51.01 Root Christmas Control

Author

Discord: `v_anadium`

Challenge

> *Santa's running the Reindeer Command Center from the beach, but the control panel might not be as locked down as it looks. Holiday shortcuts, festive scripts, and trusted badges all play a role in keeping things running. Somewhere in the summer heat, a small oversight could turn a regular elf into the one calling the shots.*

Link

https://ch6.christmas.acucysctf.org/

Solution

Sign in or register with any username - say, `santa`. You will be logged in as "santa (user)", as shown in the top right.

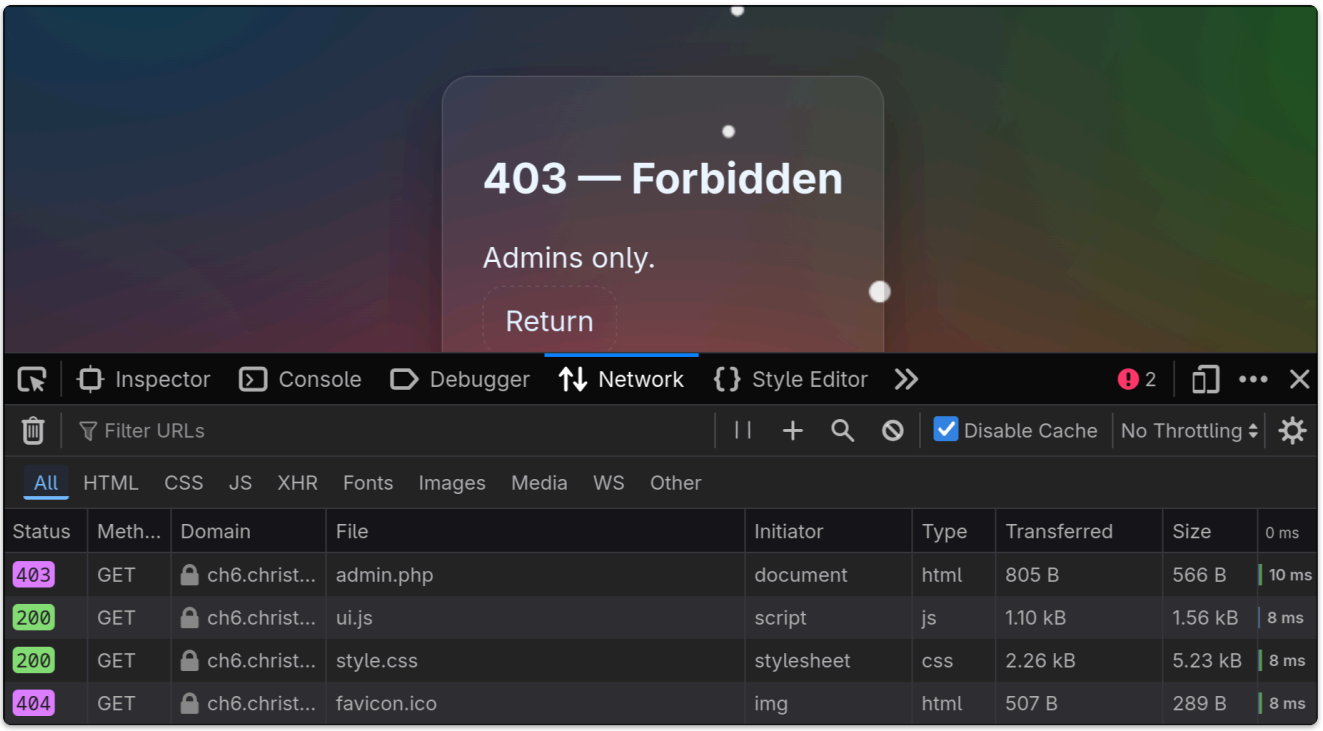
Attempting to login yields a `403 Forbidden` error at admin.php. The error text "Admins only" and webpage path suggest that an "admin" role will be required.

First, some preliminary poking around. By going to the console in section in Devtools, we can see the text `reindeer:ready` printed by `main.min.js`. This Javascript file is simple.

```
// tiny minified asset — hard-to-read map; value is base64 of the
signing key
var _m = {
  a: 'Q2hyaXN0bWFzMjAyNQ==',
  b: 'reindeer',
  c: 42
};
console.log('reindeer:ready');
```

Not much we can do with this yet, other than decode the "signing key" using base64, as suggested by the comment. It translates to `Christmas2025`. Let's try something else.

Inspecting the network traffic by using Devtools (`<F12>` → Network), we can see network requests made by the website. The one we're interested in is the one with a 403 response, made to `admin.php` when navigating to the console.



This is what the request looks like:

```
GET /admin.php HTTP/2
Host: ch6.christmas.acucysctf.org
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:146.0) Gecko/20100101
Firefox/146.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br, zstd
Referer: https://ch6.christmas.acucysctf.org/
Sec-GPC: 1
Connection: keep-alive
Cookie:
reindeer_auth=Tzo0OiJVc2VyIjoyOntzOjg6InVzZXJuYW1lIjtzOjU6InNhbnRhIjtzOj
Q6InJvbGUiO3M6NDoidXNlciI7fQ%3D%3D.bc28c03a2316e7de3144fe903e08f90e192f5
75947916520b7a0e4a79020a0e2
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Priority: u=0, i
Pragma: no-cache
Cache-Control: no-cache
TE: trailers
```

Clearly, the user authentication is done through the cookie `reindeer_auth`. We can inspect this cookie using Devtools again, in the Storage section (in Firefox), or using the options menu at the left of the address bar in Chrome.

Either way, there's only one cookie. There doesn't seem to be anything useful in the headers, but the payload appears to be a Base64 encoded array with a period separator.

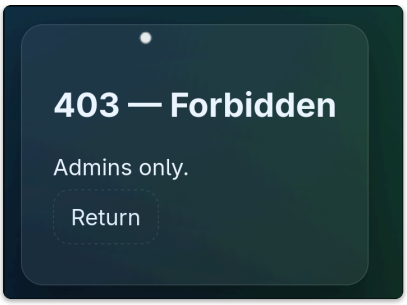
Let's try decoding the first part, before the period:

```
O:4:"User":2:{s:8:"username";s:5:"santa";s:4:"role";s:4:"user";}
```

We now have some usable output. A quick Google search will tell us that this is PHP serialized data. It seems to represent the user's login state. If this is true, we should be able to impersonate an admin by simply changing our role from "user" to "admin":

```
O:4:"User":2:{s:8:"username";s:5:"santa";s:4:"role";s:5:"admin";}
```

Encoding this back into base64 and replacing the first section of the array with our modified user data string, let's try to open the console again.



No such luck.

It seems the key lies in the second part of the array. It doesn't seem to be a base64 string like the first part, since decoding it results in non-ASCII gibberish.

Like a JWT, our string represents user data, and consists of multiple encoded strings separated by a period. It would be reasonable to assume that this system is based on the JWT specification. If we're on the right track, and the cookie is similar to a JWT, then it would make sense that the second portion is an HMAC signature, in line with the JWT spec.

If we can generate an HMAC signature, then we can get the server to accept our authorisation cookie. Normally, an HMAC signature is created by the server using a private signing key. What we need is the server's *signing key*, so that we can generate a signature that the server will trust. We did find one earlier, so let's try that:

```
echo -n 'O:4:"User":2:
{s:8:"username";s:5:"santa";s:4:"role";s:5:"admin";}' | openssl dgst -
sha-256 -hmac "Christmas2025"
```

Finally, let's assemble our user data string and the HMAC signature in the same format that we found it in, then replace our cookie with that.

"Hello, santa (admin)" now appears in the top right. We are now able to log into the admin console and view the flag.

■