

Sieci komputerowe

Warsztaty 7

Mateusz Markiewicz

7 czerwca 2020

1 Pierwsze zadanie do zaprezentowania

Pierwszym etapem było stworzenie dwóch maszyn wirtualnych z dwiema kartami (NAT oraz sieć wewnętrzna). Następnie aktywowałem oraz skonfigurowałem interfejsy.

- `V1#> dhclient -v enp0s3`
- `V1#> ip link set up dev enp0s8`
- `V1#> ip addr add 192.168.1.1/24 dev enp0s8`
- `V2#> dhclient -v enp0s3`
- `V2#> ip link set up dev enp0s8`
- `V2#> ip addr add 192.168.1.2/24 dev enp0s8`

Następnie wykonałem pierwszy tutorial.
Na maszynie *Virbian2* wykonałem polecenie:

```
V2$> ssh -N -L 7777 :localhost : 7 user@192.168.1.1
```

a po wpisaniu polecenia:

```
V2$> telnet localhost 7777
```

odpowiadał serwer echa maszyny *Virbian1*.
Następnie przyjrzałem się pakietom w Wiresharku na obu maszynach.

No.	Time	Source	Destination	Protocol	Length	Info
44	48.561664032	127.0.0.1	127.0.0.1	TCP	68	40680 → 7 [ACK] Seq=4 Ack=4 Win=65536 Len=0
45	48.561693317	192.168.1.1	192.168.1.2	SSHv2	112	Server: Encrypted packet (len=44)
46	48.561931712	192.168.1.2	192.168.1.1	TCP	68	49274 → 22 [ACK] Seq=2814 Ack=2914 Win=64128
47	122.775663650	192.168.1.2	192.168.1.1	SSHv2	112	Client: Encrypted packet (len=44)
48	122.775766730	127.0.0.1	127.0.0.1	ECHO	71	Request
49	122.775790931	127.0.0.1	127.0.0.1	ECHO	71	Response
50	122.775796540	127.0.0.1	127.0.0.1	TCP	68	40680 → 7 [ACK] Seq=7 Ack=7 Win=65536 Len=0
51	122.775819483	192.168.1.1	192.168.1.2	SSHv2	112	Server: Encrypted packet (len=44)
52	122.776134633	192.168.1.2	192.168.1.1	TCP	68	49274 → 22 [ACK] Seq=2858 Ack=2958 Win=64128
53	127.916004280	PcsCompu-ee:17:df		ARP	62	Who has 192.168.1.1? Tell 192.168.1.2
54	127.916033630	PcsCompu-eb:dc:01		ARP	44	192.168.1.1 is at 08:00:27:eb:dc:01
55	127.949688776	PcsCompu-eb:dc:01		ARP	44	Who has 192.168.1.2? Tell 192.168.1.1
56	127.950088217	PcsCompu-ee:17:df		ARP	62	192.168.1.2 is at 08:00:27:ee:17:df
57	131.690739891	192.168.1.2	192.168.1.1	SSHv2	112	Client: Encrypted packet (len=44)
58	131.690845433	127.0.0.1	127.0.0.1	ECHO	71	Request
59	131.690868657	127.0.0.1	127.0.0.1	ECHO	71	Response
60	131.690874499	127.0.0.1	127.0.0.1	TCP	68	40680 → 7 [ACK] Seq=10 Ack=10 Win=65536 Len=0
61	131.690895431	192.168.1.1	192.168.1.2	SSHv2	112	Server: Encrypted packet (len=44)
62	131.691155580	192.168.1.2	192.168.1.1	TCP	68	49274 → 22 [ACK] Seq=2902 Ack=3002 Win=64128

SSH Protocol	
SSH Version 2 (encryption:chacha20-poly1305@openssh.com mac:<implicit> compression:none)	
Packet Length (encrypted): 8851261a	
Encrypted Packet: 9a69b0cdf07fd3d393ea9132f191366850a24333613507a	
MAC: 256fb28b2cf5271fb16a158868a8c087	

Offset	Hex	ASCII
0000	00 04 00 01 00 06 08 00 27 eb dc 01 00 00 08 00@.....
0010	45 00 00 00 0e 7c 40 00 40 06 a8 c8 c0 a8 01 01	E.....@.....
0020	c0 a8 01 02 00 16 c0 7a 95 2e 1c 9a f6 65 13 79Z.....e..y
0030	00 18 01 f5 83 a6 00 00 01 01 08 9a a6 94 d7 f6z.....e..y
0040	19 5d 1a 2c 88 51 26 1a 9a 69 b0 cd f0 7f d3 d3i.....
0050	93 ea 91 32 f1 91 36 68 50 a2 43 33 36 13 50 7a	...2..6h P.C36-Pz
0060	25 6f b2 8b 2c f5 27 1f b1 6a 15 88 68 a8 c0 87	%o...j..h...

- Na maszynie *Virbian2* strumień danych występuje w postaci niezasyfrowanej jako pakiety przesyłane z :: 1 port 48492 do :: 1 port 7777, (IPv6)
- Pomiedzy maszyną *Virbian2* a maszyną *Virbian1* strumień danych występuje w postaci zaszyfrowanej jako pakiety przesyłane z 192.168.1.2 port 49274 do 192.168.1.1 port 22 (IPv4)
- Na maszynie *Virbian1* strumień danych występuje w postaci niezasyfrowanej jako pakiety przesyłane z *localhost* port 40680 do *localhost* port 7 (IPv4)

2 Drugie zadanie do zaprezentowania

Na maszynie *Virbian1* zapisałem klucz publiczny użytkownika user1 za pomocą polecenia:

```
V1$> gpg -a --export user1 > user1-pgp-key
```

Następnie na maszynie *Virbian2* wygenerowałem klucze dla użytkownika user2 i zapisałem go do pliku za pomocą poleceń:

- V2\$> gpg --gen-key
- V2\$> gpg -a --export user2 > user2-pgp-key

```
user@virbian: ~  
Change (N)ame, (E)mail, or (O)key/(Q)uit? 0  
We need to generate a lot of random bytes. It is a good idea to perform  
some other action (type on the keyboard, move the mouse, utilize the  
disks) during the prime generation; this gives the random number  
generator a better chance to gain enough entropy.  
We need to generate a lot of random bytes. It is a good idea to perform  
some other action (type on the keyboard, move the mouse, utilize the  
disks) during the prime generation; this gives the random number  
generator a better chance to gain enough entropy.  
gpg: /home/user/.gnupg/trustdb.gpg: trustdb created  
gpg: key E0BB12EC52845699 marked as ultimately trusted  
gpg: directory '/home/user/.gnupg/openpgp-revocs.d' created  
gpg: revocation certificate stored as '/home/user/.gnupg/openpgp-revocs.d/C98F15  
090B8B4E071253A1D8E0BB12EC52845699.rev'  
public and secret key created and signed.  
  
pub   rsa3072 2020-06-07 [SC] [expires: 2022-06-07]  
       C98F15090B8B4E071253A1D8E0BB12EC52845699  
uid           user2 <user2@mail.example.com>  
sub   rsa3072 2020-06-07 [E] [expires: 2022-06-07]  
  
user@virbian:~$ gpg -a --export user2 > user2-gpg-key  
user@virbian:~$
```

Następnie skopiowałem plik user1-gpg-key na maszynę *Virbian2*, a następnie zaimportowałem go do kluczy użytkownika user2 oraz podpisałem kluczem prywatnym użytkownika user2 za pomocą poleceń:

- V1\$> scp user1-gpg-key 192.168.1.2 :user1-gpg-key
- V1\$> gpg --import < user1-gpg-key
- V1\$> gpg --edit-key identyfikator-klucza
- gpg> fpr
- gpg> sign
- gpg> quit

```
user@virbian: ~  
user@virbian:~$ gpg --edit-key 622F4A544370CF74FB9AADC0FD90FC8F03C382A6  
gpg (GnuPG) 2.2.12; Copyright (C) 2018 Free Software Foundation, Inc.  
This is free software: you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law.  
  
pub rsa3072/FD90FC8F03C382A6  
  created: 2020-06-07  expires: 2022-06-07  usage: SC  
  trust: unknown      validity: unknown  
sub rsa3072/0E8AA0124C95E052  
  created: 2020-06-07  expires: 2022-06-07  usage: E  
[ unknown] (1). user1 <user1@mail.example.com>  
  
gpg> fpr  
pub rsa3072/FD90FC8F03C382A6 2020-06-07 user1 <user1@mail.example.com>  
  Primary key fingerprint: 622F 4A54 4370 CF74 FB9A ADC0 FD90 FC8F 03C3 82A6  
  
gpg> sign  
  
pub rsa3072/FD90FC8F03C382A6  
  created: 2020-06-07  expires: 2022-06-07  usage: SC  
  trust: unknown      validity: unknown  
  Primary key fingerprint: 622F 4A54 4370 CF74 FB9A ADC0 FD90 FC8F 03C3 82A6  
  
    user1 <user1@mail.example.com>  
  
This key is due to expire on 2022-06-07.  
Are you sure that you want to sign this key with your  
key "user2 <user2@mail.example.com>" (E0BB12EC52845699)  
  
Really sign? (y/N) y  
gpg> █
```

Powyższe działania powtórzyłem na drugiej maszynie.

Następnie na maszynie *Virbian1* utworzyłem plik message z treścią message321, podpisałem go kluczem użytkownika user1 oraz zaszyfrowałem ją kluczem publicznym użytkownika user2. Następnie skopiowałem go na maszynę *Virbian2* za pomocą poleceń:

- V1\$> gpg -a -r user2 -se message
- V1\$> scp message.asc 192.168.1.2:message.asc

```
user@virbian:~/Downloads$ touch message  
user@virbian:~/Downloads$ vim message  
user@virbian:~/Downloads$ cat message  
message321  
user@virbian:~/Downloads$ gpg -a -r user2 -se message  
gpg: checking the trustdb  
gpg: marginals needed: 3  completes needed: 1  trust model: pgp  
gpg: depth: 0  valid: 1  signed: 2  trust: 0-, 0q, 0n, 0m, 0f, 1u  
gpg: depth: 1  valid: 2  signed: 0  trust: 2-, 0q, 0n, 0m, 0f, 0u  
gpg: next trustdb check due at 2022-06-07  
user@virbian:~/Downloads$ ls  
message      user1-gpg-key  veracrypt.deb  
message.asc  user2-gpg-key  veracrypt.deb.sig  
user@virbian:~/Downloads$ scp message.asc 192.168.0.2:message.asc  
user@192.168.0.2's password:  
message.asc                                100% 1329      1.7MB/s   00:00  
user@virbian:~/Downloads$ █
```

Na maszynie *Virbian2* otrzymany plik odszyfrowałem kluczem prywatnym użytkownika user2, a następnie wypisałem odszyfrowaną zawartość - była poprawna.

```
V2$> gpg -d message.asc > deciphered message
```

```
user@virbian:~$ ls
message.asc user1-pgp-key user2-pgp-key
user@virbian:~$ gpg -d message.asc > deciphered message
gpg: encrypted with 3072-bit RSA key, ID 2D5D590CE9A13B64, created 2020-06-07
"user2 <user2@mail.example.com>"
gpg: Signature made Sun Jun  7 18:16:14 2020 CEST
gpg:          using RSA key 622F4A544370CF74FB9AADC0FD90FC8F03C382A6
gpg: checking the trustdb
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: depth: 0 valid: 1 signed: 1 trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: depth: 1 valid: 1 signed: 0 trust: 1-, 0q, 0n, 0m, 0f, 0u
gpg: next trustdb check due at 2022-06-07
gpg: Good signature from "user1 <user1@mail.example.com>" [full]
user@virbian:~$ cat deciphered_message
message321
```