Ethical Hacking – Lab 3

Developing a docker container affected by vulnerabilities

1. Vulnerabilities

The docker container which I have developed is affected by two vulnerabilities:

- CVE-2018-15473 (SSH Username Enumeration) – OpenSSH up to version 7.7 is prone to a username enumeration vulnerability due to not waiting for the message to fully parse before communicating invalid user error. This means that by sending a malformed public key we can check if the username is valid by the reaction of the program – if it is not, we get an error message, otherwise we get no information.
- SNMP "Public Community Strings" – in SNMP, community string serves as a kind of a password to connect to it. In SNMPv1 and SNMPv2 there are very well-known default values for the read-only and read-write access – "public" and "private". Since many system and network administrators do not change these configurations, it has become very easy for hackers to get data from SNMP and change it.

2. Resources Used

Content of the Dockerfile used to build the container:

```
FROM ubuntu:18.04

ARG DEBIAN_FRONTEND=noninteractive

RUN apt-get update && \
    apt-get install -y \
        openssh-server \
        snmpd && \
    apt-get clean

RUN echo "agentAddress udp:161" > /etc/snmp/snmpd.conf && \
    echo "rocommunity public" >> /etc/snmp/snmpd.conf && \
    echo "rwcommunity public" >> /etc/snmp/snmpd.conf

RUN sed -i 's/^#\?PasswordAuthentication .*/PasswordAuthentication yes/' /etc/ssh/sshd_config
RUN sed -i 's/^#\?PermitRootLogin .*/PermitRootLogin yes/' /etc/ssh/sshd_config

EXPOSE 22 161/udp

ENTRYPOINT service ssh restart && service snmpd start && bash
```

First, I pulled an Ubuntu 18.04 image, because its default OpenSSH version is 7.6, which is affected by the enumeration vulnerability. Then using apt-get I installed all the necessary and available tools for the exploitation. Then I configured SNMP to accept connections from everyone on port 161 using UDP and set up both read-only and read-write community strings to 'public'. After that I added 'PasswordAuthentication yes' line to my sshd_config file, because in order for the exploit to work, it needs at least one authentication method. Finally, I chose which ports I will expose (22 for SSH and 161/udp for SNMP) and put in the commands that will execute at the entrypoint of my program (when I run my container) – starting both SSH and SNMPD services and opening bash – to keep my container running.

3. Building

I built my Docker Image using the following command in the folder where my Dockerfile was:

```
sudo docker build -t vulnerable-container .
```

The '-t' flag is optional for specifying the name of the container.

Then I ran it using the command:

```
sudo docker run -dit vulnerable-container -p 22:22 161:161/udp
```

The '-dit' flags are for keeping the container alive in the background.

4. Exploitation

First, I checked if everything was set up correctly using nmap:

```
changeme@ubuntu:~/Desktop$ sudo nmap -sU -sS -sV 172.17.0.2 -p T:22,U:161
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-29 20:27 UTC
Nmap scan report for 172.17.0.2
Host is up (0.00011s latency).

PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
161/udp open  snmp    SNMPv1 server; net-snmp SNMPv3 server (public)
MAC Address: 02:42:AC:11:00:02 (Unknown)
Service Info: Host: 3f2ef96a2ae8; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.45 seconds
```

Then I focused on testing the "Public Community String" vulnerability on the SNMP service. In order to do that I used snmpwalk tool. Attached below are the first few lines of the result of that tool and the command that I used:

```
changeme@ubuntu:~/Desktop$ snmpwalk -v 1 -c public 172.17.0.2:161
iso.3.6.1.2.1.1.1.0 = STRING: "Linux 3f2ef96a2ae8 6.8.0-47-generic #47-Ubuntu SMP PREEMPT_DYNAMIC Fri Sep 27 21:40:26 UT
C 2024 x86_64"
iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.8072.3.2.10
iso.3.6.1.2.1.1.3.0 = Timeticks: (19925) 0:03:19.25
iso.3.6.1.2.1.1.4.0 = STRING: "root"
iso.3.6.1.2.1.1.5.0 = STRING: "3f2ef96a2ae8"
iso.3.6.1.2.1.1.6.0 = STRING: "Unknown"
```

It is clearly visible that this is the same host as listed with nmap tool and we can list all the information using a public community string.

After that I moved on to SSH Username Enumeration. To check this vulnerability, I used a script provided in '*Security-Learning-Hub*':

```
(.venv) changeme@ubuntu:~/Desktop$ python openssh_username_enumeration.py --port 22 172.17.0.2 root
/home/changeme/Desktop/.venv/lib/python3.12/site-packages/paramiko/pkey.py:82: CryptographyDeprecationWarning: TripleDES
 has been moved to cryptography.hazmat.decrepit.ciphers.algorithms.TripleDES and will be removed from cryptography.hazma
t.primitives.ciphers.algorithms in 48.0.0.
  "cipher": algorithms.TripleDES,
/home/changeme/Desktop/.venv/lib/python3.12/site-packages/paramiko/transport.py:253: CryptographyDeprecationWarning: Tri
pleDES has been moved to cryptography.hazmat.decrepit.ciphers.algorithms.TripleDES and will be removed from cryptography
.hazmat.primitives.ciphers.algorithms in 48.0.0.
  "class": algorithms.TripleDES,
[+] Valid username
```

We can see that using the script I was able to check that root is in fact a valid username.

5. Utilizing buildx functionality

Ater successfully creating a vulnerable container I moved on to the last part of the assignment – making my image work in both amd64 and arm64 architectures. To do that I used command line below:

```
PS C:                                                  docker buildx build --platform linux/amd64,linux/arm64 -t buildx-vulnerable-contai
ner --load .
[+] Building 192.9s (15/15) FINISHED                                                                                  docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                                  0.0s
 => => transferring dockerfile: 851B                                                                                                  0.0s
 => [linux/arm64 internal] load metadata for docker.io/library/ubuntu:18.04                                                           0.5s
 => [linux/amd64 internal] load metadata for docker.io/library/ubuntu:18.04                                                           0.5s
 => [internal] load .dockerignore                                                                                                     0.0s
 => => transferring context: 2B                                                                                                       0.0s
 => CACHED [linux/arm64 1/5] FROM docker.io/library/ubuntu:18.04@sha256:152dc042452c496007f07ca9127571cb9c29697f42acbfad72324b2bb2e43c98   0.0s
 => => resolve docker.io/library/ubuntu:18.04@sha256:152dc042452c496007f07ca9127571cb9c29697f42acbfad72324b2bb2e43c98                 0.0s
 => CACHED [linux/amd64 1/5] FROM docker.io/library/ubuntu:18.04@sha256:152dc042452c496007f07ca9127571cb9c29697f42acbfad72324b2bb2e43c98   0.0s
 => => resolve docker.io/library/ubuntu:18.04@sha256:152dc042452c496007f07ca9127571cb9c29697f42acbfad72324b2bb2e43c98                 0.0s
 => [linux/arm64 2/5] RUN apt-get update &&     apt-get install -y      openssh-server          snmpd &&     apt-get clean          180.4s
 => [linux/amd64 2/5] RUN apt-get update &&     apt-get install -y      openssh-server          snmpd &&     apt-get clean           35.4s
 => [linux/amd64 3/5] RUN echo "agentAddress udp:161" > /etc/snmp/snmpd.conf &&     echo "rocommunity public" >> /etc/snmp/snmpd.conf &&     echo "rw  0.4s
 => [linux/amd64 4/5] RUN sed -i 's/^#\?PasswordAuthentication .*/PasswordAuthentication yes/' /etc/ssh/sshd_config                  0.6s
 => [linux/amd64 5/5] RUN sed -i 's/^#\?PermitRootLogin .*/PermitRootLogin yes/' /etc/ssh/sshd_config                                 0.5s
 => [linux/arm64 3/5] RUN echo "agentAddress udp:161" > /etc/snmp/snmpd.conf &&     echo "rocommunity public" >> /etc/snmp/snmpd.conf &&     echo "rw  0.4s
 => [linux/arm64 4/5] RUN sed -i 's/^#\?PasswordAuthentication .*/PasswordAuthentication yes/' /etc/ssh/sshd_config                  0.6s
 => [linux/arm64 5/5] RUN sed -i 's/^#\?PermitRootLogin .*/PermitRootLogin yes/' /etc/ssh/sshd_config                                 0.5s
 => exporting to image                                                                                                               10.3s
 => => exporting layers                                                                                                               8.2s
 => => exporting manifest sha256:57e6cad18774ce5c4f62a7b28bf3a363caf1f678640698eb01686ff8cb4d0da6                                     0.0s
 => => exporting config sha256:082b405ce20e02b13df20f121f61c4e514a3238e6979f1d716c9f70ec6065be3                                       0.0s
 => => exporting attestation manifest sha256:86cded80a97461fe9e2b115668cf0933ac23ccb909228fe03b4fc1879767fd51                         0.0s
 => => exporting manifest sha256:d46ab0a73b535ccc4b588b684b3bfc636baf341eec77d77c8d100ea4f5e8eb97                                     0.0s
 => => exporting config sha256:7514df7289a3d1d5ff180e2e98af6448bdafb931eace287eb81132acb2bafd6f                                       0.0s
 => => exporting attestation manifest sha256:86ed7ea2a9f978376c71826179cf8dbae0155915bf8d01afa4ec976b56bebe1f                         0.0s
 => => exporting manifest list sha256:f6c8fd59abb36ec318cdbddc248ed407a2e30dde2cb33b29163296cd41f566bf                                0.0s
 => => naming to docker.io/library/buildx-vulnerable-container:latest                                                                 0.0s
 => => unpacking to docker.io/library/buildx-vulnerable-container:latest                                                              1.9s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/evdjppp87761v3468zygvk6xz
```

Next, I ran my container like normal and checked the list of active docker containers:

```
PS C:                                                  docker run -dit buildx-vulnerable-container -p 22:22, 161:161/udp
a0a6fc7e18f95cdbcb5de112ce22e75adecdeb1c3e4e218d1dd1fff7f56cd53c
PS C:                                                  docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED         STATUS         PORTS                NAMES
a0a6fc7e18f9   buildx-vulnerable-container    "/bin/sh -c 'service…"   3 seconds ago   Up 2 seconds   22/tcp, 161/udp      unruffled_ptolemy
```

We can see that my multi-platform container has been built and run successfully.

6. Additional Remarks

At first, I tried to run my Dockerfile on Windows 11 host, but while I could connect to the SSH service, I was not able to perform a snmpwalk. I think the problem may be related to Windows Defender or firewall, because nmap (or Zenmap) was showing me that port 161 was open but filtered. I tried disabling the firewall and all the antivirus programs I have on my machine, but to no avail (apparently Windows firewall can still block some connections even if it is turned off). On Windows I was only able to perform snmpwalk when I ran another Docker container and did it from there.

However, last part of my assignment is done on Windows 11, because using '—load' parameter on Ubuntu returns an error if you try to pass multiple platforms in '—platform' flag.