به‌نام خدا

دانشگاه صنعتی شریف

دانشکده مهندسی کامپیوتر

امنیت داده و شبکه

تمرین ۴

مهراد میلانلو

۹۹۱۰۵۷۷۵

Pretty Good Problem ۱

(آ)

از تکه کد زیر برای تولید یک جفت کلید عمومی و خصوصی استفاده می‌کنیم:

```
from pgpy import PGPKey, PGPUID
from pgpy.constants import PubKeyAlgorithm, KeyFlags, HashAlgorithm \
, SymmetricKeyAlgorithm, CompressionAlgorithm

NAME = "Mehrad Milanloo"
EMAIL = "milanloomehrad@gmail.com"

# Generate a new key pair
key = PGPKey.new(PubKeyAlgorithm.RSAEncryptOrSign, 2048)
uid = PGPUID.new(NAME, email=EMAIL)

# Add the UID to the key
key.add_uid(uid, usage={KeyFlags.Sign
                        , KeyFlags.EncryptCommunications
                        , KeyFlags.EncryptStorage}
                        , hashes=[HashAlgorithm.SHA256]
                        , ciphers=[SymmetricKeyAlgorithm.AES256]
                        , compression=[CompressionAlgorithm.ZLIB])

# Export the public key
with open("public_key.asc", "w") as f:
    f.write(str(key.pubkey))

# Export the private key
with open("private_key.asc", "w") as f:
    f.write(str(key))
```

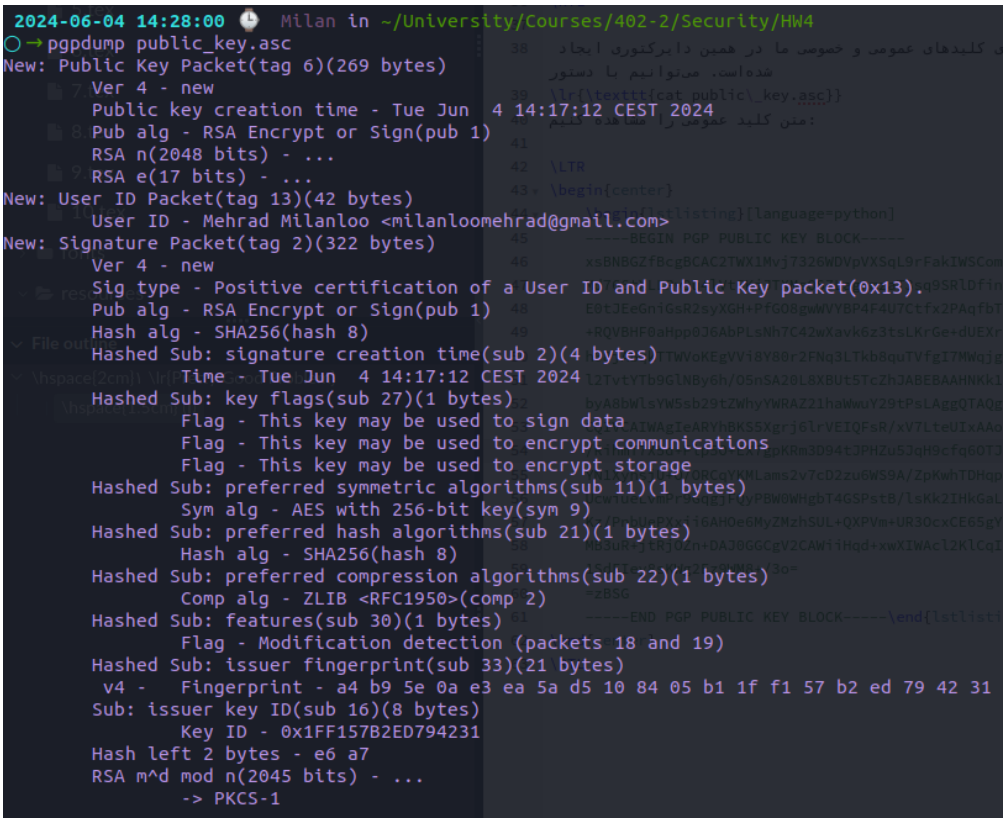
با ران کردن این کد مشاهده می‌کنیم که دو فایل حاوی کلیدهای عمومی و خصوصی ما در همین دایرکتوری ایجاد شده‌است. می‌توانیم با دستور `cat public_key.asc` متن کلید عمومی را مشاهده کنیم:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
xsBNBGZfBcgBCAC2TWX1Mvj7326WDVpVXSqL9rFakIWSCom/S+CHhHz7JyU9i2Xo
Md766tXLLPv9u4lVt6KiJTXMsOSZLagxmzaT3sq9SRlDfinw4tP9rAUMrQ3Km1FX
EOtJEeGniGsR2syXGH+PfGO8gwWVYBP4F4U7Ctfx2PAqfbTq4trWRutqQUUE9xNi
+RQVBHF0aHpp0J6AbPLsNh7C42wXavk6z3tsLKrGe+dUEXrQDtv/bxDKaeDwQ5E5
hO8jABqkTTWVoKEgVVi8Y8Or2FNq3LTkb8quTVfgI7MWqjgc8GOu/HE2p7bWsd2e
l2TvtYTb9GlNBy6h/O5nSA2OL8XBUt5TcZhJABEBAAHNKk1laHJhZCBNaWxhbmxv
byA8bWlsYW5sb29tZWhyYWRAZ21haWwuY29tPsLAggQTAQgALAUCZl8FyAIbDgIL
CQIVCAIWAgIeARYhBKS5Xgrj6lrVEIQFsR/xV7LteUIxAAoJEB/xV7LteUIx5qcH
```

```
/RihmT7X5d+Plp5O+EX7gpKRm3D94tJPHZu5JqH9cfq6OTJH6RBL8etPNfAwmfqU
YN1XyhsjU+O/ORCqYKMLams2v7cD2zu6WS9A/ZpKwhTDHqpQZfMYhw+JiNWw/1z/
UcwiueLvmPr9GqgjFQyPBW0WHgbT4GSPstB/lsKk2IHkGaLjXYLsdrPaRB4LGV86
Kz/PnbUePXxii6AHOe6MyZMzhSUL+QXPVm+UR3OcxCE65gY6E+x2XT0Zif9BQnFQ
MB3uR+jtRjOZn+DAJ0GGCgV2CAWiiHqd+xwXIWAcl2KlCqIZmnKgdC2iNmvQJ+gG
1SdFIey8sKWg2Ez9WM8+/3o=
=zBSG
-----END PGP PUBLIC KEY BLOCK-----
```

(ب)

با استفاده از دستور `pgpdump public_key.asc` اطلاعات کلید را چاپ می‌کنیم:



```
2024-06-04 14:28:00  Milan in ~/University/Courses/402-2/Security/HW4
→ pgpdump public_key.asc
New: Public Key Packet(tag 6)(269 bytes)
        Ver 4 - new
        Public key creation time - Tue Jun  4 14:17:12 CEST 2024
        Pub alg - RSA Encrypt or Sign(pub 1)
        RSA n(2048 bits) - ...
        RSA e(17 bits) - ...
New: User ID Packet(tag 13)(42 bytes)
        User ID - Mehrad Milanloo <milanloomehrad@gmail.com>
New: Signature Packet(tag 2)(322 bytes)
        Ver 4 - new
        Sig type - Positive certification of a User ID and Public Key packet(0x13).
        Pub alg - RSA Encrypt or Sign(pub 1)
        Hash alg - SHA256(hash 8)
        Hashed Sub: signature creation time(sub 2)(4 bytes)
                Time - Tue Jun  4 14:17:12 CEST 2024
        Hashed Sub: key flags(sub 27)(1 bytes)
                Flag - This key may be used to sign data
                Flag - This key may be used to encrypt communications
                Flag - This key may be used to encrypt storage
        Hashed Sub: preferred symmetric algorithms(sub 11)(1 bytes)
                Sym alg - AES with 256-bit key(sym 9)
        Hashed Sub: preferred hash algorithms(sub 21)(1 bytes)
                Hash alg - SHA256(hash 8)
        Hashed Sub: preferred compression algorithms(sub 22)(1 bytes)
                Comp alg - ZLIB <RFC1950>(comp 2)
        Hashed Sub: features(sub 30)(1 bytes)
                Flag - Modification detection (packets 18 and 19)
        Hashed Sub: issuer fingerprint(sub 33)(21 bytes)
         v4 -   Fingerprint - a4 b9 5e 0a e3 ea 5a d5 10 84 05 b1 1f f1 57 b2 ed 79 42 31
        Sub: issuer key ID(sub 16)(8 bytes)
                Key ID - 0x1FF157B2ED794231
        Hash left 2 bytes - e6 a7
        RSA m^d mod n(2045 bits) - ...
                -> PKCS-1
```

- **شناسه‌ی کلید عمومی:**

```
        Key ID - 0x1FF157B2ED794231
```

- **الگوریتم‌های استفاده‌شده برای رمزگذاری، امضا و چکیده:** همان‌طور که در تصویر بالا مشاهده می‌شود، از الگوریتم RSA برای رمزگذاری کلید عمومی و امضا استفاده می‌شود. همچنین از SHA256 برای چکیده استفاده شده‌است.

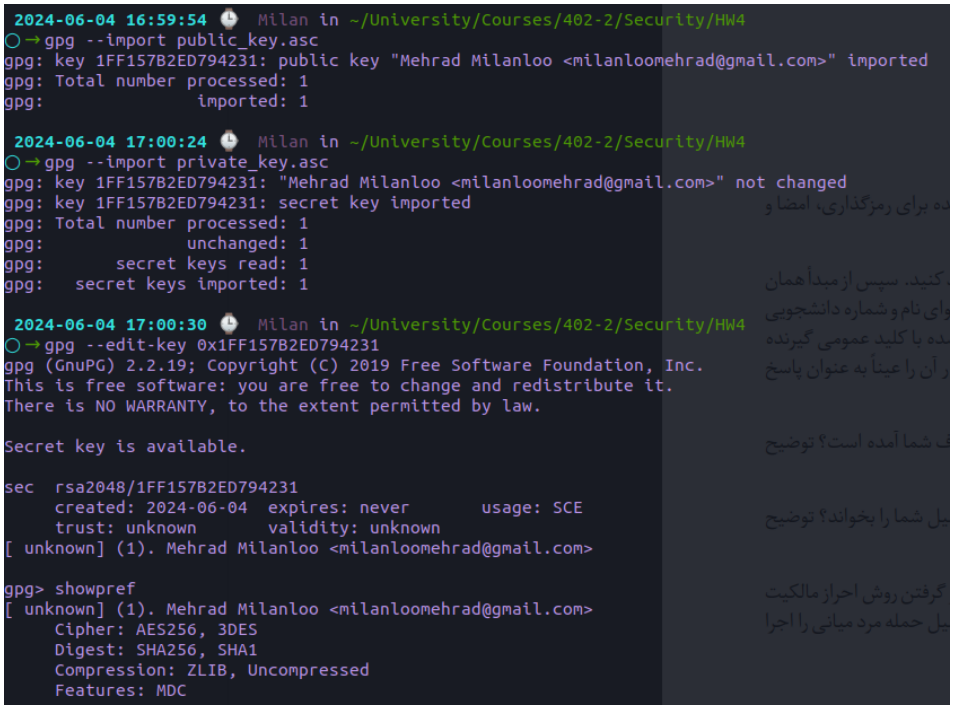
```
        Pub alg - RSA Encrypt or Sign(pub 1)
      Hash alg - SHA256(hash 8)
```

همچنین با وارد کردن دستور زیر می‌توانیم الگوریتم‌های دیگری که در رمزگذاری کلید، امضا و چکیده استفاده می‌توانند بشوند را ببینیم:

```
gpg --import public_key.asc
gpg --import private_key.asc
gpg --edit-key 0x1FF157B2ED794231
showpref
```

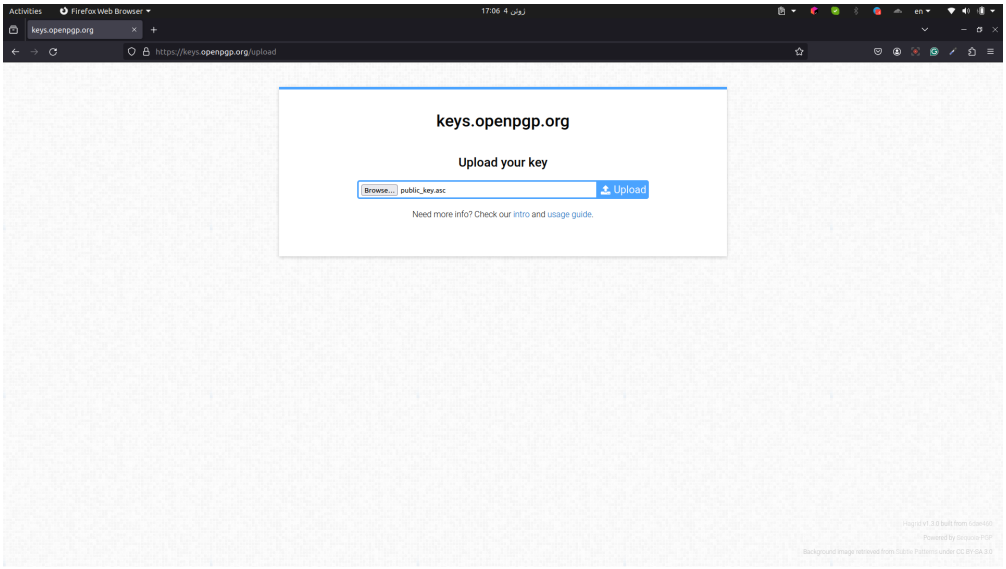
خروجی:

```
[ unknown] (1). Mehrad Milanloo <milanloomehrad@gmail.com>
     Cipher: AES256, 3DES
     Digest: SHA256, SHA1
     Compression: ZLIB, Uncompressed
     Features: MDC
```

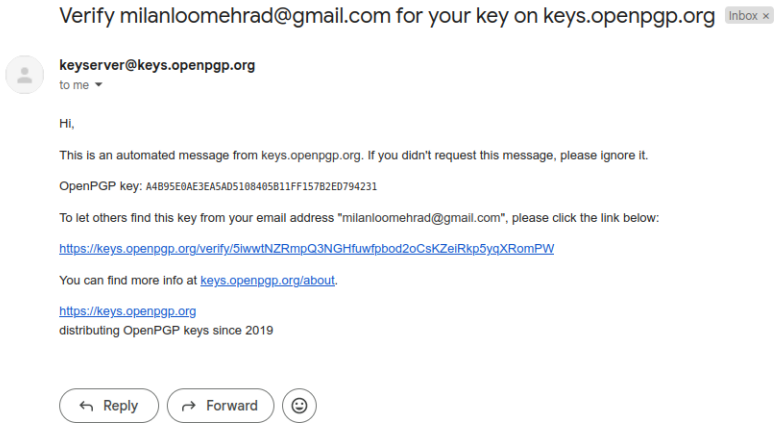


(پ)

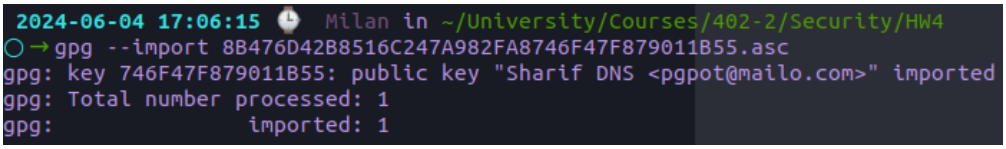
ابتدا وارد آدرس `https://keys.openpgp.org/upload` می‌شویم و کلید عمومی خود را آپلود می‌کنیم:



سپس در صفحه‌ای که به آن ریدایرکت شده‌ایم، گزینه‌ی تایید ایمیل را انتخاب می‌کنیم. سپس ایمیلی حاوی لینک تایید ارسال می‌شود که به راحتی با کلیک روی آن می‌توانیم ایمیل خود را تایید کنیم.



اکنون با وارد کردن ایمیل pgpot@mailo.com، کلید عمومی آن را دریافت می‌کنیم.
سپس با استفاده از دستوراتی که در تصویر زیر می‌توان مشاهده کرد، متن ایمیل خواسته‌شده را ایجاد می‌کنیم.



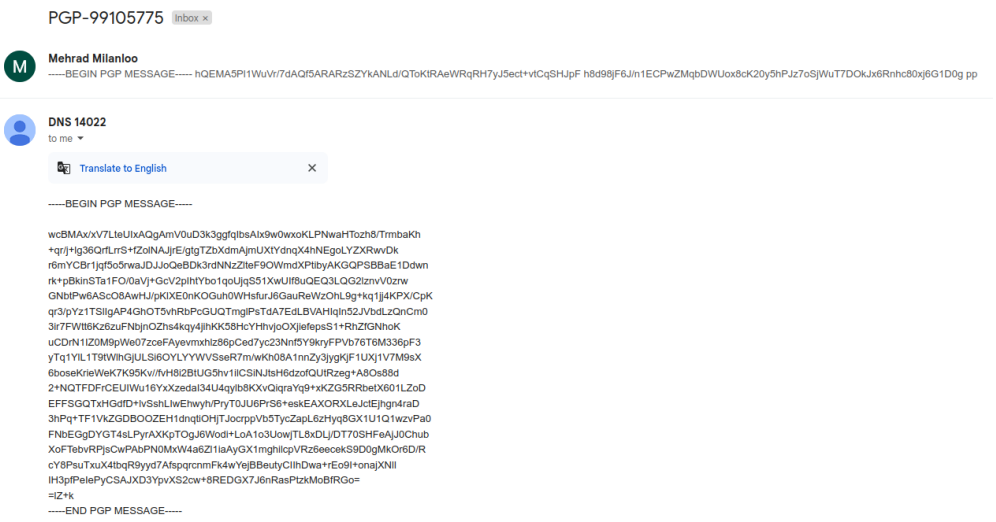

```
echo "Mehrad Milanloo - 99105775" > message.txt
```

```
2024-06-04 17:20:04  Milan in ~/University/Courses/402-2/Security/HW4
○ → gpg --recipient pgpot@mailo.com --encrypt --sign --armor --output email_message.asc message.txt
gpg: 93E5D56B95AFFEDD: There is no assurance this key belongs to the named user

sub  rsa2048/93E5D56B95AFFEDD 2024-05-18 Sharif DNS <pgpot@mailo.com>
 Primary key fingerprint: 8B47 6D42 B851 6C24 7A98  2FA8 746F 47F8 7901 1B55
     Subkey fingerprint: 8DC5 661A D348 60AD 4005  3D7A 93E5 D56B 95AF FEDD

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
```

در نهایت پیام رمزشده این‌گونه است:

```
-----BEGIN PGP MESSAGE-----
hQEMA5Pl1WuVr/7dAQf5ARARzSZYkANLd/QToKtRAeWRqRH7yJ5ect+vtCqSHJpF
h8d98jF6J/n1ECPwZMqbDWUox8cK20y5hPJz7oSjWuT7DOkJx6Rnhc80xj6G1D0g
ppxhnFP8UI627VQ+3KLSUYqxK8Zw4vXSnyYyYVWOScIRliDqHF4HiG2xFQnfxVXi
QkOAyBn6huxv4LgxxQSm4ldnvSSzvagPpI7WkUB7U6J9i4xaKZDPE60oz3D4mVYl
XI/Cai7KyCI2vnixf93n3eqvj9X5mHlnvO3ZoUgi8858UT2hgzxeBif0hwG1nGxl
R8eYBntWWMSkWBklr7FYIkiN7zz4zqcmHIWDkhJV89LA5wFsrM6mbrXJoKALmdRY
stOB72Ww8aqHMy2uvreQdDBPU16g7FzHGKu++MnWL1CSnvuC2rVIQbHk+g/oBKqT
w2IfnNoolwgFwfuuAljX4TNaeMfV+cSXkVAWIuDv3lcSFIbUcbzfww8xu4bJTnv7
H+tbc5RQnZnlMtZU8SH/pQKImjNDwa/CwJ7BIscPY+A+J108XZ5TiZsWBPNnzp00
GL7tiVnFcK8r4HrW6u+PpsCC7+1/P1zdkH+pNn769Xdt8XroVWjZHJB+2icDY1JJ
pQ/nr7iORHcLhqoA4ek46EbMljV3FATUj3rYAEOpyPJVB35gAGaovSqqsGHn0N/o
5OVDR8vKhjjWEyZKv93qjEDqws5PB2dPKWfMFwpoH+992ZeOZx145kUoTNr1kXoL
/+gUixNxqVAeyxiDjrvKdILAj8UFd/LM7hhppM3ed5RkOG7dQSACeqYyLS/pwsZJ
hU1H2hZX3Z3AIVlkdtMwzDcLVzKrc5VXUQz+VKvFTLaNxu58PfwusnSCzfHRFQro
1FWFLaPPoMIkGPcPrAgJn5I6tCXAnu/gMQ==
=kJFV
-----END PGP MESSAGE-----
```

این پیام را به آدرس ایمیل داده شده ارسال می‌کنیم و پاسخ ایمیل چنین است:



اکنون با استفاده از دستورات زیر، محتوای ایمیل را که در فایل `response.asc` قرار دادیم رمزگشایی می‌کنیم:

خروجی این‌گونه است:

```
Accepted from:Mehrad Milanloo <milanloomehrad@gmail.com>
body:b'Mehrad Milanloo - 99105775\n'
time:1717514823
token:IM6HA9-WD9NDR-R6SLA4-9RRBIP-OGQTVF-UM545C-G6S5E0-EUXI47
```

(ت)

بله. از آن‌جایی که در پروتکل PGP با استفاده از کلید خصوصی خودمان ایمیل را امضای دیجیتال می‌کنیم، دریافت‌کننده با استفاده از کلید عمومی ما می‌تواند صحت امضا را بررسی کند. از آن‌جایی که انتظار می‌رود کلید خصوصی تنها در اختیار خود ما باشد و کلید عمومی تنها می‌تواند صحت امضایی را تایید کند که با آن کلید خصوصی امضا شده‌است، پس دریافت‌کننده مطمئن می‌شود که این ایمیل از سمت ما بوده‌است.

(ث)

بله. از آن‌جایی که ایمیل با استفاده از کلید عمومی دریافت‌کننده رمز شده‌است و کلید خصوصی متناظر با آن تنها در اختیار دریافت‌کننده‌ی مد نظر است و این رمز تنها با همان کلید خصوصی باز می‌شود، می‌توانیم مطمئن باشیم که فقط دریافت‌کننده‌ی مد نظر ما می‌تواند به محتوای ایمیل دسترسی داشته‌باشد.

(ج)

بله. وقتی یک کلید در سرور کلید عمومی آپلود می‌شود، باید اطمینان حاصل شود که کلید واقعا متعلق به مالک ایمیل مرتبط است. این کار از طریق ارسال ایمیل تایید به صاحب ایمیل برای تایید مالکیت انجام می‌شود. اگر مهاجم به سرور ایمیل دسترسی داشته باشد، می‌تواند ایمیل تایید ارسالی از طرف سرور کلید عمومی را رهگیری کند. حمله را می‌تواند به این شکل انجام دهد که به جای مالک اصلی، به ایمیل تایید پاسخ دهد و کلید عمومی خود را به سرور کلید عمومی مرتبط با ایمیل فرد مورد تهاجم آپلود کند. در نتیجه، افراد دیگر که کلید عمومی را از سرور کلید عمومی دانلود می‌کنند، به اشتباه کلید مهاجم را به جای کلید اصلی قربانی دریافت می‌کنند. وقتی آن‌ها پیام‌های رمزگذاری‌شده ارسال می‌کنند، پیام‌ها با کلید عمومی مهاجم رمزگذاری می‌شوند و مهاجم می‌تواند آن‌ها را رمزگشایی کرده، خوانده و دستکاری کند.

SSL/TLS ۲

(آ)

- نقش SSL/TLS در امن کردن ارتباطات وب:

پروتکل **SSL (Secure Sockets Layer)** و پروتکل **TLS (Transport Layer Security)** پروتکل‌هایی هستند که برای امن کردن ارتباطات در بستر اینترنت طراحی شده‌اند. این پروتکل‌ها به صورت گسترده برای محافظت از داده‌هایی که بین مرورگر وب و سرور رد و بدل می‌شوند، استفاده می‌شوند. در ادامه به توضیح ویژگی‌ها و مکانیزم‌های کلیدی این پروتکل‌ها می‌پردازیم:

- ویژگی‌های کلیدی SSL/TLS:

  - **رمزنگاری داده‌ها**: SSL/TLS داده‌های تبادل شده بین کلاینت (مثلاً مرورگر وب) و سرور را رمزنگاری می‌کند. این رمزنگاری مانع از خوانده شدن و دستکاری اطلاعات توسط افراد در میانه‌ی راه می‌شود.
  - **احراز هویت**: SSL/TLS با استفاده از گواهینامه‌های دیجیتال اطمینان می‌دهد که کلاینت با یک سرور معتبر در ارتباط است و نه با یک سرور جعلی. گواهینامه‌ها توسط مراجع صدور گواهینامه (CA) تأیید می‌شوند.
  - **یکپارچگی داده‌ها**: SSL/TLS تضمین می‌کند که داده‌ها در حین انتقال تغییر نکرده‌اند. این کار با استفاده از کدهای تأیید صحت پیام (MAC) انجام می‌شود.
  - **تبادل کلید امن**: SSL/TLS از مکانیزم‌های تبادل کلید (مانند Diffie-Hellman) برای تبادل کلیدهای رمزنگاری به صورت امن استفاده می‌کند.

- مکانیزم‌های کلیدی SSL/TLS:

این پروتکل‌ها از طریق فرایند Handshaking ارتباط امنی را بین کلاینت و سرور برقرا می‌کنند.
مراحل فرایند Handshaking:

  - **پیام ClientHello** : کلاینت (مثلاً مرورگر وب) یک پیام «Hello» به سرور ارسال می‌کند که شامل نسخه‌های TLS پشتیبانی شده، مجموعه‌های رمزنگاری (Cipher Suites) و یک رشته بایت تصادفی به نام Client Random است.
  - **پیام ServerHello** : سرور به پیام «Hello» کلاینت پاسخ می‌دهد که شامل نسخه انتخابی TLS، مجموعه رمزنگاری انتخابی و یک رشته بایت تصادفی به نام Server Random است. همچنین سرور گواهینامه دیجیتال خود را برای احراز هویت ارسال می‌کند.
  - **احراز هویت**: کلاینت گواهینامه دیجیتال سرور را با استفاده از کلید عمومی صادر شده توسط مرجع صدور گواهینامه (CA) بررسی می‌کند. این فرآیند تضمین می‌کند که سرور همان چیزی است که ادعا می‌کند.
  - **تولید کلید Premaster** : کلاینت یک رشته بایت تصادفی دیگر به نام Premaster Secret تولید می‌کند و آن را با استفاده از کلید عمومی سرور رمزگذاری می‌کند و به سرور ارسال می‌کند.
  - **رمزگشایی کلید Premaster** سرور با استفاده از کلید خصوصی خود، Premaster Secret را رمزگشایی می‌کند.
  - **تولید کلیدهای جلسه (Session Keys)** : کلاینت و سرور هر دو از Client Random، Server Random و Premaster Secret برای تولید کلیدهای جلسه استفاده می‌کنند که برای رمزگذاری متقارن استفاده خواهند شد.
  - **پیام‌های پایانی**: کلاینت یک پیام «Finished» ارسال می‌کند که با کلید جلسه رمزگذاری شده است و سرور نیز با ارسال پیام «Finished» خود که با کلید جلسه رمزگذاری شده است، پاسخ می‌دهد.

از این نقطه به بعد، ارتباط بین کلاینت و سرور با استفاده از کلیدهای جلسه به صورت متقارن رمزگذاری می‌شود.

(ب)

- **نحوه بهره‌برداری مهاجم از ضعف‌های SSL/TLS**

مهاجمان می‌توانند با استفاده از ضعف‌ها و آسیب‌پذیری‌های موجود در پیاده‌سازی‌های این پروتکل‌ها و یا با بهره‌برداری از تنظیمات نادرست، امنیت ارتباطات را تهدید کنند. برخی از این حملات عبارتند از:

- **حملات مرد میانی (MITM)** در این نوع حمله، مهاجم بین کلاینت و سرور قرار می‌گیرد و ارتباطات را رهگیری و حتی دستکاری می‌کند. اگر مهاجم بتواند از گواهینامه‌های جعلی استفاده کند یا گواهینامه‌های اعتبارسنجی نشده را به کار ببرد، می‌تواند خود را به عنوان سرور واقعی جا بزند.
- **حملات داونگرید Downgrade Attacks** مهاجم می‌تواند ارتباط را مجبور به استفاده از نسخه‌های قدیمی‌تر و آسیب‌پذیرتر پروتکل‌های SSL/TLS کند. برای مثال، استفاده از SSL 2.0 یا SSL 3.0 که دارای ضعف‌های امنیتی شناخته‌شده هستند.
- **حملات مبتنی بر آسیب‌پذیری‌های پیاده‌سازی** پیاده‌سازی‌های مختلف SSL/TLS ممکن است دارای باگ‌ها و آسیب‌پذیری‌هایی باشند که مهاجمان می‌توانند از آنها بهره‌برداری کنند. برای مثال، آسیب‌پذیری Heartbleed در OpenSSL که امکان دسترسی به اطلاعات حساس سرور را فراهم می‌کرد. از حملات دیگر می‌توان به BEAST و POODLE اشاره کرد که از ضعف‌های موجود در این پروتکل‌ها برای رسیدن به اطلاعات حساس استفاده می‌کنند.

(پ)

- **اهمیت تایید گواهی‌نامه وب‌سایت**

تایید گواهی‌نامه وب‌سایت، که به آن احراز هویت گواهی‌نامه نیز گفته می‌شود، فرایندی است که برای تایید اعتبار مالکیت وب‌سایت و هویت سرور وب انجام می‌شود. این کار با استفاده از گواهی‌نامه‌های امنیتی دیجیتال انجام می‌شود که توسط یک مرجع صدور گواهی (CA) معتبر صادر شده‌اند. از مزایای تایید گواهینامه‌ی وب‌سایت می‌توان به موارد زیر اشاره کرد:

- **افزایش امنیت**: تایید گواهی‌نامه به ایجاد یک اتصال امن بین وب‌سایت و مرورگر کاربر کمک می‌کند. این امر از رهگیری و سرقت اطلاعات حساس مانند اطلاعات کارت اعتباری یا اطلاعات ورود به سیستم جلوگیری می‌کند.
- **افزایش اعتماد**: وب‌سایت‌هایی که دارای گواهی‌نامه تایید شده هستند، معتبرتر و قابل اعتمادتر به نظر می‌رسند. این امر می‌تواند منجر به افزایش نرخ تبدیل و وفاداری مشتری شود.
- **بهبود سئو**: موتورهای جستجو مانند گوگل به وب‌سایت‌هایی که دارای گواهی‌نامه تأیید شده هستند، رتبه بالاتری می‌دهند.

- **نحوه‌ی استفاده از SSLLab برای تجزیه و تحلیل پیکربندی SSL/TLS**

SSLLab ک ابزار آنلاین رایگان است که برای تجزیه و تحلیل پیکربندی SSL/TLS سرورهای وب استفاده می‌شود. این ابزار می‌تواند برای بررسی موارد زیر استفاده شود:

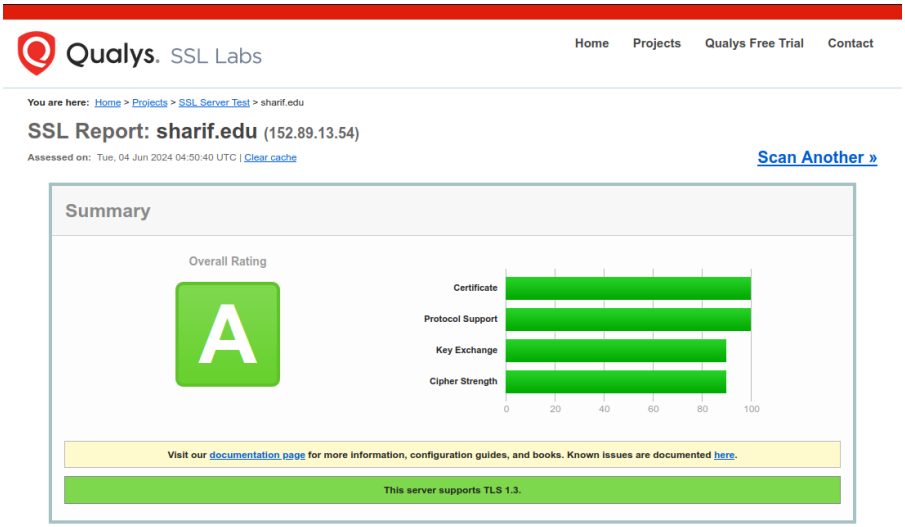
- **نوع گواهینامه**: SSLLab می‌تواند نوع گواهی‌نامه SSL/TLS که توسط یک سرور وب استفاده می‌شود را تعیین کند، مانند DV ، OV یا EV .
- **اعتبار گواهینامه**: SSLLab می‌تواند اعتبار گواهی‌نامه SSL/TLS را بررسی کند و اینکه آیا توسط یک CA معتبر صادر شده است یا خیر.

- **قدرت رمزگذاری**: می‌تواند قدرت رمزگذاری پروتکل SSL/TLS را که توسط یک سرور وب استفاده می‌شود، تعیین کند.
- **وجود آسیب‌پذیری**: می‌تواند سرورهای وب را برای وجود آسیب‌پذیری‌های SSL/TLS مانند Heartbleed و Poodle اسکن کند.

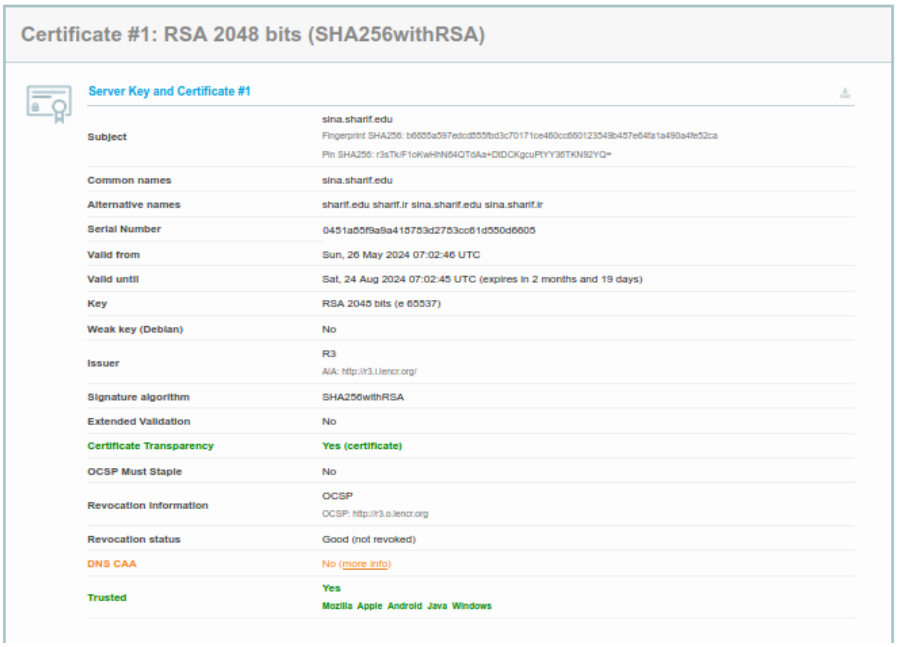
- **استفاده از SSLLab برای اسکن سایت دانشگاه و سایت aibuz**

برای استفاده از SSLLab کافی است نام دامنه‌ی وب‌سایتی را که می‌خواهیم تجزیه و تحلیل کنیم، وارد کنیم. SSLLab سپس یک گزارش دقیق از پیکربندی SSL/TLS وب‌سایت ارائه می‌دهد.

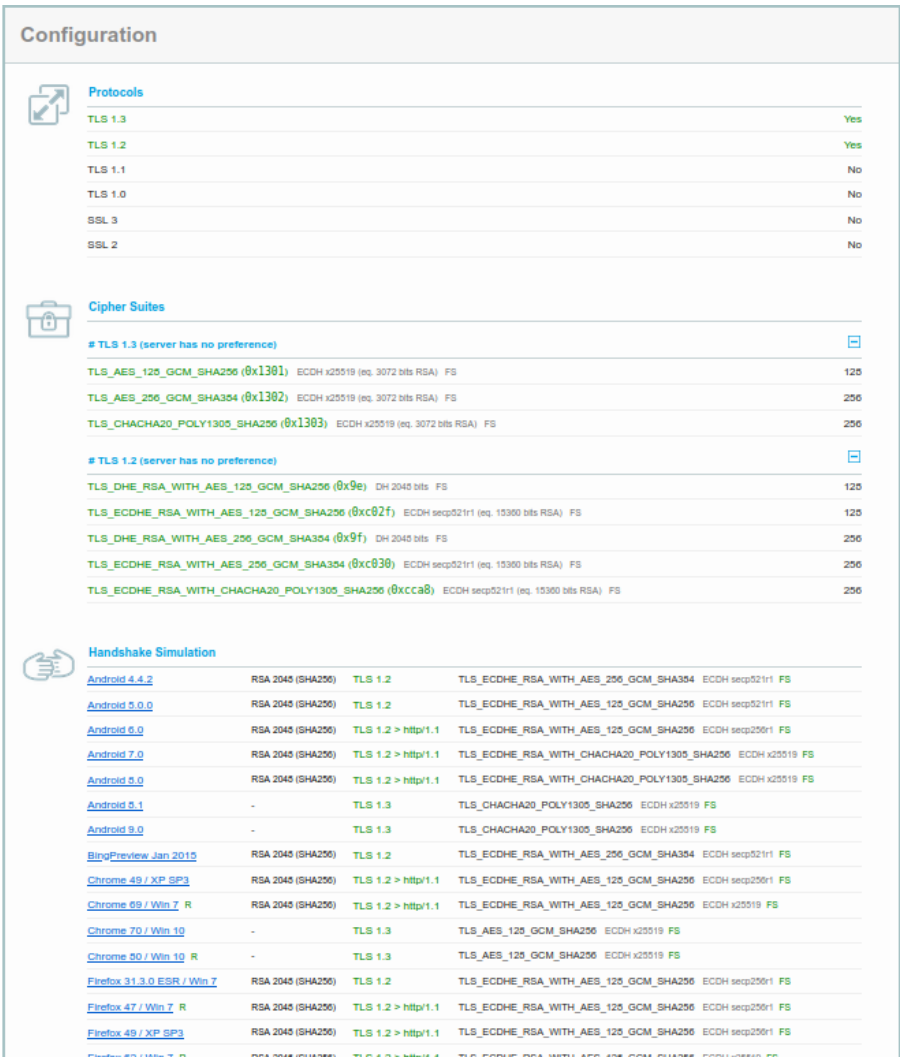
- `sharif.edu` خلاصه‌ای از نمره‌دهی و گزارش وضعیت گواهی و پیکربندی SSL/TLS سایت را می‌توان در ابتدا مشاهده کرد:



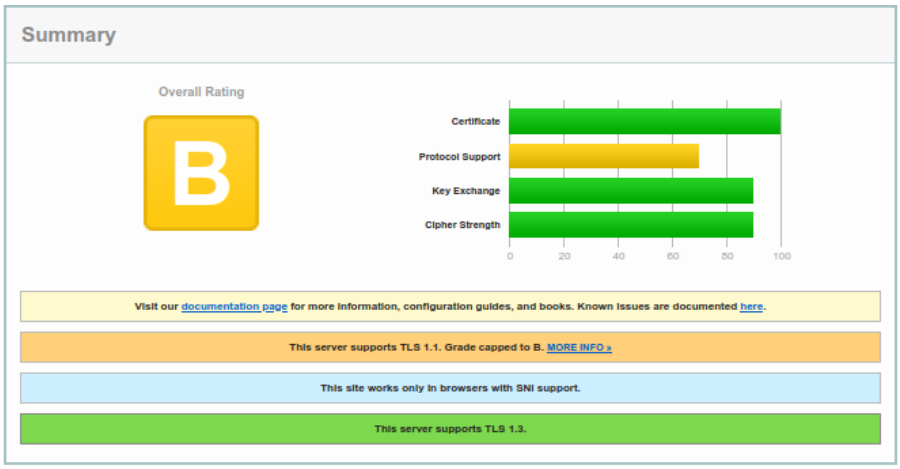
در ادامه، می‌توان گزارش دقیق‌تر از هر یک از پارامترهایی که در خلاصه‌ی گزارش نمره‌دهی داده‌شده‌اند را مشاهده کرد. برای مثال در بخش Certificate، اطلاعاتی نظیر نوع گواهینامه، تاریخ صدور، الگوریتم‌های استفاده شده در امضا و صادرکننده‌ی گواهینامه آمده‌است.



در بخش Configuration اطلاعاتی از نظیر نسخه‌های SSL/TLS که وب‌سایت پشتیبانی می‌کند و وضعیت Handshaking در مرورگرهای مختلف آمده‌است.



– `panel.aibuz.net` خلاصه‌ای از نمره‌دهی و گزارش وضعیت گواهی و پیکربندی SSL/TLS سایت را می‌توان در ابتدا مشاهده کرد:

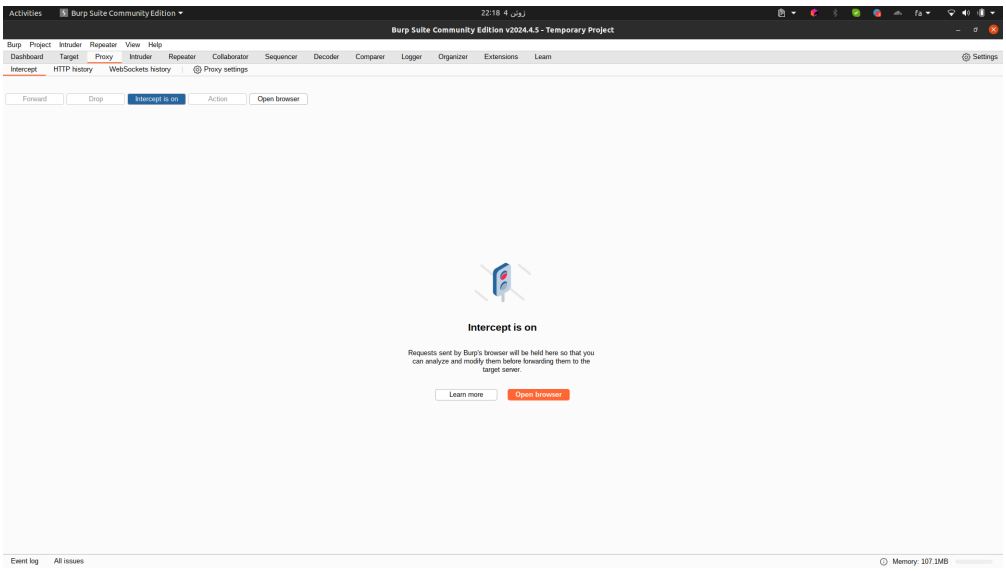


همان‌طور که در تصویر مشخص است، این سایت نمره‌ی پایین‌تری گرفته است و مهم‌ترین دلیل آن پشتیبانی از نسخه‌های قدیمی

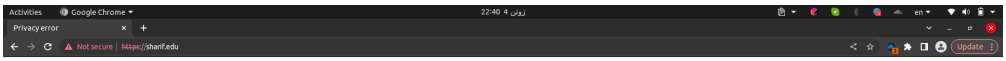
TLS است که آسیب‌پذیری‌های شناخته‌شده‌ای دارند.

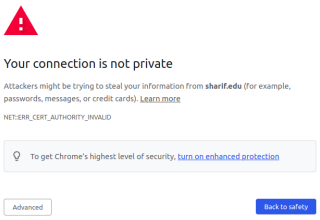
(ت)

پس از نصب BurpSuite، یک پروژه‌ی جدید ساخته و وارد تب Proxy می‌شویم و Intercept را روشن می‌کنیم.

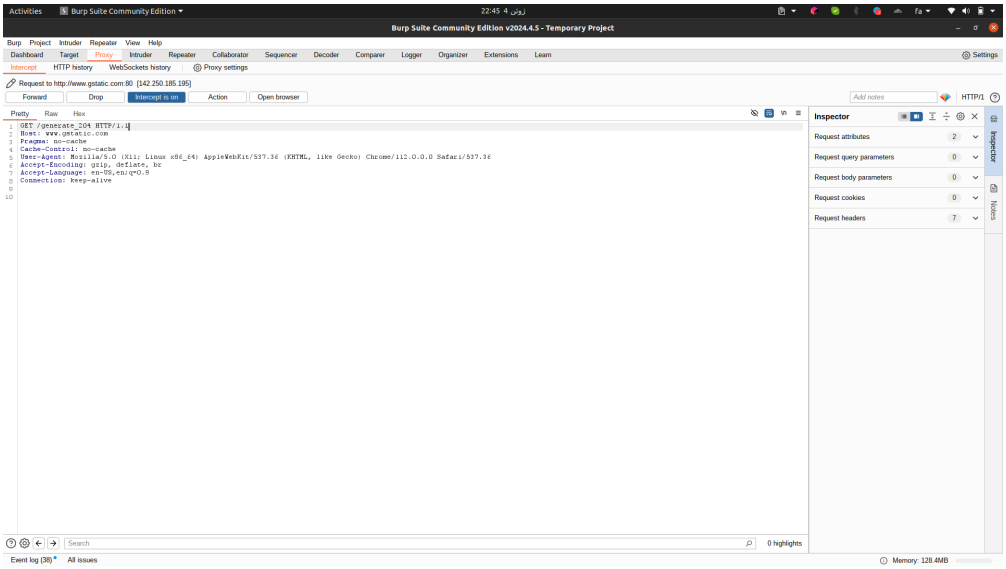


اکنون تنظیمات پراکسی دستگاه را روی پورت دیفالت ۸۰۸۰ فعال می‌کنیم و در مرورگر آدرس یک سایت دلخواه مانند `sharif.edu` را جست‌وجو می‌کنیم.





همان‌طور که انتظار می‌رفت به اخطار سرتیفیکیت مرورگر بر می‌خوریم. زیرا Burp در میانه‌ی راه مرورگر و سرور قرار دارد و می‌تواند درخواست‌ها را مشاهده و یا حتی تغییر دهد.



در شکل می‌بینید که به راحتی می‌توان تمامی درخواست‌ها را در BurpSuite مشاهده کرد. سپس به‌ازای هر درخواست می‌توان تغییرات دلخواه را در آن ایجاد کرد و درخواست جدید را forward کرد. همچنین می‌توان بسته‌ها را drop کرد و در رسیدن آن‌ها به مقصد اختلال ایجاد کرد.

۳ دیوار آتش

(آ)

دقت کنید که در این پروتکل سرور روی پورت ۲۵ پیام‌ها را دریافت و ارسال کرده و کلاینت روی پورتی با مقدار بزرگ‌تر از ۱۰۲۳ پیام‌های این پروتکل را ارسال و دریافت می‌کند. بنابراین می‌توانیم قوانین داده‌شده را این‌گونه تفسیر کنیم:

- **آ**: این قانون اجازه می‌دهد یک کلاینت خارج از شبکه به سروری درون شبکه روی پورت ۲۵ (روی پروتکل TCP) وصل شود. بنابراین این قانون اجازه‌ی دریافت پیام‌های SMTP را برای سرورهای درون شبکه صادر می‌کند.
- **ب**: این قانون اجازه می‌دهد یک سرور درون شبکه به کلاینتی خارج از شبکه روی پورتی بزرگ‌تر از ۱۰۲۳ (روی پروتکل TCP) وصل شود. بنابراین این قانون اجازه‌ی ارسال پیام‌های SMTP را برای سرورهای درون شبکه صادر می‌کند.
- **ج**: این قانون اجازه می‌دهد یک کلاینت درون شبکه به سر خارج از شبکه روی پورت ۲۵ (روی پروتکل TCP) وصل شود. بنابراین این قانون اجازه‌ی ارسال پیام‌های SMTP را برای کلاینت‌های درون شبکه صادر می‌کند.
- **د**: این قانون اجازه می‌دهد یک سرور خارج از شبکه به کلاینتی درون شبکه روی پورتی بزرگ‌تر از ۱۰۲۳ (روی پروتکل TCP) وصل شود. بنابراین این قانون اجازه‌ی دریافت پیام‌های SMTP را برای کلاینت‌های درون شبکه صادر می‌کند.
- **ه**: این قانون مانع رد شدن هرگونه پیام دیگری از این شبکه می‌شود.

(ب)

بله. اگر سرویسی که ارائه می‌شود روی پروتکل TCP کار کند، فرد می‌تواند طبق قانون (د) به پراکسی وصل‌شده و پاسخ‌های خود را می‌تواند طبق قانون (ب) دریافت کند.

(پ)

در لینک زیر می‌توانید ویدیوی توضیحات کامل و تست عملکرد این سوال را مشاهده کنید:

- Link to Video

کد زیر، کدی است که در ویدیو توضیحات کامل آن داده شده‌است.

```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP

iptables -A INPUT -p tcp --dport 25 -s 192.168.56.1 -d 192.168.56.104 -j ACCEPT

iptables -A INPUT -p tcp --dport 1024:65535 -s 192.168.56.1 -d 192.168.56.104 -j ACCEPT

iptables -A OUTPUT -p tcp --dport 1024:65535 -s 192.168.56.104 -d 192.168.56.1 -j ACCEPT

iptables -A OUTPUT -p tcp --dport 25 -s 192.168.56.104 -d 192.168.56.1 -j ACCEPT
```

دقت کنید که می‌توانستیم به‌راحتی IP ها را با رنج آیپی زیرشبکه جابجا کنیم و قواعد به این شکل می‌بودند:

```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP

iptables -A INPUT -p tcp --dport 25 ! -s 192.168.56.0/24 -d 192.168.56.0/24 -j ACCEPT

iptables -A INPUT -p tcp --dport 1024:65535 ! -s 192.168.56.0/24 -d 192.168.56.0/24 -j ACCEPT

iptables -A OUTPUT -p tcp --dport 1024:65535 -s 192.168.56.0/24 ! -d 192.168.56.0/24 -j
ACCEPT

iptables -A OUTPUT -p tcp --dport 25 -s 192.168.56.0/24 ! -d 192.168.56.0/24 -j ACCEPT
```