"The Best Festival Company's brand new OpenVPN server has been hacked. This is a crisis!

▶ Start Machine

The attacker has damaged various aspects of the company infrastructure -- including using the Christmas Control Centre to shut off the assembly line!

It's only 24 days until Christmas, and that line has to be operational or there won't be any presents! You have to hack your way back into Santa's account (blast that hacker changing the password!) and getting the assembly line up and running again, or Christmas will be ruined!"

*After giving you the assignment, McSkidy hands you the following dossier of important information for the task. Before reading it, you **press the big green "Deploy" button** to start the Control Centre, as well as the **"Start AttackBox" button at the top of the page** *

The Web:

The Internet is one of those things that everyone uses, but few people bother to learn about. As hackers, it is vital that we understand what exactly the web is, and how it works.

When you open up your web browser and navigate to a website, it seems so simple, but what is really happening behind the scenes?
First of all, your computer communicates with a known DNS (**D**omain **N**ame **S**ystem) server to find out where the website can be found on the internet. The DNS server will then return an **IP address** for the remote server. This can be used to go directly to the website. You can think of the internet as being quite like the planet itself -- we have lots of locations, all over the world. These places all have a street address -- this is akin to the domain name of a website (i.e. tryhackme.com, or google.com); but they also have co-ordinates which can be used to pinpoint their location with absolute accuracy. These co-ordinates are like the IP address of a website. If you know the street address of a location, you can enter it into Google Maps and be given the exact coordinates, which can then be put into a SatNav to take you there with pinpoint accuracy!
In the same way, your browser is given the address of a website (i.e. tryhackme.com). It sends this address off to a DNS server, which tells it the "co-ordinates" (the IP address) of the site. Your computer doesn't understand the original, human-readable domain name, but it *does* understand what an IP address is! The IP can then be used to find the server across the internet, allowing your computer to request the content of the website. Of course, in reality, this is a highly simplified analogy, so a more in-depth explanation of this process can be found here.

HTTP(S):

Once your computer knows where it can find the target website, it sends something called a HTTP (**H**yper**t**ext **T**ransfer **P**rotocol) request to the webserver.
https://i.imgur.com/r2IU7Ka.png
This is just a standard network request, but it is formatted in a way that both your web browser and the server can understand. In practice, this means adding certain "headers" to the request which identify it as a HTTP request, and tell the server a variety of other information about the request, as well as your own browser. Amongst many other headers, HTTP requests always have a *method* and a *target*. These specify *what* to retrieve from the server (the target), and *how* to retrieve it (the method). The method most commonly used to *retrieve* information is called the GET method. When sending data *to* the server, it's more common to use a method called POST.

For more information about HTTP requests, methods and headers, check out the Web Fundamentals room!

Once the content has been retrieved from the server, your browser reads the retrieved code and renders it as a web page. This usually means taking the layout of the page from a HTML (**H**yper **T**ext **M**arkup **L**anguage) document, styling it with a connected CSS (**C**ascading **S**tyle **S**heets) file, then adding any dynamic content with one or more connected Javascript files.

HTTP has one inherent disadvantage: namely, it is not secure. Anyone can see what you're requesting, and what's being sent back to you. For this reason, HTTPS (**H**yper**t**ext **T**ransfer **P**rotocol **S**ecure) was invented. This works in exactly the same way as standard HTTP but provides an encrypted connection (the functionality of which is beyond the level of this dossier)

Cookies:

HTTP is an inherently *stateless* protocol. This means that no data persists between connections; your computer could make two requests immediately after each other, and, without relying on separate software, the web server would have no way to know that it was you making both the requests. This begs the important question: if HTTP is stateless, then how do login systems work? The web server must have a way to identify that you have the right level of access, and it can hardly ask you to enter your password every time you request a new page!

The answer is cookies -- tiny little pieces of information that get stored on your computer and get sent to the server along with every request that you make. Authentication (or session) cookies are used to identify you (these will be *very* important in your mission today!). The server receives your request with the attached cookie, and checks the cookie to see what level of access you are allowed to have. It then returns a response appropriate to that level of access.

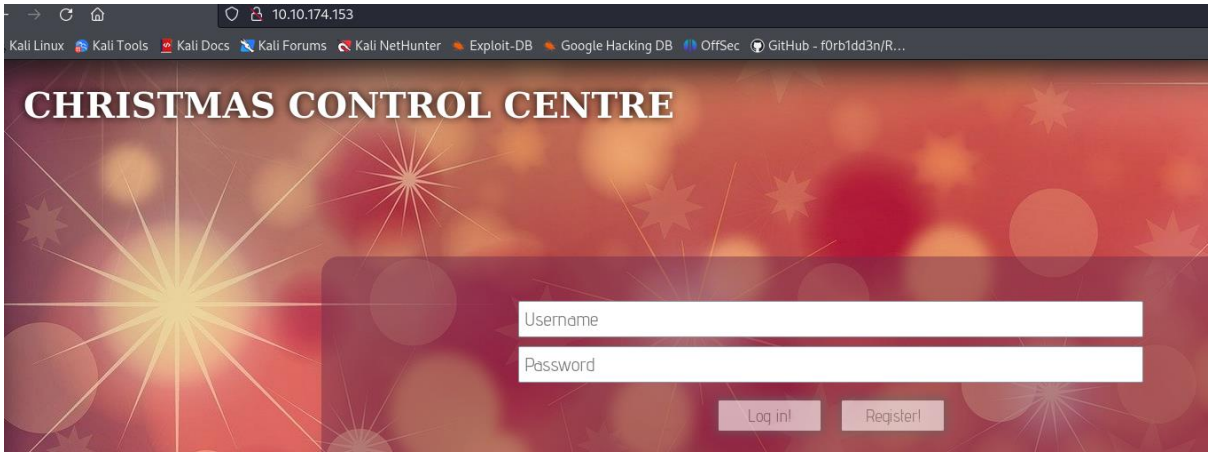
For example, a standard user should be able to see (but not interact with) our control panel; but Santa should be able to access everything! Cookies are also often used for other purposes such as advertising and storing user preferences (light/dark theme, for example); however, this will not be important in your task today. Any site can set cookies with a variety of properties -- the most important of these for today's task are the name and value of the cookies, both of which will always be set. It's worth noting that a site can only access cookies that are associated with its own domain (i.e. google.com can't access any cookies stored by tryhackme.com, and vice versa).

It's important to note that cookies are stored locally on *your* computer. This means that they are under your control -- i.e. you can add, edit, or delete them as you wish. There are a few ways to do this, however, it's most commonly done by using your Browser Developer Tools, which can be accessed in most browsers by pressing `F12`, or `Ctrl + Shift + I`. With the developer tools open, navigate to the `Storage` tab in FireFox, or the `Application` tab in Chrome/Edge and select the `Cookies` menu on the left hand side of the console.

In the above image you can see a test cookie for a website. The important attributes "Name" and "Value" are shown. The name of a cookie is used to identify it to the server. The value of the cookie is the data stored by the server. In this example the server would be looking for a cookie called "Cookie Name". It would then retrieve the value "CookieValue" from this cookie.

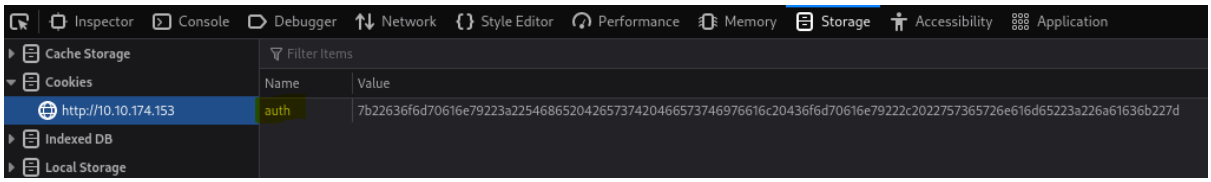
These values can be edited by double-clicking on them, which is great if you can edit a session or authorisation cookie, as this can lead to an escalation of privileges, assuming you have access to an Administrator's authorisation cookie.

1.Deploy your AttackBox (the blue "Start AttackBox" button) and the tasks machine (green button on this task) if you haven't already.. Once both have deployed, open FireFox on the AttackBox and copy/paste the machines IP into the browser search bar.

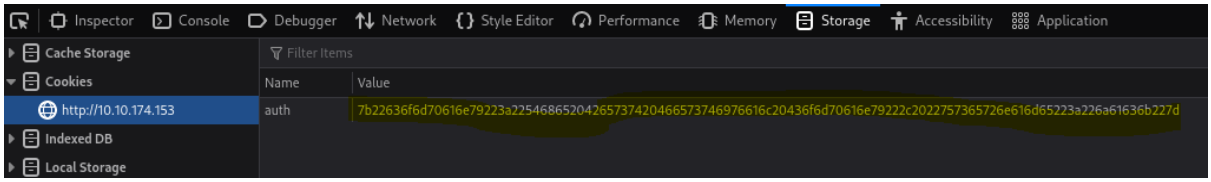


Register for an account, and then login.

What is the name of the cookie used for authentication?
auth



In what format is the value of this cookie encoded?
hexadecimal



Having decoded the cookie, what format is the data stored in?
JSON

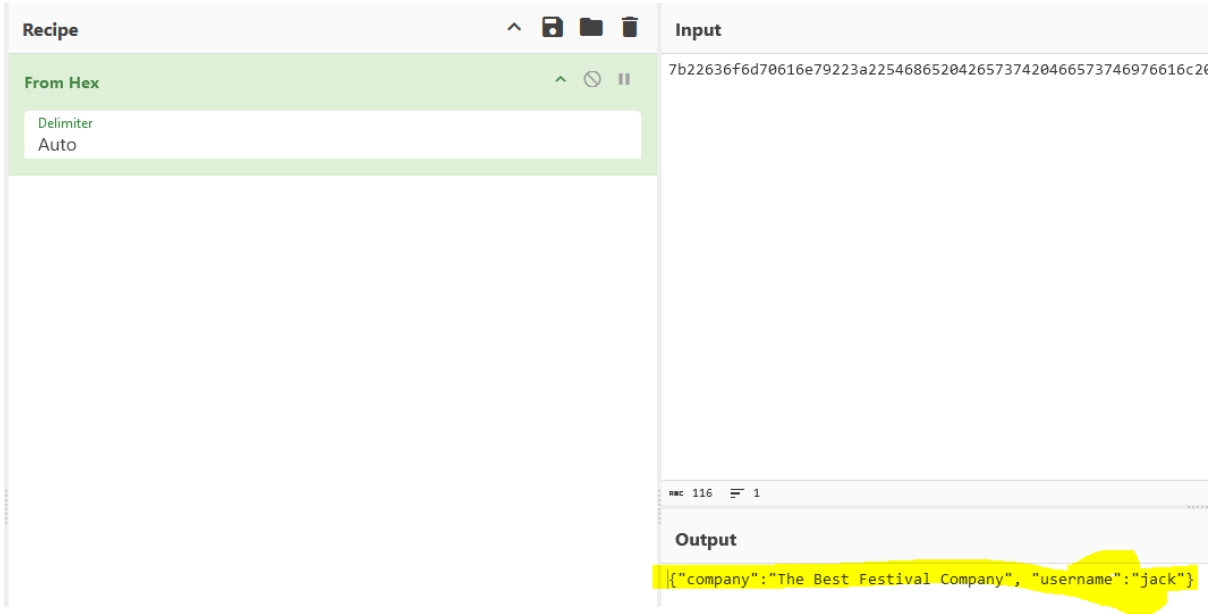
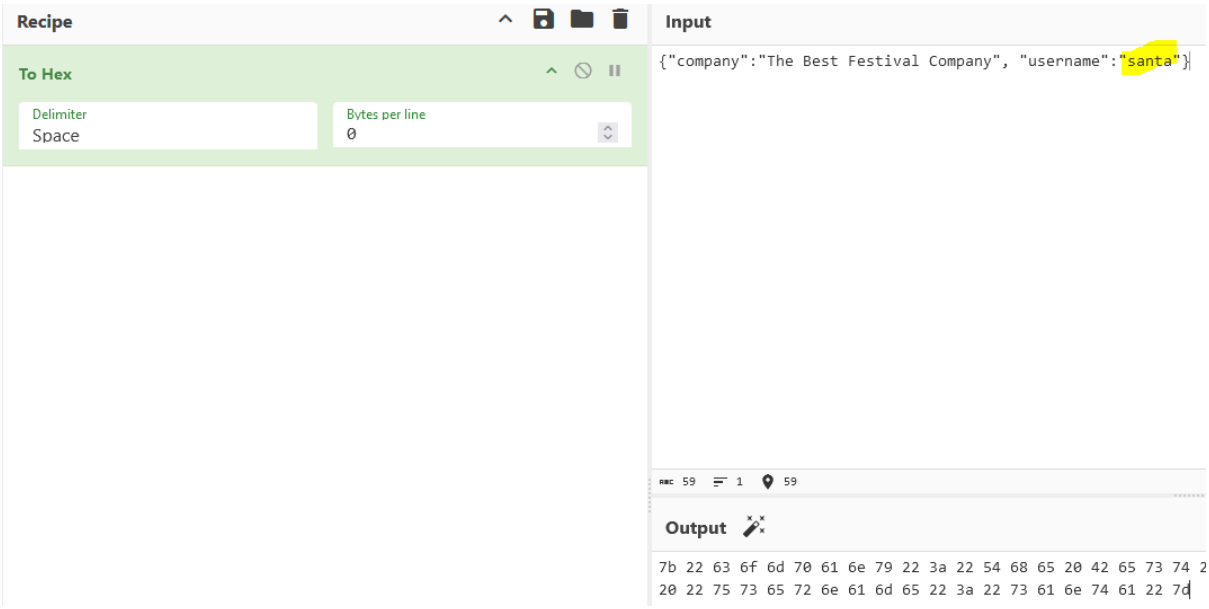


Figure out how to bypass the authentication.

What is the value of Santa's cookie?
7b 22 63 6f 6d 70 61 6e 79 22 3a 22 54 68 65 20 42 65 73 74 20 46 65 73 74 69 76 61 6c 20 43 6f 6d 70 61 6e 79 22 2c 20 22 75 73 65 72 6e 61 6d 65 22 3a 22 73 61 6e 74 61 22 7d



Now that you are the santa user, you can re-activate the assembly line!

What is the flag you're given when the line is fully active?