Assignment #6: nmap and iptables

Ravera Alessio (4193948)

Dopo aver configurato la VM metasploitable su virtualbox (network in modalita Bridged Adapter), ho effettuato lo scan delle porte tramite nmap con l'obbiettivo di vedere i servizi di rete aperti, ottenendo il seguente risultato:

```
nmap -p0-65535 192.168.1.187

Starting Nmap 7.60 ( https://nmap.org ) at 2018-12-15 19:05 CET
Nmap scan report for 192.168.1.187
Host is up (0.00038s latency).
Not shown: 65506 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
3632/tcp  open  distccd
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
6697/tcp  open  ircs-u
```

8009/tcp open ajp13
8180/tcp open unknown
8787/tcp open msgsrvr
35099/tcp open unknown
35821/tcp open unknown
50281/tcp open unknown
55822/tcp open unknown

Nmap done: 1 IP address (1 host up) scanned in 0.71 seconds

L’esercitazione consiste nel configurare tramite l’applicazione iptables, il firewall a livello kernel in linux(implementato come diversi moduli Netfilter) allo scopo di filtrare i pacchetti basandosi sulle porte, sui protocolli o altri criteri.

Per prima cosa, dato il numero limitato di servizi da consentire ho deciso di adottare un approccio default drop per i pacchetti in entrata, approccio che come discusso a lezione presenta maggiore sicurezza in quanto se il filtraggio è configurato male sarà piu probabile che si verificheranno malfunzionamenti pittosto che l’introduzione di vulnerbilità, anche se non è da escludere.
In seguito ho inserito le regole di filtraggio con l’obbiettivo di consentire solo i servizi richiesti, le regole vengono inserite in sequenza e sono chiamate “chains of rules”.
Come visto a lezione ho inserito la regola che permette di lasciar passare tutti i pacchetti tcp con il flag ACK, questo perchè tali pacchetti non sono da considerarsi pericolosi in quanto o appartengono a connessioni già stabilite o verranno scartati in seguito.
Inoltre è opportuno anche consentire le risposte alle richieste DNS che utilizza il protocollo UDP, quindi è stato necessario un approccio stateful in modo tale da permettere le risposte provenienti dalla porta 53 e da un indirizzo da cui si è fatta una richiesta DNS; questo lo si puo fare utilizzando l’opzione -m state --state ESTABLISHED.
Stesso discorso per i pacchetti ICMP utilizzati per effettuare il ping verso altre macchine.
Per verificare il corretto funzionamento del servizio DNS ho utilizzato il comando curl.

L’output ora fornito da nmap è il seguente:

```
nmap 192.168.1.187

Starting Nmap 7.60 ( https://nmap.org ) at 2018-12-15 20:02 CET
Nmap scan report for 192.168.1.187
Host is up (0.00037s latency).
Not shown: 997 filtered ports
PORT    STATE  SERVICE
22/tcp  open   ssh
80/tcp  open   http
443/tcp closed https

Nmap done: 1 IP address (1 host up) scanned in 4.95 seconds
```

Da notare che il servizio https non è tuttavia fornito dalla VM.

Vista la debolezza delle password utilizzate dalla macchina virtuale ho voluto approfondire la configurazione delle iptables, in particolare ho provato ad effettuare un attacco di brute force tramite il tool “medusa” sul servizio ssh creandomi un mio dizionario “giocattolo” dove conteneva anche la password corretta; dopo pochi minuti ho potuto constatare il successo dell’attacco.

```
medusa -u user -P  ' dictionary' -h 192.168.1.187 -M ssh -t 32
                                                    .
                                                    .
                                                    .
ACCOUNT FOUND: [ssh] Host: 192.168.1.187 User: user Password: user [SUCCESS]
```

Facendo una ricerca ho trovato che è possibile limitare il numero di tentativi di connessione via ssh consegutivi provenienti dallo stesso indirizzo:

```
iptables -A INPUT -p tcp -m tcp --dport 22 -m state --state NEW -m recent --set --name SSH –rsource

iptables -A INPUT -p tcp -m tcp --dport 22 -m recent --rcheck --seconds 30 --hitcount 4 --rttl --name SSH --
rsource -j REJECT --reject-with tcp-reset
```

Il primo comando dice al sistema di contrassegnare i pacchetti tcp in

entrata che tentano di stabilire una connessione ssh come SSH e di prestare attenzione alla fonte del pacchetto.
Il secodo dice se arriva un pacchetto che tenta di stabilire una connessione SSH, ed è il quarto pacchetto che proviene dalla stessa fonte negli ultimi trenta secondi, lo rifiuta.
Configurando in questo modo ripetendo l’attacco dopo che vengono provate 4 password si ottiene la seguente risposta dal tool:

NOTICE: ssh.mod : failed to connect, port 22 was not open on 192.168.43.37