

به نام خدا



امنیت داده و شبکه

تمرین چهارم

استاد :

دکتر مهدی خرازی، دکتر مرتضی امینی، دکتر کامبیز میزانیان

نویسنده :

محمد هومان کشوری

شماره دانشجویی :

۹۹۱۰۵۶۶۷

سوال ۱

آ) برای اولین قسمت از پروتکل diffie-helman برای ساخت و تبادل کلید استفاده می‌کنیم.

$$C \rightarrow KDC : IP_S$$

$$KDC \rightarrow C : K_{C,S}$$

$$C \rightarrow S : E(K_{C,S}, \alpha || q || Y_C = (\alpha^{X_C} \bmod q))$$

$$S \rightarrow C : E(K_{C,S}, Y_S = (\alpha^{X_S} \bmod q))$$

$$K' = Y_C^{X_S} \bmod q = Y_S^{X_C} \bmod q$$

حال عملاً با دیفی هلمن توانستیم کلیدی ایجاد کنیم که KDC از آن اطلاعی ندارد اما طرفین آن را می‌دانند.

ب) در صورتی که KDC توان تغییر ترافیک را داشته باشد می‌تواند حمله Man-In-The-Middle را پیاده‌سازی کند و عملاً خود را برای S جای C و برای C جای S جا بزند.

$$C \rightarrow KDC : IP_S$$

$$KDC \rightarrow C : K_{C,S}$$

$$C \rightarrow S : E(K_{C,S}, \alpha || q || Y_C = (\alpha^{X_C} \bmod q))$$

$$KDC \text{ Intercept} - KDC \rightarrow S : E(K_{C,S}, \alpha || q || Y'_C = (\alpha^{X_{KDC}} \bmod q))$$

$$S \rightarrow C : E(K_{C,S}, Y_S = (\alpha^{X_S} \bmod q))$$

$$KDC \text{ Intercept} - KDC \rightarrow C : E(K_{C,S}, \alpha || q || Y'_S = (\alpha^{X_{KDC}} \bmod q))$$

ج) می‌دانیم KDC ها با یکدیگر امکان تبادلی ندارند پس منطقاً از کلیدهای یکدیگر نیز خبر ندارند. تنها کاری که لازم است انجام دهیم این است که K' را با هر دو کلید رمز کنیم.

$$C \rightarrow S : E(K_{C,S,2}, E(K_{C,S,1}, K'))$$

حال در صورتی که یکی از KDC ها پیام را باز کند، نمی‌تواند متوجه پیام شود چون KDC_1 نمی‌تواند پیام بیرونی را باز کند و KDC_2 نیز نمی‌تواند پیام داخلی را باز کند.

سوال ۲

برای راه اندازی pgp مراحل زیر را دنبال می کنیم. قابل ذکر است که سیستم عامل ما Ubuntu 22.04 می باشد. ابتدا برای نصب از دستور زیر استفاده می کنیم:

```
1 apt install gap gnupg2
```

2

سپس باید جفت کلید خود را جنریت کنیم.

```
Welcome to fish, the friendly interactive shell
Type help for instructions on how to use fish
bigwhoman@bigwhoman /m/b/l/S/t/S/D/HW4 (main)> gpg --generate-key
gpg (GnuPG) 2.2.27; Copyright (C) 2021 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.

Note: Use "gpg --full-generate-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name: Mohammad Hooman Keshvari
Email address: hooman.keshvari@sharif.edu
You selected this USER-ID:
    "Mohammad Hooman Keshvari <hooman.keshvari@sharif.edu>"

Change (N)ame, (E)mail, or (O)kay/(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: key 12226338D3536D1C marked as ultimately trusted
gpg: revocation certificate stored as '/home/bigwhoman/.gnupg/openpgp-revocs.d/70C6A76A
4923F7C6A8547CE112226338D3536D1C.rev'
public and secret key created and signed.
```

بعد از جنریت کردن جفت کلیدهای خود، باید کلید عمومی را با زدن دستور زیر import کنیم.

```
1 gpg --import Reza_0xCFBEEE88_public.asc
```

سپس با زدن دستورات زیر ابتدا plain text را امضا کرده و سپس آنرا encrypt می کنیم.

```

bigwhoman@bighwhoman /m/b/l/S/t/S/D/HW4 (main)> echo -e "Subject: Mohammad Hooman Keshvari\n\n99105667"
| gpg --clearsign --armor > message
bigwhoman@bighwhoman /m/b/l/S/t/S/D/HW4 (main)> gpg --encrypt --armor --recipient Reza message
gpg: 5F4AB93836D3E77F: There is no assurance this key belongs to the named user

sub cv25519/5F4AB93836D3E77F 2023-12-26 Reza <reza.saeedi9@yahoo.com>
Primary key fingerprint: 981C 65B7 BA35 83E2 6981 A39B B334 7414 CFBE EE88
Subkey fingerprint: 4395 AAE0 6458 29FC 1ACF 8443 5F4A B938 36D3 E77F

It is NOT certain that the key belongs to the person named
in the user ID. If you *really* know what you are doing,
you may answer the next question with yes.

Use this key anyway? (y/N) y

```

با دستورات بالا یک فایل message.asc ساخته می‌شود که حاوی پیام و امضا شده آن است که در سمت گیرنده با دستور زیر می‌توان به خود پیام رسید.

```
1 gpg --decrypt --output message.txt message.asc
```

در صورتی که امضا درست باشد باید با زدن دستور زیر خروجی مانند خروجی زیر را مشاهده کنید.
ساخته می‌شود که ابتدا باید مطمئن شویم درست امضا شده :

```

bigwhoman@bighwhoman /m/b/l/S/t/S/D/HW4 (main)> gpg --verify message
gpg: Signature made 0330+ ۱۲:۴۴:۴۷ ، ۲۴ چارشنبه ۱۰ ژانویه ۱۴۰۴
gpg: using RSA key 70C6A76A4923F7C6A8547CE112226338D3536D1C
gpg: Good signature from "Mohammad Hooman Keshvari <hooman.keshvari@sharif.edu>" [ultimate]

```

با زدن دستور زیر نیز کلید عمومی خود را در فایل keshvari.asc قرار می‌دهیم و آنرا به ایمیل ضمیمه می‌کنیم.

```
1 gpg --armor --output keshvari.asc --export "Mohammad Hooman Keshvari"
```

- آ) ● جست و جوی فراگیر: در این حمله، حمله کننده عملاً تمامی ترکیبات ممکن برای شکستن رمزنگاری را امتحان می کند. چون IPSec عملاً یک لایه رمزنگاری به قسمت بالایی خود اضافه می کند پس شکستن دو لایه رمزنگاری کار ساده ای (حدافل با کامپیوترهای امروزی) نخواهد بود.
- متن اصلی معلوم: در این حمله، حمله کننده یک plain text را به همراه رمز شده آن دارد و عملاً می تواند به رابطه بین این دو پی ببرد. در IPSec باز هم چون رمزنگاری های ما پارامتر تصادفی بودن را دارند پس حمله کننده نمی تواند به رابطه بین متن اصلی و رمزنگاری شده پی ببرد.
- حمله تکرار: حمله ای است که حمله کننده در آن یک بسته را گرفته و آنرا عیناً دوباره ارسال می کند. در هر دو سرویس AH و ESP در IPSec پارامترهای SA مربوطه یک Sequence Number Counter گذاشته می شود و همچنین یک Anti Replay Window که پنجره بسته های ارسال شده را دارد و اگر فرض کنیم بسته های m تا n را پوشش می دهد و آمار آنها را دارد، در صورتی که بسته $n + 1$ بیاید، پنجره را یکی به جلو می برد. پس می توان مطمئن بود که بسته های بزرگتر از n که هنوز نیامده اند. بسته های بین m تا n نیز وضعیت آنها مشخص است و در صورتی که بسته کمتر از m بیاید، آنرا تکراری در نظر می گیریم و اجازه عبور نمی دهیم.
- شنود رمز عبور: حمله ای که در آن، حمله کننده با استفاده از ابزارهای متفاوت اطلاعات حساس و پسوردها را در حین عبور در شبکه شنود می کند. پروتکل ESP در IPSec به این علت که عملاً قسمت data را به صورت کامل encrypt می کند، بجز خود و مقصد، هیچ کس در این بین نمی تواند از محتویات بسته اطلاع یابد. در بین مودهای انتقال و تونل عملاً مود انتقال چون end-to-end است بهتر است چون در مود تونل عملاً چون دو سمت تونل یک دور کل بسته را باز می کنند، خود می توانند محتویات data را ببینند مگر اینکه محتویات data را نیز به صورت جداگانه رمز کنیم.
- جعل IP: حمله ای که در آن، حمله کننده قسمت Source IP بسته ها را عوض کرده تا بنظر برسد مبدا متفاوتی دارند. در پروتکل AH، عملاً بخش هایی از سرایند IP و بخش data در IP یک MAC گرفته می شود که در صورت تغییر هر کدامیک از آنها متوجه تغییر آن می شویم. این کارکرد برای هر دو مود تونل و انتقال در AH صادق است. همچنین این پروتکل از cookie استفاده می کند که صحت سنجی برقراری ارتباط صرفاً با استفاده از صاحب اصلی ای پی ممکن است.
- سرقت IP: حمله ای که در آن حمله کننده کنترل شبکه فرد را به دست می گیرد و و عملاً خود را به جای وی جا می زند که باعث می شود بسته ها به جای صاحب اصلی به حمله کننده برسند. چون از cookie در ارتباط استفاده می کنیم، عملاً سرور کوکی را ابتدا برای صاحب اصلی ای پی ارسال می کند و سرور نیز نمی تواند آنرا ببیند.
- حمله SYN flooding: حمله ای است که حمله کننده در مرحله سوم handshake در لایه انتقال، به جای فرستادن ACK برای کارگزار، دوباره یک SYN می فرستد و عملاً کارگزار هیچ گاه ACK مربوطه از کلاینت را دریافت نمی کند و این باعث dos می شود. IPSec عملاً با استفاده از Cookie جلوی این امر را می گیرد. البته مطابق این لینک چون این حمله در لایه بالاتر یعنی لایه انتقال اتفاق می افتد، عملاً IPSec مسئول جلوگیری از این حمله نیست.
- ب) برای راه اندازی پروتکل IPSec بر روی سیستم عامل لینوکس، ابتدا مطابق زیر پکیج های خواسته شده را نصب می کنیم.

```
1 sudo apt update
2 sudo apt install strongswan
```

سپس برای تغییر تنظیمات باید این دو فایل زیر را تغییر دهیم:

```
/etc/ipsec.conf ●
/etc/ipsec.secret ●
```

محتویات تنظیمات ماشین مجازی اول با ای پی 192.168.2.189:

```

1 config setup
2     uniqueids=no
3     charondebug=ike 2
4
5
6 # Add connections here.
7
8 conn %default
9     ikelifetime=60m
10    keylife=20m
11    keyingtries=1
12    ike=aes128-sha256-modp1024 , aes256-sha384-modp1024 !
13    keyexchange=ikev2
14
15 conn ubuntu
16     authby=secret
17     leftsubnet=192.168.2.0/24
18     leftsendcert=never
19     right=192.168.2.188
20     rightsubnet=192.168.2.0/24
21     auto=start

```

محتویات تنظیمات ماشین مجازی اول با ای پی 192.168.2.188 :

```

1 config setup
2     uniqueids=no
3     charondebug=ike 2
4
5
6 # Add connections here.
7
8 conn %default
9     ikelifetime=60m
10    keylife=20m
11    keyingtries=1
12    ike=aes128-sha256-modp1024 , aes256-sha384-modp1024 !
13    keyexchange=ikev2
14
15 conn ubuntu
16     authby=secret
17     leftsubnet=192.168.2.0/24
18     right=192.168.2.189
19     rightsubnet=192.168.2.0/24
20     auto=start

```

حال در نهایت باید فایل /etc/ipsec.secrets را تغییر دهیم تا دو ماشین مجازی بتوانند یکدیگر را authenticate کنند. محتوایات ipsec.secrets در هر دو ماشین مجازی :

```

1 192.168.2.189 192.168.2.188 : PSK "123"

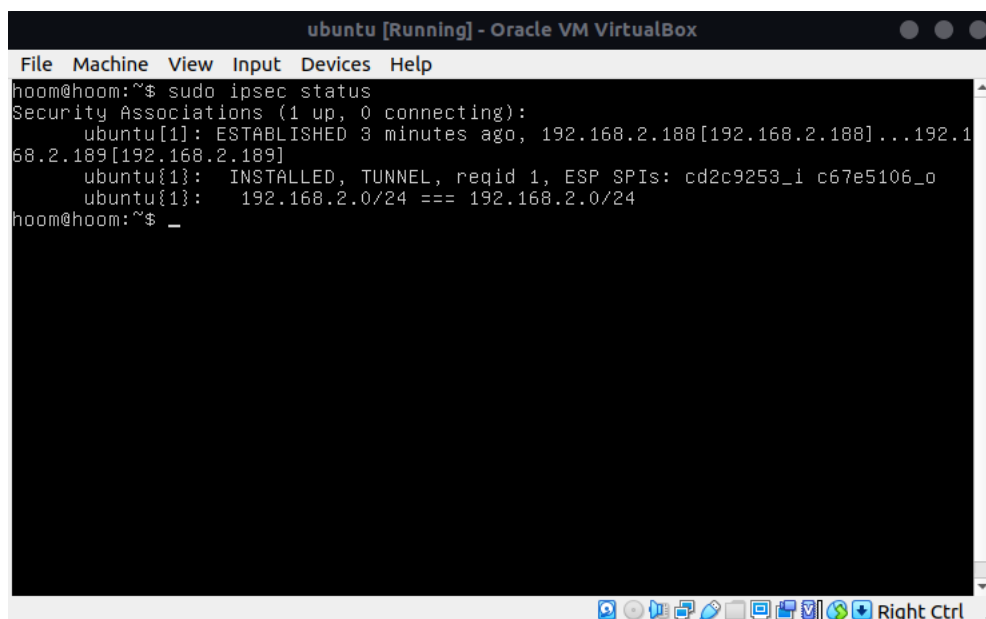
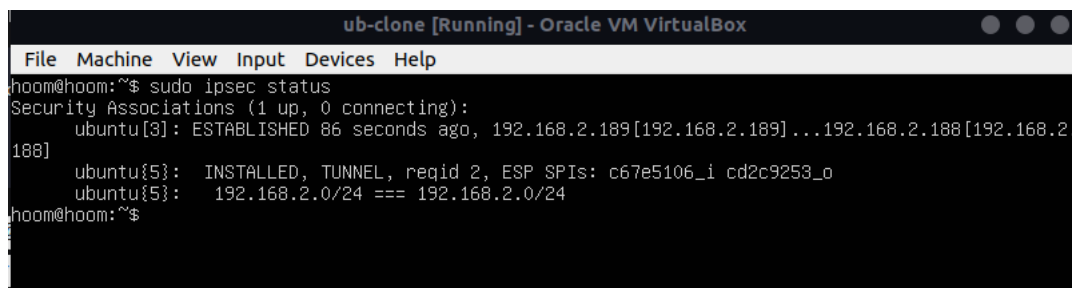
```

در بالا، یک کلید مشترک^۱ "۱۲۳" بین دو ماشین مجازی تنظیم شده است. حال با زدن دستورات زیر ipsec ران می‌شود.

```
1 sudo ipsec rereadsecrets
2 sudo ipsec start
```

در ماشین‌های مجازی با زدن دستور زیر می‌توانیم وضعیت دو ماشین را ببینیم:

```
1 sudo ipsec status
```



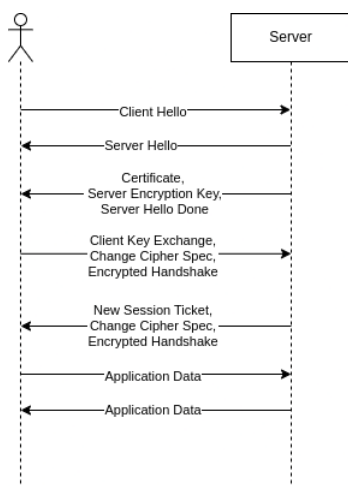
Pre-Shared Key^۱

سوال ۴

آ ابتدا عکس وایرشارک را در این قسمت قرار می‌دهیم.

tcp.stream eq 2 && tls						
No.	Time	Source	Destination	Protocol	Length	Info
414	1.336038785	172.27.54.245	204.12.192.218	TLSv1.2	733	Client Hello
446	1.848561125	204.12.192.218	172.27.54.245	TLSv1.2	2962	Server Hello
450	1.848723727	204.12.192.218	172.27.54.245	TLSv1.2	1574	Certificate, Server Key Exchange, Server Hello Done
470	1.853518266	172.27.54.245	204.12.192.218	TLSv1.2	192	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
493	2.463720462	204.12.192.218	172.27.54.245	TLSv1.2	324	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
532	3.080497643	172.27.54.245	204.12.192.218	TLSv1.2	541	Application Data
545	3.385005099	204.12.192.218	172.27.54.245	TLSv1.2	703	Application Data
548	3.442619426	172.27.54.245	204.12.192.218	TLSv1.2	480	Application Data
578	4.024484311	204.12.192.218	172.27.54.245	TLSv1.2	1873	Application Data
3603	70.356199424	172.27.54.245	204.12.192.218	TLSv1.2	97	Encrypted Alert

فریم اول را کلاینت به سمت سرور ارسال کرده که عملاً Client Hello در tls handshake است و پاسخ آنرا سرور با فریم دوم یعنی Server Hello داده است. سپس سرور و کلاینت در دو قسمت بعدی با یکدیگر تبادل کلید کرده‌اند و نهایتاً سرور یک Session Ticket ساخته و برای کلاینت ارسال می‌کند و بعد از آن صحبت کلاینت و سرور آغاز می‌شود. دیاگرام به صورت زیر است :



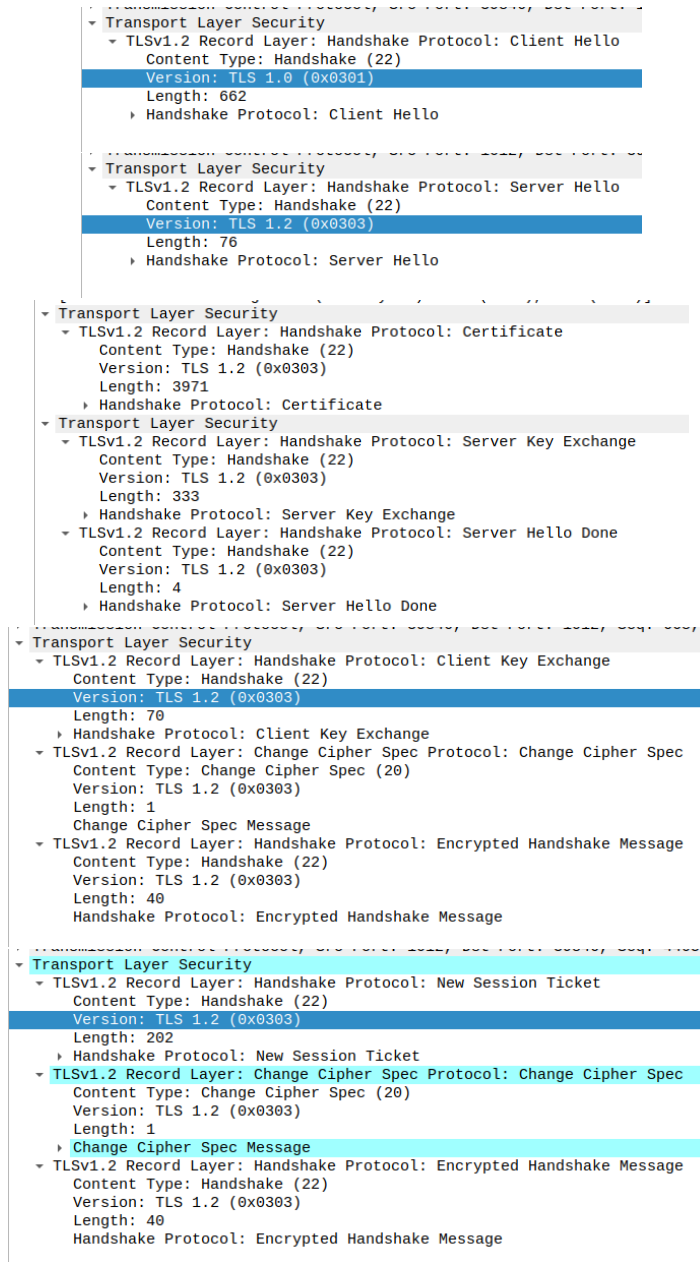
حال طول فیلدها را مشخص می‌کنیم.

- Client Hello : ۶۶۲ بایت
- Server Hello : ۷۶ بایت
- Certificate : ۳۹۷۱ بایت
- Server Key Exchange : ۳۲۹ بایت
- Server Hello Done : ۴ بایت
- Client Key Exchange : ۷۰ بایت
- Change Cipher Spec : ۱ بایت
- Encrypted Handshake (Server) : ۴۰ بایت

● New Session Ticket : ۲۰۲ بایت

● Change Cipher Spec : ۱ بایت

● Encrypted Hanshake : ۴۰ بایت



ب Client Hello

● Handshake : Content Type

- Nonce (Challenge) : اشاره به مقدار **random** دارد که عملاً یک عدد رندوم 2^{32} بایتی دارد که برای این به کار می‌رود که سرور آن را عوض کرده و کلاینت مطمئن شود که حمله replay attack رخ نداده است. همچنین در ابتدای هر session این عدد به صورت یکتا برای همان session ساخته می‌شود که عملاً در صورتی که حمله‌کننده به سشن‌های قبلی دست یابد نمی‌تواند آنها را بازنساز کند. همچنین از این رندوم برای ساخت Master Key در قدم‌های بعدی استفاده می‌شود.
- Cipher Suites : مجموعه‌ای از الگوریتم‌های رمزنگاری پشتیبانی شده. می‌توانید لیست آنها را در عکس زیر مشاهده کنید.

```

▼ TLSv1.2 Record Layer: Handshake Protocol: Client Hello
  Content Type: Handshake (22)
  Version: TLS 1.0 (0x0301)
  Length: 662
▼ Handshake Protocol: Client Hello
  Handshake Type: Client Hello (1)
  Length: 658
  Version: TLS 1.2 (0x0303)
  Random: cb4061534103328b90b70dc907a75934e677f207e8f3df8e913ccf27aa70c416
  Session ID Length: 32
  Session ID: 2565c6195a5d8f1aaa616e373ee53e7258cfc1c8b0a38210a1ddacd37dedb1c7
  Cipher Suites Length: 34
▼ Cipher Suites (17 suites)
  Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
  Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
  Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
  Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xc030)
  Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xc031)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc032)
  Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc033)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc034)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc035)
  Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc036)
  Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc037)
  Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x003d)
  Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x003e)
  Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x003c)
  Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x003f)

```

ج) Server Hello

- Cipher Suite : TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
- Nounce : مانند Client Hello یک Random که به آن Server Random می‌گوییم ساخته می‌شود و به کلاینت ارسال می‌شود.
- Session ID : یک مقدار یکتا است که برای resume کردن سشن‌های قبلی به کار می‌رود و عملاً باعث می‌شود که یک handshake کامل دیگر نیاز نباشد. در این کانکشن ما Session ID نداشتیم که به این معنی است که از قبل سشنی وجود نداشته و یا سرور Session Resume را ساپورت نمی‌کند.

pseudo-random^۲

```

▼ Transport Layer Security
  ▼ TLSv1.2 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 76
    ▼ Handshake Protocol: Server Hello
      Handshake Type: Server Hello (2)
      Length: 72
      Version: TLS 1.2 (0x0303)
      ▶ Random: 952aa632f0b91b97583bc77721c6c552bc89f3902441d499f6980ea04e4447da
      Session ID Length: 0
      Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
      Compression Method: null (0)
      Extensions Length: 32
      ▶ Extension: renegotiation_info (len=1)
      ▶ Extension: ec_point_formats (len=4)
      ▶ Extension: session_ticket (len=0)
      ▶ Extension: application_layer_protocol_negotiation (len=11)
      [JA3S Fullstring: 771,49200,65281-11-35-16]
      [JA3S: 2b33c1374db4ddf06942f92373c0b54b]

```

● Certificate : در فریم دیگری ارسال شده که در قسمت بعدی آنرا توضیح می‌دهیم.

Certificate (د)

- الگوریتم رمزنگاری کلید عمومی : RSAEncryption
- طول کلید عمومی : از قسمت subjectPublicKey قابل محاسبه است که ۲۰۴۸ بیت است.
- الگوریتم امضا : sha256WithRSAEncryption
- مقدار امضا : در پایین همان قسمت در encrypted در عکس زیر می‌توانید آنرا به فرمت C Array مشاهده کنید.

```

0x19, 0x89, 0x39, 0x0d, 0xeb, 0x14, 0x6e, 0xb8,
0x16, 0xdf, 0x90, 0x20, 0x03, 0x85, 0xf5, 0xca,
0x68, 0xa3, 0xc6, 0x95, 0x90, 0x91, 0xbf, 0xac,
0xd9, 0x6c, 0x15, 0xb1, 0xa9, 0x88, 0x69, 0xc5,
0x59, 0x5a, 0x2e, 0xd3, 0xd0, 0xe7, 0x72, 0xb8,
0x1b, 0x97, 0x3a, 0x58, 0x1b, 0xd1, 0x07, 0x57,
0x6d, 0x4a, 0x1d, 0x5a, 0x02, 0xe6, 0x35, 0x15,
0x61, 0x79, 0x0c, 0x36, 0xa7, 0x26, 0xe6, 0xa1,
0x95, 0x54, 0xf3, 0xa1, 0x14, 0x74, 0x11, 0xb7,
0xc0, 0x10, 0xb7, 0x1b, 0x9e, 0x26, 0xd2, 0x62,
0x29, 0x45, 0x92, 0xb1, 0xd3, 0xf6, 0x06, 0xc3,
0xf9, 0xa5, 0x95, 0x7b, 0x04, 0x9a, 0xae, 0xe1,
0xf9, 0x9c, 0x0c, 0x06, 0xbe, 0x2d, 0xf4, 0x78,
0x20, 0xc3, 0xae, 0x90, 0xba, 0x4e, 0x10, 0x64,
0x8d, 0xd6, 0x40, 0xdf, 0x7f, 0x49, 0xa3, 0x93,
0x6a, 0x0c, 0xc8, 0xb1, 0x24, 0xd5, 0x1d, 0xe2,
0x22, 0x86, 0x52, 0x9d, 0x0a, 0x78, 0xae, 0x52,
0x90, 0xc5, 0x42, 0xd2, 0x36, 0xef, 0xf9, 0xde,
0x7f, 0x68, 0x14, 0xd7, 0xc1, 0x68, 0x61, 0xde,
0xbe, 0x5c, 0x74, 0xd0, 0xe5, 0xda, 0x43, 0x90,
0x94, 0x35, 0xb3, 0x60, 0xf9, 0x2e, 0x03, 0xe2,
0xde, 0x43, 0x13, 0x8b, 0xc1, 0x2f, 0x41, 0x1b,
0x5b, 0x9f, 0xef, 0x80, 0x07, 0x2c, 0x1c, 0x72,
0x0b, 0x5d, 0x33, 0xdc, 0x4a, 0xa9, 0x2b, 0xdf,
0x43, 0xc3, 0x62, 0x3a, 0x72, 0xfe, 0xfc, 0x27,
0x36, 0xc2, 0x3b, 0x37, 0x9e, 0xca, 0x2d, 0x57,
0xad, 0x4f, 0x4b, 0x52, 0xad, 0x48, 0xd4, 0x24,
0xd8, 0x0b, 0x8c, 0x10, 0xb8, 0x03, 0x4e, 0xf8,
0xda, 0xf9, 0x3b, 0xba, 0x62, 0x77, 0x09, 0x04,
0xfa, 0x67, 0x7c, 0x72, 0x52, 0xa7, 0xe7, 0x16,
0x5b, 0x89, 0x7b, 0xfc, 0xa7, 0x63, 0x6d, 0x8f,
0x07, 0xa7, 0xc9, 0x4d, 0x16, 0xef, 0xda, 0xab

```

● صادرکننده گواهی : می‌توانید در عکس زیر صادرکننده گواهی را ببینید :

```

▼ issuer: rdnSequence (0)
  ▼ rdnSequence: 3 items (id-at-commonName=R3,id-at-organizationName=Let's Encrypt,id-at-...
    ▶ RDNSSequence item: 1 item (id-at-countryName=US)
    ▶ RDNSSequence item: 1 item (id-at-organizationName=Let's Encrypt)
    ▶ RDNSSequence item: 1 item (id-at-commonName=R3)

```

Client Key Exchange (6)

- Pre-Master Secret : مقداری که به صورت مستقیم از تبادل کلیدهای عمومی بدست می‌آید. مثلاً این مقدار در تبادل کلید Diffie-Hellman از این رابطه بدست می‌آید : $g^{ab} \bmod p$ در تبادل کلید به روش دیفی هلمن این مقدار عملاً همان Pubkey ارسال شده است. در عکس زیر می‌توانید این Pubkey را مشاهده کنید.

```
▼ Transport Layer Security
  ▼ TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 70
    ▼ Handshake Protocol: Client Key Exchange
      Handshake Type: Client Key Exchange (16)
      Length: 66
      ▼ EC Diffie-Hellman Client Params
        Pubkey Length: 65
        Pubkey: 04811bad330d969b75ac421370cd6b3d48fbde2576d5516a877fa930e67540c05a356b55...
```

- 7 Change Cipher Spec : این پیام از طرف سرور و کلاینت برای یکدیگر ارسال می‌شود تا مشخص کند که تمامی پیام‌های آینده به واسطه Cipher Suite مشخص شده رمزنگاری می‌شوند. مقداری که سرور ارسال می‌کند با مقداری که کلاینت ارسال می‌کند یکسان نیست. دلیل این تفاوت این است که طرفی که پیام اولیه را ارسال کرده مطمئن شود پیام دریافتی تازه و عملاً بازارسال نشده است.

- 8 Application Data : پس از تبادل کلید و طی شدن مراحل handshake با الگوریتم رمزنگاری مشخص شده که در این مورد مثلاً AES است رمزنگاری داده انجام شده و از آن به بعد ارتباط کلاینت و سرور با آن و کلید مشترکشان رمزنگاری می‌شود.

سوال ۵

با توجه به اینکه تمامی ورودی‌ها بجز تعداد ورودی محدود باز هستند، پس تنها راه منع خدمت ورود از آن مسیرها است پس برای جلوگیری از dos در هر قسمت موقع تعریف قوانین آن روش جلوگیری از داس زدن روی آن مورد را نیز بیان می‌کنیم برای حل این سوال ابتدا باید ترافیک ورودی را کامل ببندیم.

```
1 sudo iptables -P INPUT DROP
```

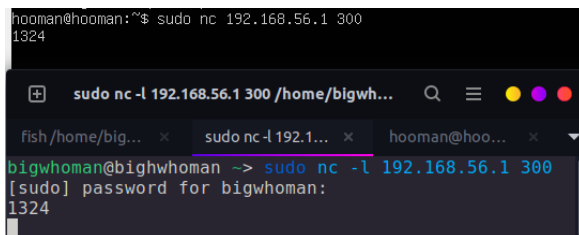
```
hooman@hooman:~$ sudo iptables -L --line-numbers
Chain INPUT (policy DROP)
num target prot opt source destination
Chain FORWARD (policy ACCEPT)
num target prot opt source destination
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
```

سپس باید ورودی‌های کانکشن‌های Establish شده را باز کنیم.

```
1 sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

می‌توانیم تست کنیم که آیا قانون ما درست کار می‌کند یا خیر. برای این کار از nc استفاده می‌کنیم تا ببینیم در صورت وصل شدن به بیرون آیا می‌توانیم کانکشن ایجاد کنیم یا خیر. همانطور که از عکس مشاهده می‌کند توانسته‌ایم از ای پی مورد نظر به ای پی دیگر با استفاده از nc وصل شویم و یک پیام نیز بفرستیم.

```
hooman@hooman:~$ sudo nc 192.168.56.1 300
1324
```



سپس باید قوانین ضد dos را تنظیم کنیم تا اگر فردی با کانکشن‌های ورودی سعی به dos کردن ما داشت جلوی وی گرفته شود. دستورات زیر را استفاده می‌کنیم.

```
1 sudo iptables -A INPUT -p tcp -m connlimit --connlimit-above 3 --connlimit-mask 32 -j REJECT
```

با این کار عملاً اجازه داده نمی‌شود یک ای پی بیش از ۳ درخواست در دقیقه برای ما ارسال کند. حال باید پورت ssh یا همان پورت ۲۲ را باز کنیم اما برای این کار باید ابتدا مطمئن شویم کسی به این پورت عملاً حمله نمی‌کند و پورت را بروت فورس نمی‌کند. برای این کار دستورات زیر را می‌زنیم:

```
1 sudo iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW -m recent --set
2 sudo iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW -m recent --update --seconds 60 --hitcount 10 -j DROP
3 sudo iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW, ESTABLISHED -j ACCEPT
```

در ۳ دستور بالا، اولین دستور عملاً کانکشن‌هایی که به پورت ۲۲ زده می‌شوند را ضبط می‌کند، دومی می‌گوید در هر ۶۰ ثانیه فقط ۱۰ کانکشن جدید به این پورت می‌تواند زده شود و سومی عملاً می‌گوید پورت ۲۲ آماده کانکشن زدن است. اگر فردی با یک ای پی سعی کند در دقیقه بیش از ۱۰ کانکشن با پورت ۲۲ برقرار کند، عملاً کانکشن وی دراپ می‌شود. در عکس زیر می‌توانید این را ببینید.

```

bigwhoman@bighwhoman -> ssh -p 22 hooman@192.168.56.101
hooman@192.168.56.101's password:
bigwhoman@bighwhoman - [SIGINT]> ssh -p 22 hooman@192.168.56.101
hooman@192.168.56.101's password:
bigwhoman@bighwhoman - [SIGINT]> ssh -p 22 hooman@192.168.56.101
hooman@192.168.56.101's password:
bigwhoman@bighwhoman - [SIGINT]> ssh -p 22 hooman@192.168.56.101
hooman@192.168.56.101's password:
bigwhoman@bighwhoman - [SIGINT]> ssh -p 22 hooman@192.168.56.101
hooman@192.168.56.101's password:
bigwhoman@bighwhoman - [SIGINT]> ssh -p 22 hooman@192.168.56.101
hooman@192.168.56.101's password:
bigwhoman@bighwhoman - [SIGINT]> ssh -p 22 hooman@192.168.56.101
hooman@192.168.56.101's password:
bigwhoman@bighwhoman - [SIGINT]> ssh -p 22 hooman@192.168.56.101
hooman@192.168.56.101's password:
bigwhoman@bighwhoman - [SIGINT]> ssh -p 22 hooman@192.168.56.101
^C
bigwhoman@bighwhoman - [SIGINT]> ssh -p 22 hooman@192.168.56.101
^C
bigwhoman@bighwhoman - [SIGINT]>

```

در قسمت بعدی باید طوری تنظیم کنیم که پکت‌های icmp اجازه ورود داشته باشند اما حمله icmp flood نیز رخ ندهد و همچنین اجازه icmp redirect را ندهیم. برای این کار کامندهای زیر را استفاده می‌کنیم.

```

1 sudo iptables -A INPUT -p icmp --icmp-type redirect -j DROP
2 sudo iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit
  1/second --limit-burst 1 -j ACCEPT
3 sudo iptables -A INPUT -p icmp --icmp-type echo-request -j DROP

```

اولین قانون بالا می‌گوید icmp ها از مدل redirect را کلا دراپ کن در قسمت بعدی نیز می‌گوییم که کلا ۱ درخواست icmp در ثانیه می‌تواند به سیستم وارد شود و در غیر این صورت خط بعدی اجرا شده و درخواست icmp رد می‌شود. عکس زیر نشان می‌دهد که در قسمت اول جواب بسته icmp آمده و در قسمت بعدی پکت icmp دراپ شده است چون فاصله بین دو درخواست کمتر از ۱ ثانیه است.

```

bigwhoman@bighwhoman -> ping 192.168.56.101
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data:
64 bytes from 192.168.56.101: icmp_seq=1 ttl=64 time=0.592 ms
^C
--- 192.168.56.101 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.592/0.592/0.592/0.000 ms
bigwhoman@bighwhoman -> ping 192.168.56.101
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data:
^C
--- 192.168.56.101 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

```

در قسمت بعدی باید ترافیک وارد شده به پورت ۸۰ را به ۸۰۸۰ ری‌دایرکت کنیم.

```

1 sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-
  port 8080

```

نهایتا کل قوانین تعریف شده را می‌توانیم در شکل زیر مشاهده کنیم.

Chain INPUT (policy DROP)					hooman@hooman: ~	
num	target	prot	opt	source	destination	
1	ACCEPT	all	--	anywhere	anywhere	ctstate ESTABLISHED
2		tcp	--	anywhere	anywhere	tcp dpt:ssh ctstate NEW recent: SET name: DEFAU
LT	side: source	mask: 255.255.255.255				tcp dpt:ssh ctstate NEW recent: UPDATE seconds:
3	DROP	tcp	--	anywhere	anywhere	tcp dpt:ssh ctstate NEW, ESTABLISHED
60	hit count: 10	name: DEFAULT	side: source	mask: 255.255.255.255		icmp redirect
4	ACCEPT	tcp	--	anywhere	anywhere	icmp echo-request limit: avg 1/sec burst 1
5	DROP	icmp	--	anywhere	anywhere	icmp echo-request
6	ACCEPT	icmp	--	anywhere	anywhere	
7	DROP	icmp	--	anywhere	anywhere	
Chain FORWARD (policy ACCEPT)						
num	target	prot	opt	source	destination	
Chain OUTPUT (policy ACCEPT)						
num	target	prot	opt	source	destination	