

به نام خدا



دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر

امنیت داده و شبکه

تمرین ۴

مهراد میلانلو

۹۹۱۰۵۷۷۵

Pretty Good Problem ۱

(۱)

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

```

1  from pgpy import PGPKey, PGPUID
2  from pgpy.constants import PubKeyAlgorithm, KeyFlags, HashAlgorithm \
3  , SymmetricKeyAlgorithm, CompressionAlgorithm
4
5  NAME = "Mehrad Milanloo"
6  EMAIL = "milanloomehrad@gmail.com"
7
8  # Generate a new key pair
9  key = PGPKey.new(PubKeyAlgorithm.RSAEncryptOrSign, 2048)
10 uid = PGPUID.new(NAME, email=EMAIL)
11
12 # Add the UID to the key
13 key.add_uid(uid, usage={KeyFlags.Sign
14                        , KeyFlags.EncryptCommunications
15                        , KeyFlags.EncryptStorage}
16                        , hashes=[HashAlgorithm.SHA256]
17                        , ciphers=[SymmetricKeyAlgorithm.AES256]
18                        , compression=[CompressionAlgorithm.ZLIB])
19
20 # Export the public key
21 with open("public_key.asc", "w") as f:
22     f.write(str(key.pubkey))
23
24 # Export the private key
25 with open("private_key.asc", "w") as f:
26     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:

```

1  -----BEGIN PGP PUBLIC KEY BLOCK-----
2
3  xsBNBGZfBcgBCAC2TWX1Mvj7326WDVpVXSqL9rFakIWSCom/S+CHhHz7JyU9i2Xo
4  Md766tXLLPv9u41Vt6KiJTXMs0SZLagxmzaT3sq9SR1Dfinw4tP9rAUMrQ3Km1FX
5  E0tJEeGniGsR2syXGH+PfG08gwWVYBP4F4U7Ctfx2PAqfbTq4trWRutqQUUE9xNi
6  +RQVBHF0aHpp0J6AbPLsNh7C42wXavk6z3tsLKrGe+dUEXrQDtv/bxDKaeDwQ5E5
7  h08jABqkTTWVoKEgVV18Y80r2FNq3LTkb8quTVfGI7MWqjgc8G0u/HE2p7bWsd2e
8  12TvtYtb9G1NBy6h/05nSA20L8XBUt5TcZhJABEBAAHNKk1laHJhZCBNaWxhbmVx
9  byA8bWlsYW5sb29tZW50YWR5YWR5YWR5YWR5YWR5YWR5YWR5YWR5YWR5YWR5YWR5

```

```

10 /RihmT7X5d+Plp50+EX7gpKRm3D94tJPHZu5JqH9cfq60TJH6RBL8etPNfAwmfQ
11 YN1XyhsjU+0/ORCqYKMLams2v7cD2zu6WS9A/ZpKwhTDHqpQZfMYhw+JiNWw/1z/
12 UcwIueLvmPr9GqgjFQyPBWOWHgbT4GSPstB/lsKk2IHkGaLjXYLsdrPaRB4LGV86
13 Kz/PnbUePXxii6AH0e6MyZMzhSUL+QXPVm+UR30cxCE65gY6E+x2XT0Zif9BQnFQ
14 MB3uR+jtRj0Zn+DAJ0GGCgV2CAWiiHqd+xwXIWAcl2K1CqIZmnKgdC2iNmVQJ+gG
15 1SdFiey8sKWg2Ez9WM8+/3o=
16 =zBSG
17 -----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.

```

1 Pub alg - RSA Encrypt or Sign(pub 1)
2 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:

```

1 gpg --import public_key.asc
2 gpg --import private_key.asc
3 gpg --edit-key 0x1FF157B2ED794231
4 showpref

```

خروجی:

```

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

```

```

2024-06-04 16:59:54 Milan in ~/University/Courses/402-2/Security/HW4
➔ gpg --import public_key.asc
gpg: key 1FF157B2ED794231: public key "Mehrad Milanloo <milanloomehrad@gmail.com>" imported
gpg: Total number processed: 1
gpg: imported: 1

2024-06-04 17:00:24 Milan in ~/University/Courses/402-2/Security/HW4
➔ gpg --import private_key.asc
gpg: key 1FF157B2ED794231: "Mehrad Milanloo <milanloomehrad@gmail.com>" not changed
gpg: key 1FF157B2ED794231: secret key imported
gpg: Total number processed: 1
gpg: unchanged: 1
gpg: secret keys read: 1
gpg: secret keys imported: 1

2024-06-04 17:00:30 Milan in ~/University/Courses/402-2/Security/HW4
➔ gpg --edit-key 0x1FF157B2ED794231
gpg (GnuPG) 2.2.19; Copyright (C) 2019 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.

Secret key is available.

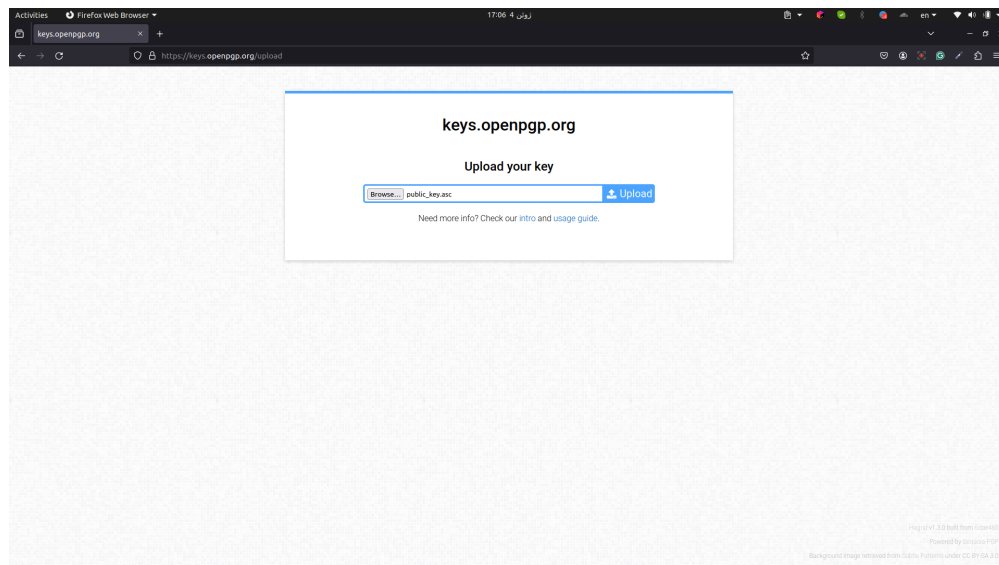
sec rsa2048/1FF157B2ED794231
created: 2024-06-04 expires: never usage: SCE
trust: unknown validity: unknown
[ unknown] (1). Mehrad Milanloo <milanloomehrad@gmail.com>

gpg> showpref
[ 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.

Verify milanloomehrad@gmail.com for your key on keys.openpgp.org Inbox x

 **keyserver@keys.openpgp.org**
to me ▾

Hi,

This is an automated message from keys.openpgp.org. If you didn't request this message, please ignore it.

OpenPGP key: A4B95E0AE3EASAD5108405B11FF15782ED794231

To let others find this key from your email address "milanloomehrad@gmail.com", please click the link below:

<https://keys.openpgp.org/verify/5iwwtNZRmpQ3NGHfufwfbod2oCsKZeIRkp5yqXRomPW>

You can find more info at keys.openpgp.org/about.

<https://keys.openpgp.org>

distributing OpenPGP keys since 2019

↩ Reply ➡ Forward 😊

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.

```
2024-06-04 17:06:15 Milan in ~/University/Courses/402-2/Security/HW4
➜ gpg --import 8B476D42B8516C247A982FA8746F47F879011B55.asc
gpg: key 746F47F879011B55: public key "Sharif DNS <pgpot@mailo.com>" imported
gpg: Total number processed: 1
gpg: imported: 1
```

```
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 رمز شده این گونه است:

```

1 -----BEGIN PGP MESSAGE-----
2 hQEMA5Pl1WuVr/7dAQf5ARARzSZYkANLd/QTOKtRAeWRqRH7yJ5ect+vtCqSHJpF
3 h8d98jF6J/n1ECPwZMqbDWUox8cK20y5hPjZ7oSjWuT7D0kJx6Rnhc80xj6G1D0g
4 ppxhnpFP8UI627VQ+3KLSUYqxK8Zw4vXSnyYyYVW0ScIRliDqHF4HiG2xFQnfVXi
5 Qk0AyBn6huxv4LgxxQSm4ldnvSSzvagPpI7WkUB7U6J9i4xaKZDPE60oz3D4mVYl
6 XI/Cai7KyCI2vnixf93n3eqvj9X5mHlnvO3ZoUgi8858UT2hgzzxBif0hwG1nGx1
7 R8eYBntWMSkWBklr7FYIkiN7zz4zqcmHIWDkhJV89LA5wFsrM6mbrXJoKALmdRY
8 st0B72Ww8aqHMy2uvreQdDBPU16g7FzHGKu++MnWL1CSnvuC2rVIQbHk+g/oBKqT
9 w2IfnNoolwgFwfuuAljX4TNaemfV+cSXkVAWIuDv3lcSFibUcbzfww8xu4bJTnv7
10 H+tbcs5RQnZn1MtZU8SH/pQKImjNDwa/CwJ7BIscPY+A+J108XZ5TiZsWBPnNzp00
11 GL7tiVnFcK8r4HrW6u+PpsCC7+1/P1zdkH+pNn769Xdt8XroVWjZJHJB+2icDY1JJ
12 pQ/nr7i0RHcLhqoA4ek46EbM1jV3FATUj3rYAE0pyPJVB35gAGaovSqqSGHn0N/o
13 50VDR8vKhjJWEyZKv93qjEDqws5PB2dPKWfMFwpoH+992Ze0Zx145kUoTNr1kXoL
14 /+gUixNqxPAeyxiDjrvKdILAj8UFD/LM7hhppM3ed5Rk0G7dQSACeqYyLS/pwsZJ
15 hU1H2hZX3Z3AIVlkdMwzDcLVzKrc5VXUQz+VKvFTLaNxu58PfwusnSCzfHRFQro
16 1FWFLaPPoMIkGPcPrAgJn5I6tCXAnu/gMQ==
17 =kJFV
18 -----END PGP MESSAGE-----

```

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

PGP-99105775 [Inbox x](#)



Mehrad Milanloo

-----BEGIN PGP MESSAGE----- hQEMA5Pl1WuVr/7dAQf5ARARzSZYkANLd/QTOKtRAeWRqRH7yJ5ect+vtCqSHJpF h8d98jF6J/n1ECPwZMqbDWUox8cK20y5hPjZ7oSjWuT7D0kJx6Rnhc80xj6G1D0g pp



DNS 14022

to me

[Translate to English](#)

-----BEGIN PGP MESSAGE-----

```

wcBMAv/v7LteUlxAQgAmVouD3k3ggfqlbsAix0w0xokLPNwaHTozh8TmbakH
*ggJh36QrLr5+2ZnNAJrEigqTZbXdmAlmUxTYdnqK4hNgeqLYZXrRwOk
r6mYCB1jgt5o5wauDJJoQeB0k3rdNzZtF9OWmdXPlibyAKGQPSB8aE1Ddwm
rk+pBkinStA1FO0aVjHgcV2phtYbo1qoUjgS51XwUfBuQE03LQG2lzmV0zrw
GNtBPw6ASc08AwHUpKXEOnKOGuhOWhtsfurJ6GauRwZ0hL8g+q1j4KPx/CpK
qr3pYz1TSlgAP4GH0T5vRbPcGUQTmgPpTsA7EdLbVAHlqin52JvbdLQnCM0
3lr7FWl6Kz6zuFNbjrOZhs4kzy4jhKK58HcyHrvjOxOJefepsS1+RhZGNhok
uCDn1I20M9pWe07zceFAyewmhz86pCad7yc23Nnt5Y9kyFPVb76T6M336pF3
yTq1Yl1T9lWtHglJULS6OYLYYVWVSeR7mWmKh08A1nnZy3yqKJF1UX1V7M9sX
8boseKrieWeK7K95KvifH82BILUG5nv1ICSINJtsH6dzQIurZeg+A8Os88d
2+NOTDFDFCEUIWu16YxXzedal34U4qyb8KXvOlqraYq9+XKZG5RRbetX601LZoD
EFFSGQTxHGdID+lvShLlwEhwyhPryT0JUEPrS6+eskEAXORXLeJclEjhgnd4raD
3hPq+TF1VikZGDBOOZEH1dnqtlOHJTJocrrpVb5TycZapL6zYq8GX1U1Q1wzVPa0
FnbEGgDYGT4sLpYrAXKpTogJ6Wodi+LoA1o3UowjTL8xDLJD705HFEAJUChub
XoFtebVRPjsCwPabPN0MxW4a6Z1laAyGX1mgihcpVRz6eecekS9D0gMkOr6D/R
cY8PauTxx4tbgR9yyd7AfsqprnmFk4wYejBBeutyCllhDwa+e09H+onajXNll
lH3pPelePyCSAJXD3YpX3Zcw+8REDGX7J6nRasPlzKMoBRGo=
=iZ*k
-----END PGP MESSAGE-----

```

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

```
2024-06-04 17:32:43 Milan in ~/University/Courses/402-2/Security/HW4
➜ gpg --recipient milanloomehrad@gmail.com --decrypt --armor --output response.txt response.asc
gpg: encrypted with 2048-bit RSA key, ID 1FF157B2ED794231, created 2024-06-04
"Meherad Milanloo <milanloomehrad@gmail.com>"
File 'response.txt' exists. Overwrite? (y/N) y
gpg: Signature made :۱۷, ۲۴, ۲۰۲۴
gpg: using RSA key 8B476D42B8516C247A982FA8746F47F879011B55
gpg: Good signature from "Sharif DNS <pgpot@mailo.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: 8B47 6D42 B851 6C24 7A98 2FA8 746F 47F8 7901 1B55

2024-06-04 17:33:28 Milan in ~/University/Courses/402-2/Security/HW4
➜ cat response.txt
Accepted from:Meherad Milanloo <milanloomehrad@gmail.com>
body:b'Mehrad Milanloo - 99105775\n'
time:1717514823
token:IM6HA9-WD9NDR-R6SLA4-9RRBIP-OGQTVF-UM545C-G6S5E0-EUXI47
```

خروجی این‌گونه است:

```
1 Accepted from:Meherad Milanloo <milanloomehrad@gmail.com>
2 body:b'Mehrad Milanloo - 99105775\n'
3 time:1717514823
4 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 رمزگذاری شده ارسال می‌کنند، پیام‌ها با کلید عمومی مهاجم رمزگذاری می‌شوند و مهاجم می‌تواند آن‌ها را رمزگشایی کرده، خوانده و دستکاری کند.

۲ SSL/TLS

(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 برقرار 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 پیاده‌سازی‌های 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 تایید شده هستند، رتبه بالاتری می‌دهند.

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

SSLLab یک 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:



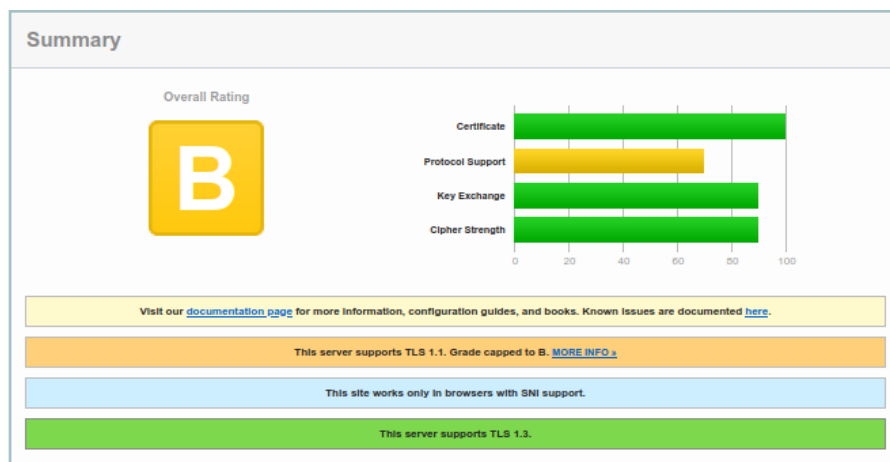
در ادامه، می‌توان گزارش دقیق‌تر از هر یک از پارامترهایی که در خلاصه‌ی گزارش نمره‌دهی داده‌شده‌اند را مشاهده کرد. برای مثال در بخش Certificate، اطلاعاتی نظیر نوع گواهینامه، تاریخ صدور، الگوریتم‌های استفاده شده در امضا و صادرکننده‌ی گواهینامه آمده‌است.

Certificate #1: RSA 2048 bits (SHA256withRSA)	
<div>Server Key and Certificate #1</div>	
Subject	sina.sharif.edu Fingerprint: SHA256: b6655a597edcd055bd3c70171ce40cc060123549b457e647a1a490a4fe52ca Pin SHA256: r3sTkxF1oKwHhN64QTdAa+DIDCKguPYY36TKNSZYQ~
Common names	sina.sharif.edu
Alternative names	sharif.edu sharif.ir sina.sharif.edu sina.sharif.ir
Serial Number	0451a55f9a9a415753d2753cc61d550d6605
Valid from	Sun, 26 May 2024 07:02:46 UTC
Valid until	Sat, 24 Aug 2024 07:02:45 UTC (expires in 2 months and 19 days)
Key	RSA 2048 bits (e 65537)
Weak key (Debian)	No
Issuer	R3 AIA: http://r3.o.lemon.org/
Signature algorithm	SHA256withRSA
Extended Validation	No
Certificate Transparency	Yes (certificate)
OCSP Must Staple	No
Revocation Information	OCSP: http://r3.o.lemon.org
Revocation status	Good (not revoked)
DNS CAA	No (more info)
Trusted	Yes Mozilla Apple Android Java Windows

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

Configuration				
Protocols				
TLS 1.3				Yes
TLS 1.2				Yes
TLS 1.1				No
TLS 1.0				No
SSL 3				No
SSL 2				No
Cipher Suites				
# TLS 1.3 (server has no preference)				
TLS_AES_128_GCM_SHA256 (0x1301)	ECDH x25519 (eq. 3072 bits RSA)	FS		128
TLS_AES_256_GCM_SHA384 (0x1302)	ECDH x25519 (eq. 3072 bits RSA)	FS		256
TLS_CHACHA20_POLY1305_SHA256 (0x1303)	ECDH x25519 (eq. 3072 bits RSA)	FS		256
# TLS 1.2 (server has no preference)				
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x9e)	DH 2048 bits	FS		128
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)	ECDH secp256r1 (eq. 15360 bits RSA)	FS		128
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x9f)	DH 2048 bits	FS		256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)	ECDH secp256r1 (eq. 15360 bits RSA)	FS		256
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)	ECDH secp256r1 (eq. 15360 bits RSA)	FS		256
Handshake Simulation				
Android 4.4.2	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1 FS
Android 5.0.0	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ECDH secp256r1 FS
Android 6.0	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ECDH secp256r1 FS
Android 7.0	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	ECDH x25519 FS
Android 8.0	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	ECDH x25519 FS
Android 8.1	-	TLS 1.3	TLS_CHACHA20_POLY1305_SHA256	ECDH x25519 FS
Android 9.0	-	TLS 1.3	TLS_CHACHA20_POLY1305_SHA256	ECDH x25519 FS
BitroPreview Jan 2015	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDH secp256r1 FS
Chrome 49 / XP SP3	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ECDH secp256r1 FS
Chrome 59 / Win 7 R	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ECDH x25519 FS
Chrome 70 / Win 10	-	TLS 1.3	TLS_AES_128_GCM_SHA256	ECDH x25519 FS
Chrome 80 / Win 10 R	-	TLS 1.3	TLS_AES_128_GCM_SHA256	ECDH x25519 FS
Firefox 31 3.0 ESR / Win 7	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ECDH secp256r1 FS
Firefox 47 / Win 7 R	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ECDH secp256r1 FS
Firefox 49 / XP SP3	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ECDH secp256r1 FS

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

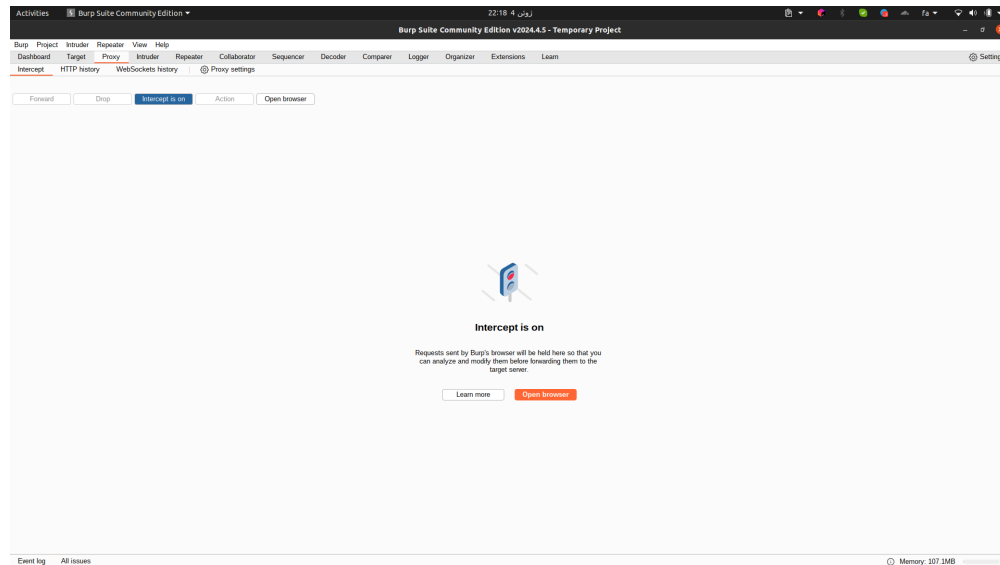


همان‌طور که در تصویر مشخص است، این سایت نمره‌ی پایین‌تری گرفته است و مهم‌ترین دلیل آن پشتیبانی از نسخه‌های قدیمی

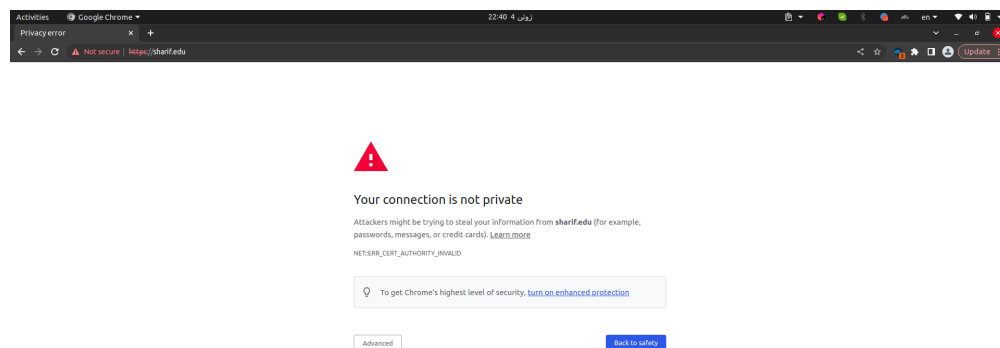
TLS است که آسیب‌پذیری‌های شناخته‌شده‌ای دارند.

(ت)

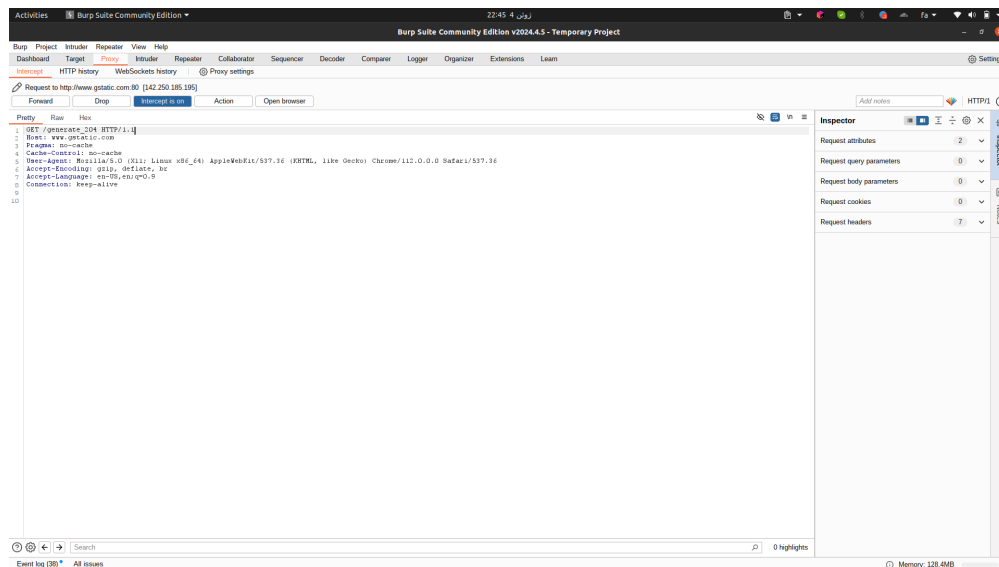
پس از نصب BurpSuite، یک پروژه‌ی جدید ساخته و وارد تب Proxy می‌شویم و Intercept را روشن می‌کنیم.



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 پورته 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 سرور 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 پورته 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 پورته 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 وصل شده 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.

```
۱ iptables -P INPUT DROP
۲ iptables -P FORWARD DROP
۳ iptables -P OUTPUT DROP
۴
۵ iptables -A INPUT -p tcp --dport 25 -s 192.168.56.1 -d 192.168.56.104 -j ACCEPT
۶
۷ iptables -A INPUT -p tcp --dport 1024:65535 -s 192.168.56.1 -d 192.168.56.104 -j ACCEPT
۸
۹ iptables -A OUTPUT -p tcp --dport 1024:65535 -s 192.168.56.104 -d 192.168.56.1 -j ACCEPT
۱۰
۱۱ iptables -A OUTPUT -p tcp --dport 25 -s 192.168.56.104 -d 192.168.56.1 -j ACCEPT
```

دقت کنید که می‌توانستیم به راحتی IP ها را با رنج آبی زیر شبکه جابجا کنیم و قواعد به این شکل می‌بودند:

```
۱ iptables -P INPUT DROP
۲ iptables -P FORWARD DROP
۳ iptables -P OUTPUT DROP
۴
۵ iptables -A INPUT -p tcp --dport 25 ! -s 192.168.56.0/24 -d 192.168.56.0/24 -j ACCEPT
۶
۷ iptables -A INPUT -p tcp --dport 1024:65535 ! -s 192.168.56.0/24 -d 192.168.56.0/24 -j ACCEPT
۸
۹ iptables -A OUTPUT -p tcp --dport 1024:65535 -s 192.168.56.0/24 ! -d 192.168.56.0/24 -j
ACCEPT
۱۰
۱۱ iptables -A OUTPUT -p tcp --dport 25 -s 192.168.56.0/24 ! -d 192.168.56.0/24 -j ACCEPT
```