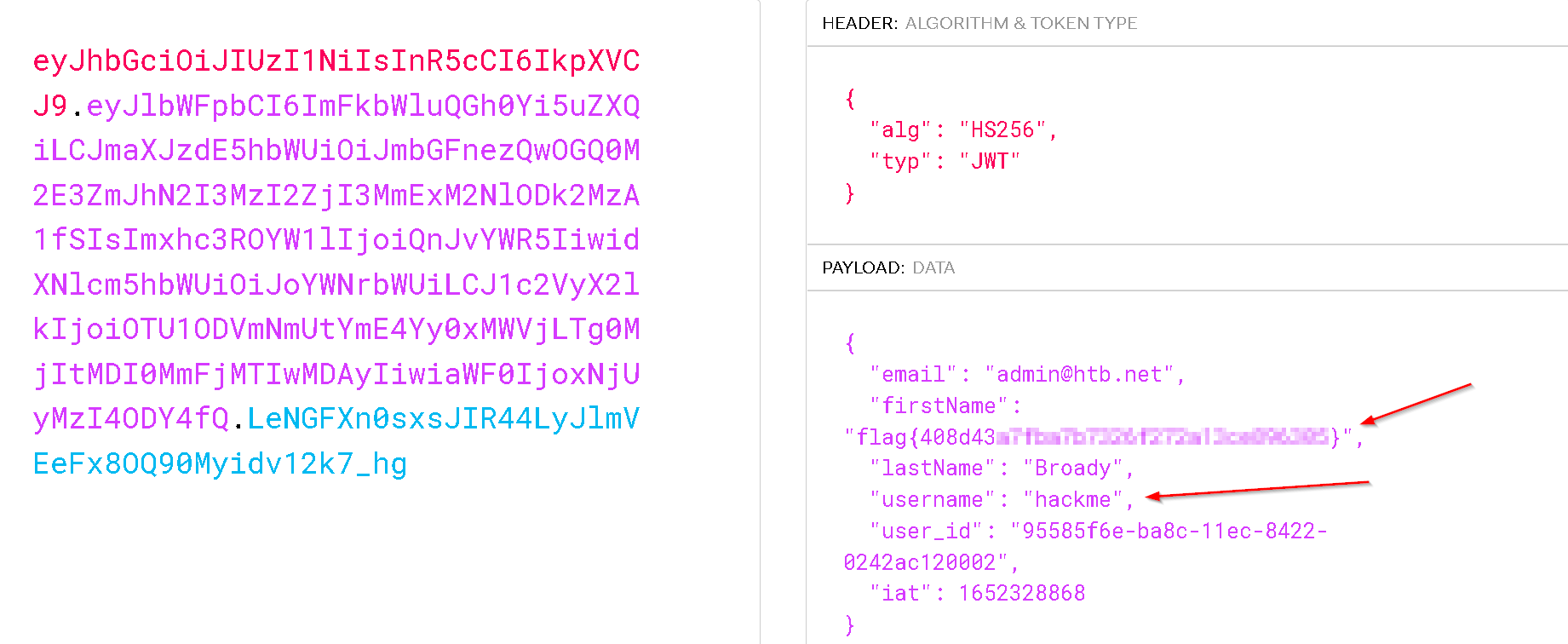
Copy and paste that URL Encoded payload and go back to Burp Suite. Remove the old encoded data and replace it with the manipulated data.



Forward the request.



Navigate to [jwt.io](https://jwt.io) and paste the string you got on the output.



As we can see, we have been able to modify our request without having a valid signature.