

Practical 4

How to avoid intruders or attacks?

Alejandro Pérez

June 14th, 2023

Table of Contents

- Part 1 - Integrity with SHA
- Part 2 - Symmetric Encryption with GPG
- Part 3 - Asymmetric Encryption with GPG
- Part 4 - Apache Setup

Part 1 - Integrity with SHA

Exercise 1

a) I downloaded a small file and checked its corresponding SHA hash file from the following link:

Hak5 Download Portal

The file I downloaded is called `upgrade-2.1.3-stable.2022101708401.bin`. Here are the screenshots showing that the output hash is the same:

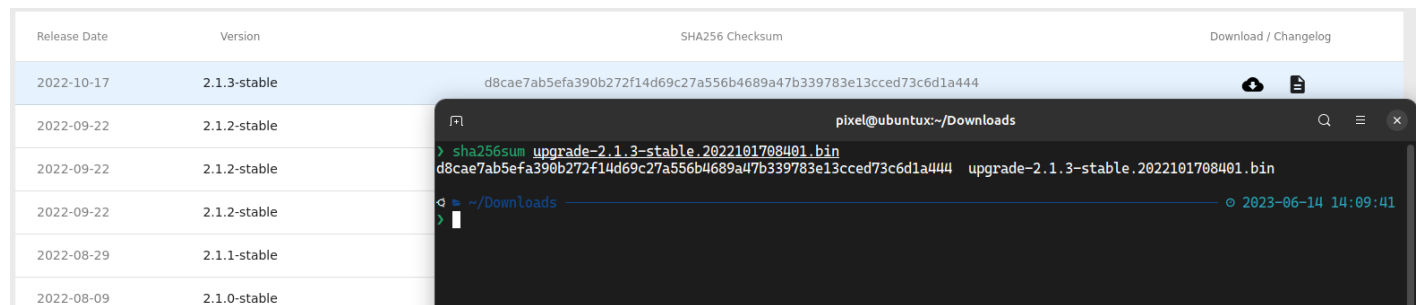


Figure 1: SHA File Check

b) There are three versions in the SHA family:

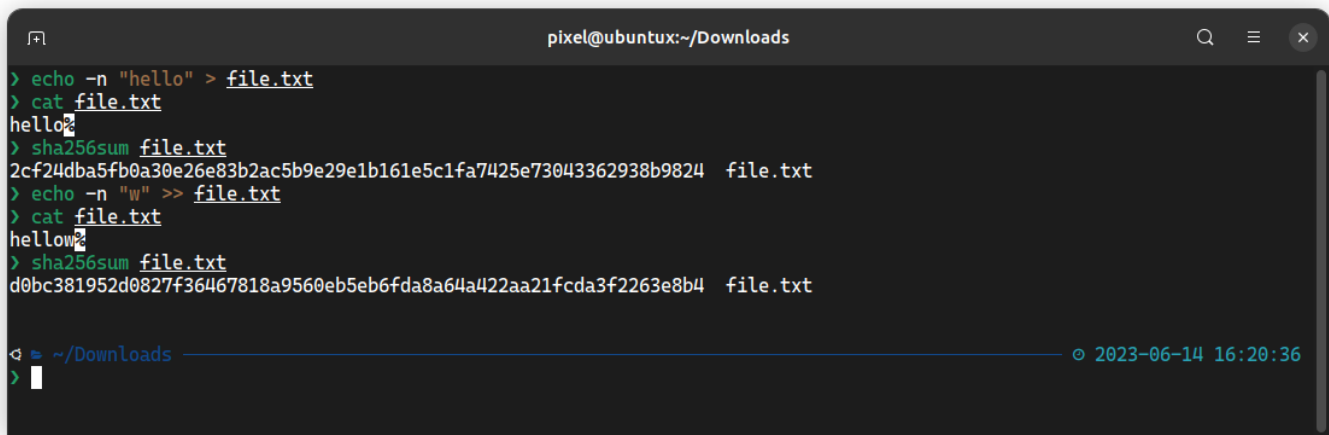
- **SHA1** produces a 160-bit checksum commonly used to verify data integrity and authenticate digital signatures.
- **SHA2** includes several hash functions that produce checksums of various lengths, from 224 bits to 512 bits.
- **SHA3** also includes multiple hash functions that generate checksums of various lengths from 224 bits to 512 bits.

The command above generated a 256-bit hash.

c) SHA0 was found to be vulnerable to collision attacks as early as 1998, which allowed an attacker to generate two different messages with the same checksum. This vulnerability made SHA0 rather insecure and it was soon replaced by SHA1.

SHA1 was also found to be vulnerable to collision attacks, and is no longer recommended for use in security-related applications. Both SHA2 and SHA3 were developed as alternatives to SHA1.

d)

A terminal window titled 'pixel@ubuntu:~/Downloads' with search, menu, and close icons in the top right. The terminal shows the following commands and output:

```
> echo -n "hello" > file.txt
> cat file.txt
hello
> sha256sum file.txt
2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824  file.txt
> echo -n "w" >> file.txt
> cat file.txt
hellow
> sha256sum file.txt
d0bc381952d0827f36467818a9560eb5eb6fda8a64a422aa21fcda3f2263e8b4  file.txt
```

At the bottom, a status bar shows a file icon, the path '~/.Downloads', a separator line, a clock icon, and the timestamp '2023-06-14 16:20:36'. A prompt character is visible on the line below the status bar.

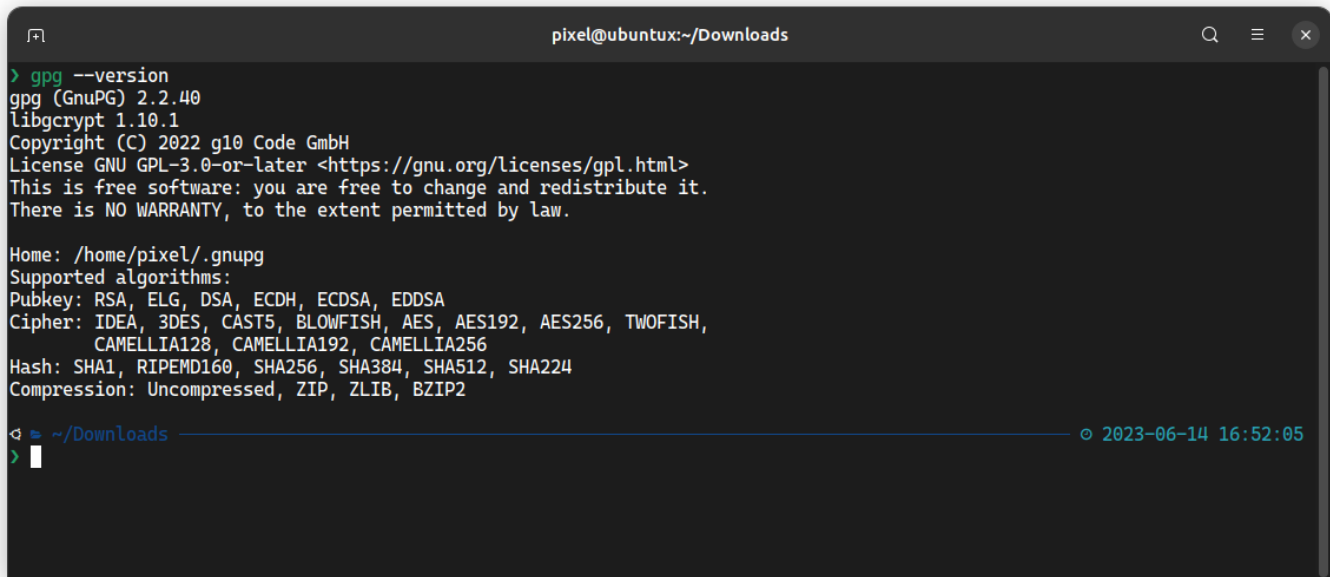
Figure 2: file.txt SHA2 Hash

- e) No, SHA functions are one way, meaning that you get the ciphertext from a plaintext but not the other way around

Part 2 - Symmetric Encryption with GPG

Exercise 2

- a) Here are the two main features of the key in symmetric encryption that make it tough to break:
- **Length:** A longer key indicates that an attacker would have to try more possible combinations, making a brute-force attack more time-consuming. For instance, cracking a 256-bit key would necessitate trying all 2256 potential keys, which is virtually difficult.
 - **Randomness:** The encrypted data could display patterns that can be exploited to determine the key if the key is not random or is predictable. Because it assures that the key and the encrypted data don't have any obvious patterns that an attacker may use to figure out the key, randomization is a crucial component for safeguarding against statistical assaults.
- b)
1. **Advanced Encryption Standard (AES):** the key length can be 128, 192, or 256 bits.
 2. **Blowfish:** the key length for Blowfish can range from 32 to 448 bits.
 3. **RC4:** its key length for can range from 40 to 2048 bits.
- c) According to the `gpg --version` command, the supported symmetric encryption algorithms are IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH, CAMELLIA128, CAMELLIA192 and CAMELLIA256.

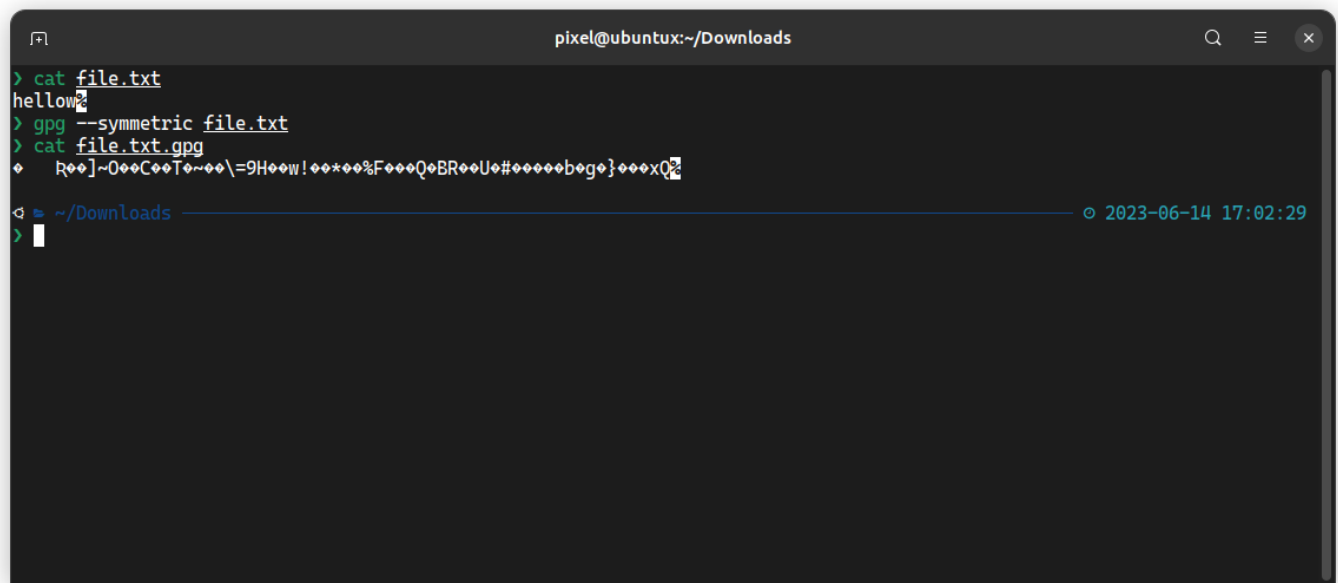


```
pixel@ubuntu:~/Downloads
> gpg --version
gpg (GnuPG) 2.2.40
libgcrypt 1.10.1
Copyright (C) 2022 g10 Code GmbH
License GNU GPL-3.0-or-later <https://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Home: /home/pixel/.gnupg
Supported algorithms:
Pubkey: RSA, ELG, DSA, ECDH, ECDSA, EDDSA
Cipher: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH,
        CAMELLIA128, CAMELLIA192, CAMELLIA256
Hash: SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compression: Uncompressed, ZIP, ZLIB, BZIP2

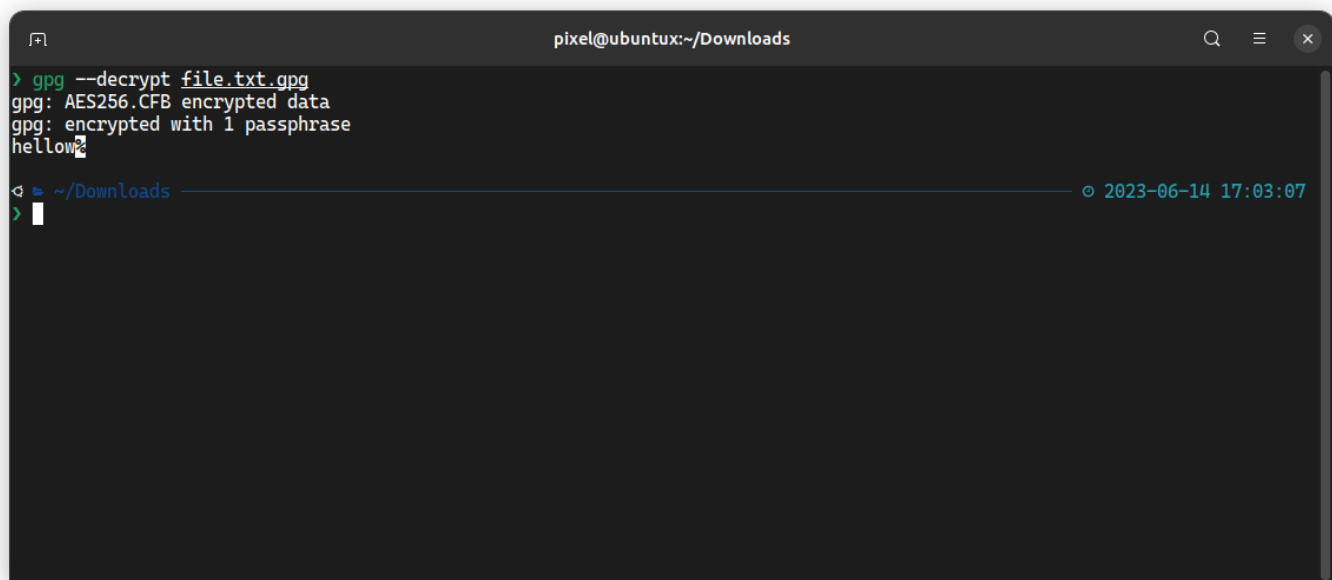
~ /Downloads 2023-06-14 16:52:05
```

Figure 3: `gpg --version`



```
pixel@ubuntu:~/Downloads
> cat file.txt
hello
> gpg --symmetric file.txt
> cat file.txt.gpg
R~0C~T~\=9H~w!~*~%F~Q~B~U~#~*~b~g~}~*~x~Q~
~ /Downloads 2023-06-14 17:02:29
```

d)

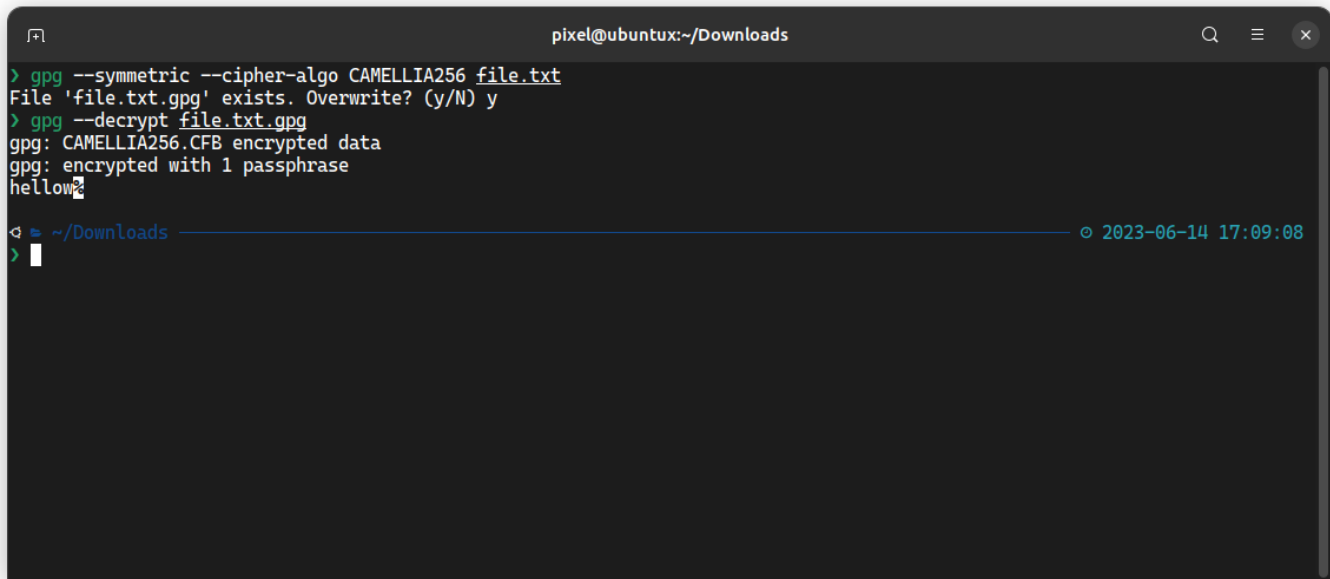


```
pixel@ubuntu:~/Downloads
> gpg --decrypt file.txt.gpg
gpg: AES256.CFB encrypted data
gpg: encrypted with 1 passphrase
hello
~ /Downloads 2023-06-14 17:03:07
```

Figure 4: Symmetric decryption with GPG

e)

f) The default symmetric cipher used is AES-128, but one can use the `--cipher-algo` to use another algorithm. Here is an example with CAMELLIA256:

A terminal window titled 'pixel@ubuntu: ~/Downloads' showing the execution of GPG commands. The user runs 'gpg --symmetric --cipher-algo CAMELLIA256 file.txt', which prompts 'File 'file.txt.gpg' exists. Overwrite? (y/N) y'. Then they run 'gpg --decrypt file.txt.gpg', which outputs 'gpg: CAMELLIA256.CFB encrypted data' and 'gpg: encrypted with 1 passphrase' before displaying the plaintext 'hellow'. The terminal also shows a directory navigation bar at the bottom with '~ /Downloads' and a timestamp '2023-06-14 17:09:08'.

```
> gpg --symmetric --cipher-algo CAMELLIA256 file.txt
File 'file.txt.gpg' exists. Overwrite? (y/N) y
> gpg --decrypt file.txt.gpg
gpg: CAMELLIA256.CFB encrypted data
gpg: encrypted with 1 passphrase
hellow
~ /Downloads 2023-06-14 17:09:08
>
```

Figure 5: CAMELLIA256 encryption and decryption

Part 3 - Asymmetric Encryption with GPG

Exercise 3

- a) Using the command `gpg --list-keys` returns no keys on this system.
- b) Here is how I created the new keys:

```

pixel@ubuntu:~/Downloads
> gpg --list-keys
> gpg --full-generate-key
gpg (GnuPG) 2.2.40; Copyright (C) 2022 g10 Code GmbH
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
  (1) RSA and RSA (default)
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072)
Requested keysize is 3072 bits
Please specify how long the key should be valid.
    0 = key does not expire
    <n> = key expires in n days
    <n>w = key expires in n weeks
    <n>m = key expires in n months
    <n>y = key expires in n years
Key is valid for? (0)
Key does not expire at all
Is this correct? (y/N) y

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

Real name: AlexKey
Email address: aperez-b@uoc.edu
Comment: N/A
You selected this USER-ID:
    "AlexKey (N/A) <aperez-b@uoc.edu>"

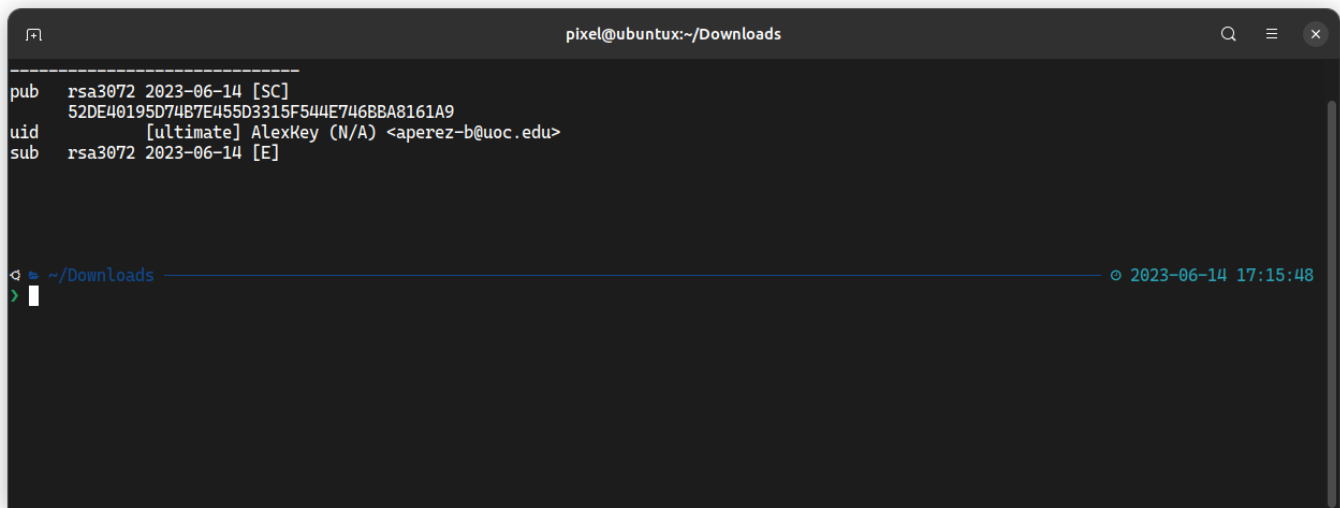
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? 0
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough 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: directory '/home/pixel/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/pixel/.gnupg/openpgp-revocs.d/52DE40195D7487E455D3315F544E7468BA8161A9.rev'
public and secret key created and signed.

pub   rsa3072 2023-06-14 [SC]
       52DE40195D7487E455D3315F544E7468BA8161A9
uid     AlexKey (N/A) <aperez-b@uoc.edu>
sub    rsa3072 2023-06-14 [E]


```

Figure 6: gpg Key Creation

- c)

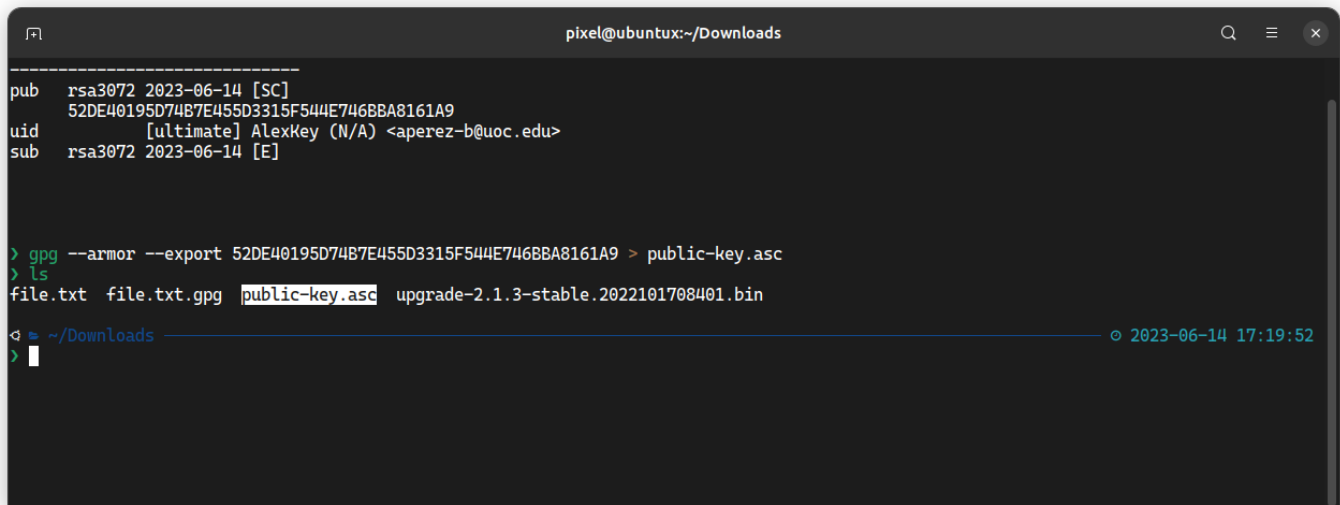


```
-----
pub  rsa3072 2023-06-14 [SC]
    52DE40195D74B7E455D3315F544E746BBA8161A9
uid  [ultimate] AlexKey (N/A) <aperez-b@uoc.edu>
sub  rsa3072 2023-06-14 [E]

gpg: ~/Downloads ----- 2023-06-14 17:15:48
> 
```

Figure 7: gpg Keys Created

Exercise 4



```
-----
pub  rsa3072 2023-06-14 [SC]
    52DE40195D74B7E455D3315F544E746BBA8161A9
uid  [ultimate] AlexKey (N/A) <aperez-b@uoc.edu>
sub  rsa3072 2023-06-14 [E]

> gpg --armor --export 52DE40195D74B7E455D3315F544E746BBA8161A9 > public-key.asc
> ls
file.txt  file.txt.gpg  public-key.asc  upgrade-2.1.3-stable.2022101708401.bin

gpg: ~/Downloads ----- 2023-06-14 17:19:52
> 
```

Figure 8: public-key.asc

- a)
- b) Note: For this part I have worked individually, so I sent the public key to myself and worked with two machines.

Exercise 5

```
> gpg --import public-key.asc
gpg: key 544E746BBA8161A9: "AlexKey (N/A) <aperez-b@uoc.edu>" not changed
gpg: Total number processed: 1
gpg:      unchanged: 1
> gpg --list-keys | tail -n 5
pub   rsa3072 2023-06-14 [SC]
      52DE40195D74B7E455D3315F544E746BBA8161A9
uid           [ unknown] AlexKey (N/A) <aperez-b@uoc.edu>
sub   rsa3072 2023-06-14 [E]

^  ~ /Downloads  ✖ arch  2023-06-14 15:26:34
> █
```

Figure 9: `gpg --import public-key.asc`

- a)
- b) See screenshot above

```
> cat NIA_P4_exer5_archive.txt
helloy
> gpg --encrypt --recipient 52DE40195D74B7E455D3315F544E746BBA8161A9 NIA_P4_exer5_archive.txt
gpg: 93119093B0D2A5F0: There is no assurance this key belongs to the named user

sub rsa3072/93119093B0D2A5F0 2023-06-14 AlexKey (N/A) <aperez-b@uoc.edu>
Primary key fingerprint: 52DE 4019 5D74 B7E4 55D3 315F 544E 746B BA81 61A9
Subkey fingerprint: 987D 2E9B 96B4 491C 7B42 FCDA 9311 9093 B0D2 A5F0

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

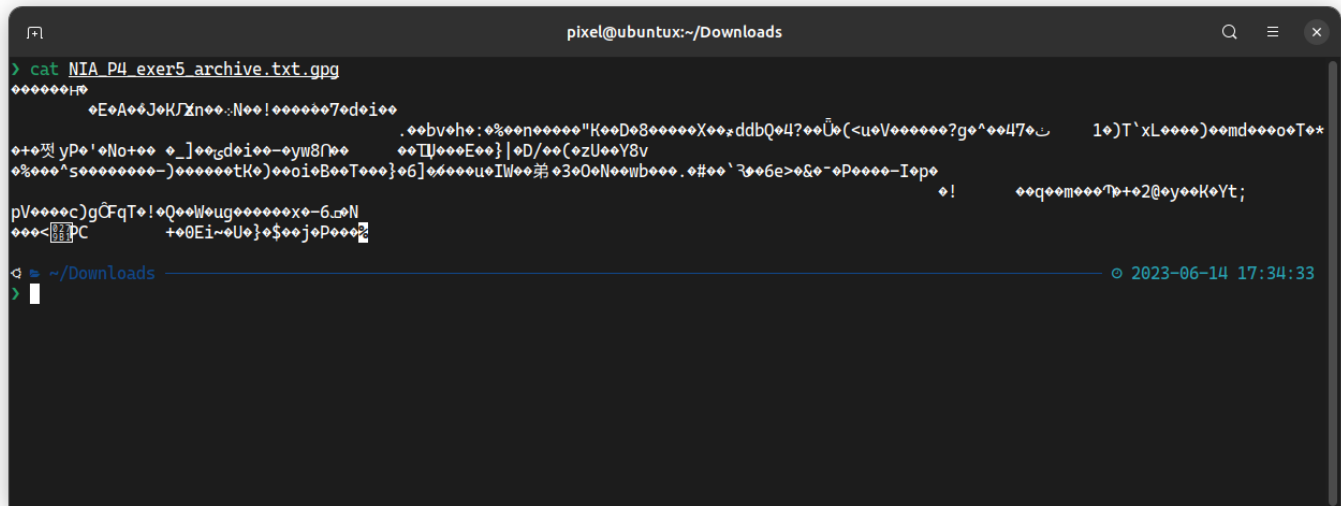
^ ~ /Downloads _____ * arch o 2023-06-14 15:30:38
> █
```

Figure 10: `gpg --encrypt --recipient 52DE40195D74B7E455D3315F544E746BBA8161A9 NIA_P4_exer5_archive.txt`

c)

d) Transferred encrypted file back to first machine (owner of the public key used for encryption)

Exercise 6



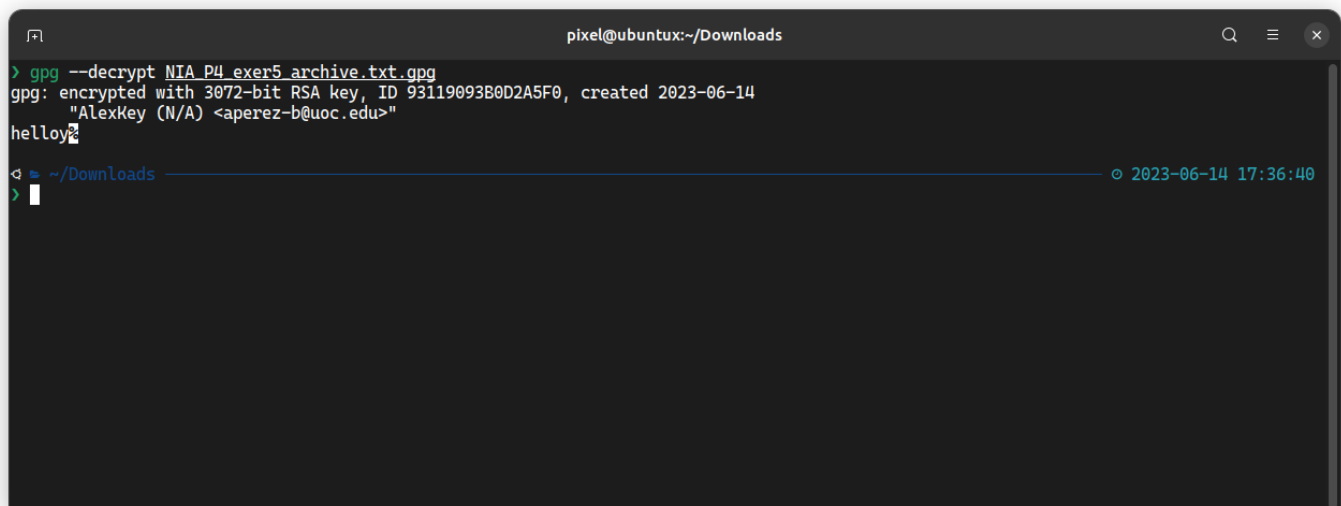
```

pixel@ubuntu:~/Downloads
> cat NIA_P4_exer5_archive.txt.gpg
#####
EAAAJKfXno.N!!#####7di+
. bvh: %en+ "K+D+8+X+ddBQ+4?+Ue(<uV+g+^+47+ 1+)T`xL+md+T+
+ yP+No+ _]++d+i+--yW8+ +[]++E+}|+D/++(+zU+Y8v
%+^s+++++--)+++++tK+)++oiB+T+++}+6]+++++uIW++第3+0+N++wb+++.#++`3+6e>+S+~+P++++-I+p+
+! +q+m+++7+2@+y++k+Yt;
pV+++c)gCFqT+!+Q++W+u+g+++++x+-6.+N
+++<P+C +0Ei~+U+}+$++j+P+++
d = ~/Downloads 2023-06-14 17:34:33
>

```

Figure 11: cat NIA_P4_exer5_archive.txt.gpg

a)



```

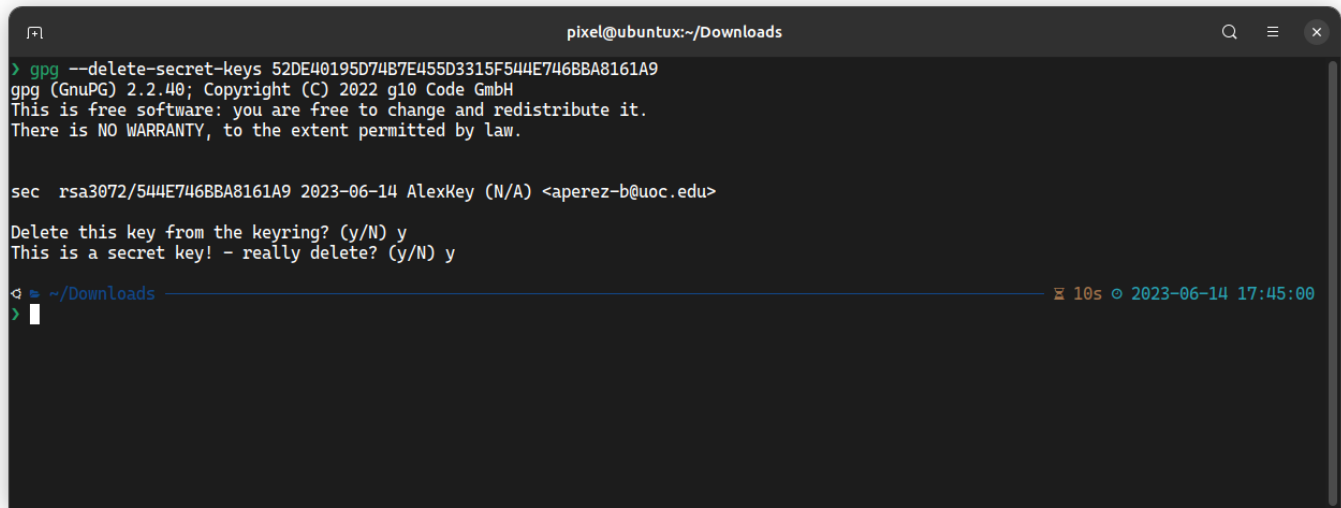
pixel@ubuntu:~/Downloads
> gpg --decrypt NIA_P4_exer5_archive.txt.gpg
gpg: encrypted with 3072-bit RSA key, ID 93119093B0D2A5F0, created 2023-06-14
"AlexKey (N/A) <aperez-b@uoc.edu>"
helloy
d = ~/Downloads 2023-06-14 17:36:40
>

```

Figure 12: gpg --decrypt NIA_P4_exer5_archive.txt.gpg

b)

c) From the screenshot above, you can see the plaintext **helloy**

Exercise 7

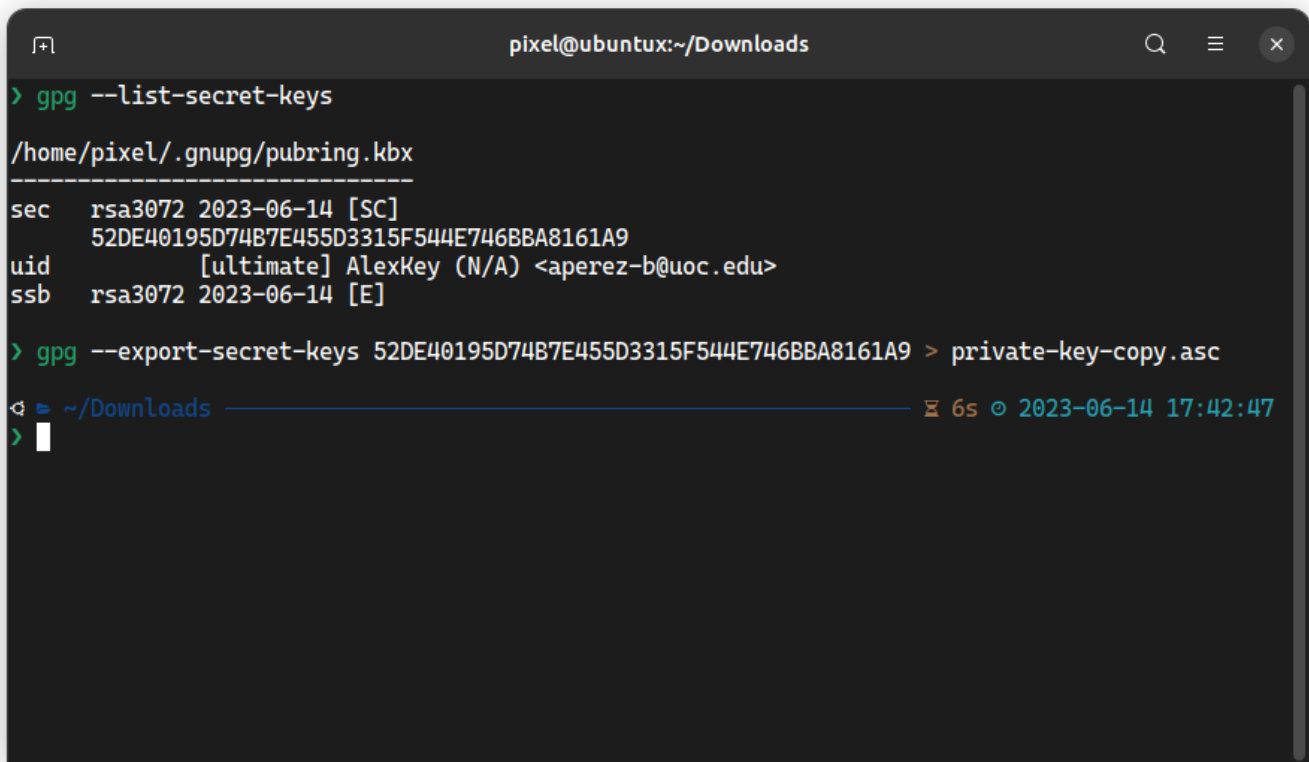
```
pixel@ubuntu:~/Downloads
> gpg --delete-secret-keys 52DE40195D74B7E455D3315F544E746BBA8161A9
gpg (GnuPG) 2.2.40; Copyright (C) 2022 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

sec rsa3072/544E746BBA8161A9 2023-06-14 AlexKey (N/A) <aperez-b@uoc.edu>
Delete this key from the keyring? (y/N) y
This is a secret key! - really delete? (y/N) y

~/.Downloads 10s 2023-06-14 17:45:00
> |
```

Figure 13: Delete a key

a)



```
pixel@ubuntu: ~/Downloads
> gpg --list-secret-keys
/home/pixel/.gnupg/pubring.kbx
-----
sec   rsa3072 2023-06-14 [SC]
      52DE40195D74B7E455D3315F544E746BBA8161A9
uid    [ultimate] AlexKey (N/A) <aperez-b@uoc.edu>
ssb    rsa3072 2023-06-14 [E]

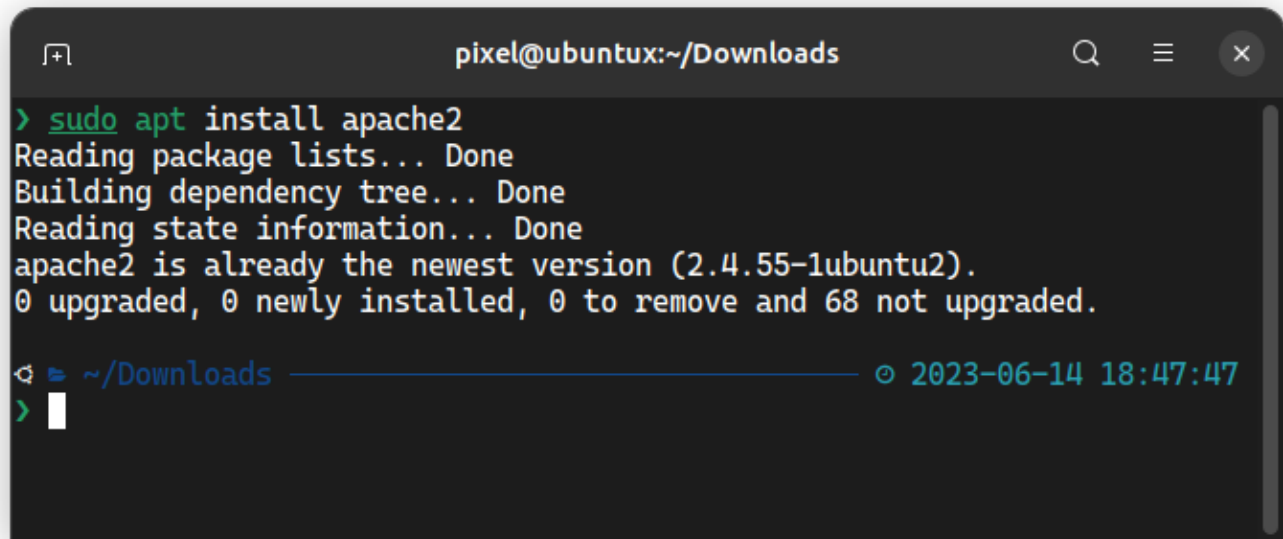
> gpg --export-secret-keys 52DE40195D74B7E455D3315F544E746BBA8161A9 > private-key-copy.asc
6s 2023-06-14 17:42:47
>
```

Figure 14: Export a private key

b)

Part 4 - Apache Setup

Exercise 8

A terminal window titled 'pixel@ubuntux:~/Downloads' with search, menu, and close icons in the title bar. The terminal shows the command 'sudo apt install apache2' being executed. The output indicates that the package lists, dependency tree, and state information are all read successfully. It then states that 'apache2 is already the newest version (2.4.55-1ubuntu2)' and that '0 upgraded, 0 newly installed, 0 to remove and 68 not upgraded.' The prompt returns to '~/' and the user is in the 'Downloads' directory.

```
> sudo apt install apache2
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apache2 is already the newest version (2.4.55-1ubuntu2).
0 upgraded, 0 newly installed, 0 to remove and 68 not upgraded.

~ /Downloads 2023-06-14 18:47:47
> 
```

Figure 15: Apache installed

a)

b)

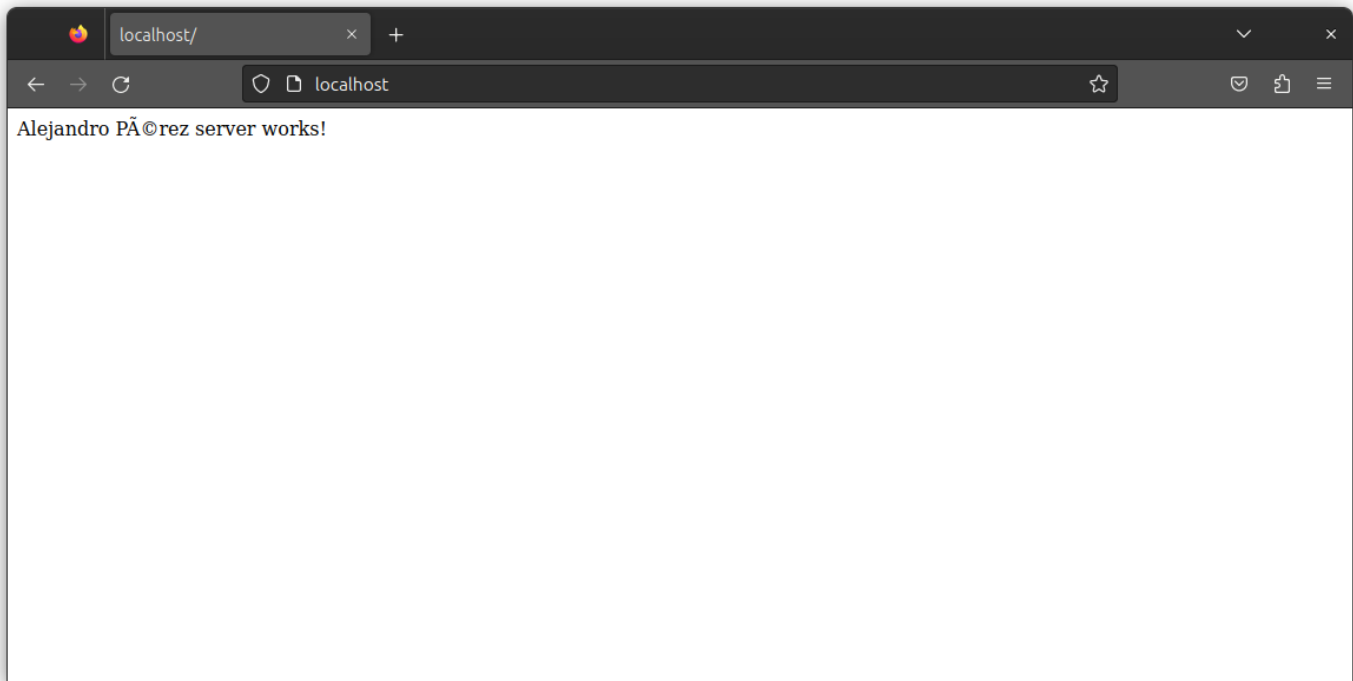


Figure 16: Apache working with custom site

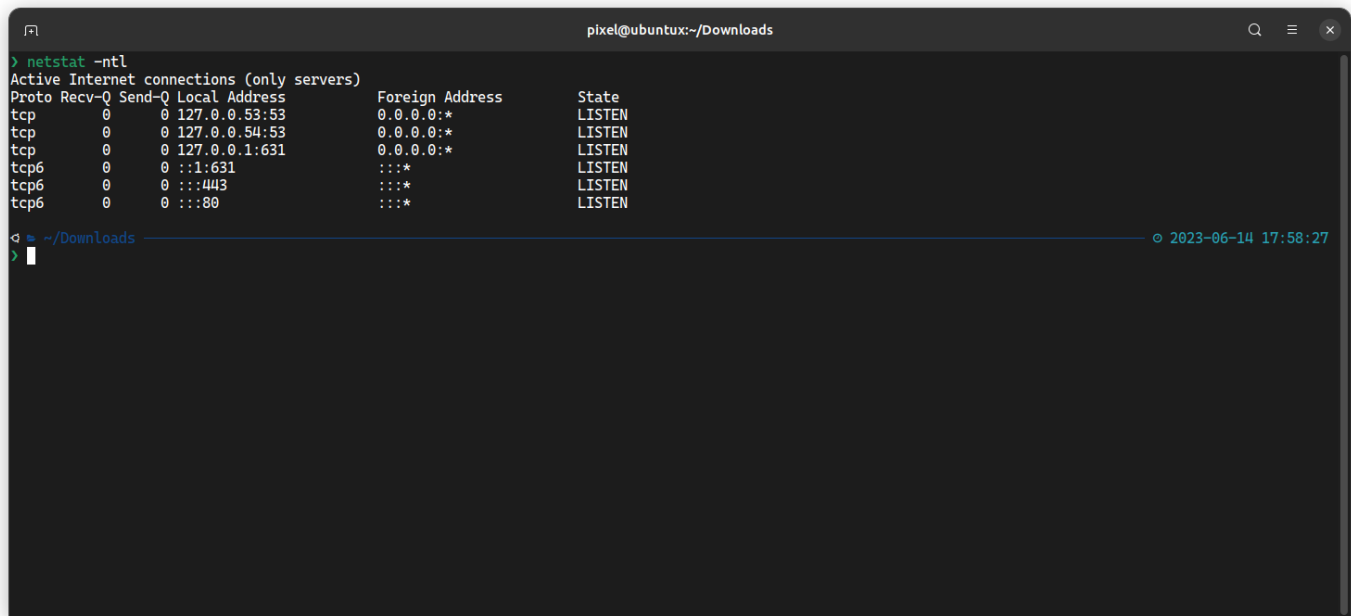
c)

- **SSLEngine**: enables the SSL/TLS encryption engine on the virtual host.
- **SSLCertificateFile**: specifies the path to the certificate file used to authenticate the server.
- **SSLCertificateKeyFile**: path to the private key file.
- **SSLCertificateChainFile**: path to the intermediate certificate file used to establish a chain of trust between the server's SSL/TLS certificate and a trusted root certificate.
- **SSLCACertificateFile**: path to the file that contains one or more trusted root certificates used to authenticate the client to the server.
- **SSLVerifyClient**: whether the server should request a client certificate for authentication.
- **SSLVerifyDepth**: maximum number of intermediate certificates allowed in a chain of trust between the client's certificate and a trusted root certificate.
- **SSLOptions**: various options that can be enabled, such as whether to require SSL/TLS encryption for all connections, whether to enable strict certificate checking, or whether to enable session caching.

```
> sudo a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
systemctl reload apache2
```

Figure 17: HTTPS enabled

d)

A terminal window titled 'pixel@ubuntu:~/Downloads' showing the output of the command 'netstat -ntln'. The output lists active internet connections for servers, showing protocols (tcp, tcp6), local addresses, foreign addresses, and states (LISTEN). The second-to-last line shows a listening socket on port 443, indicating HTTPS is enabled.

```
> netstat -ntln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.54:53           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp6       0      0 :::1:631                :::*                     LISTEN
tcp6       0      0 :::443                  :::*                     LISTEN
tcp6       0      0 :::80                   :::*                     LISTEN
```

Figure 18: netstat -ntln

- e) The second-to-last line in the output (the one with port 443) is the proof that HTTPS is enabled

Exercise 9

- **Protocol:** version of the SSL/TLS protocol in use, e.g. “TLSv1.2” or “SSLv3”.
- **Cipher:** shows that the current session uses symmetric encryption, e.g. “AES128-SHA”. This element shows how data is encrypted or decrypted in the client and server.
- **Session-ID:** 32-byte random number used as an identifier for the TLS/SSL session, used to resume a previous session.
- **TLS session ticket:** alternative to the Session-ID to resume TLD/SSL sessions. The server generates a session ticket (encrypted with a shared key) for the client to save and he uses it to resume the session afterwards.

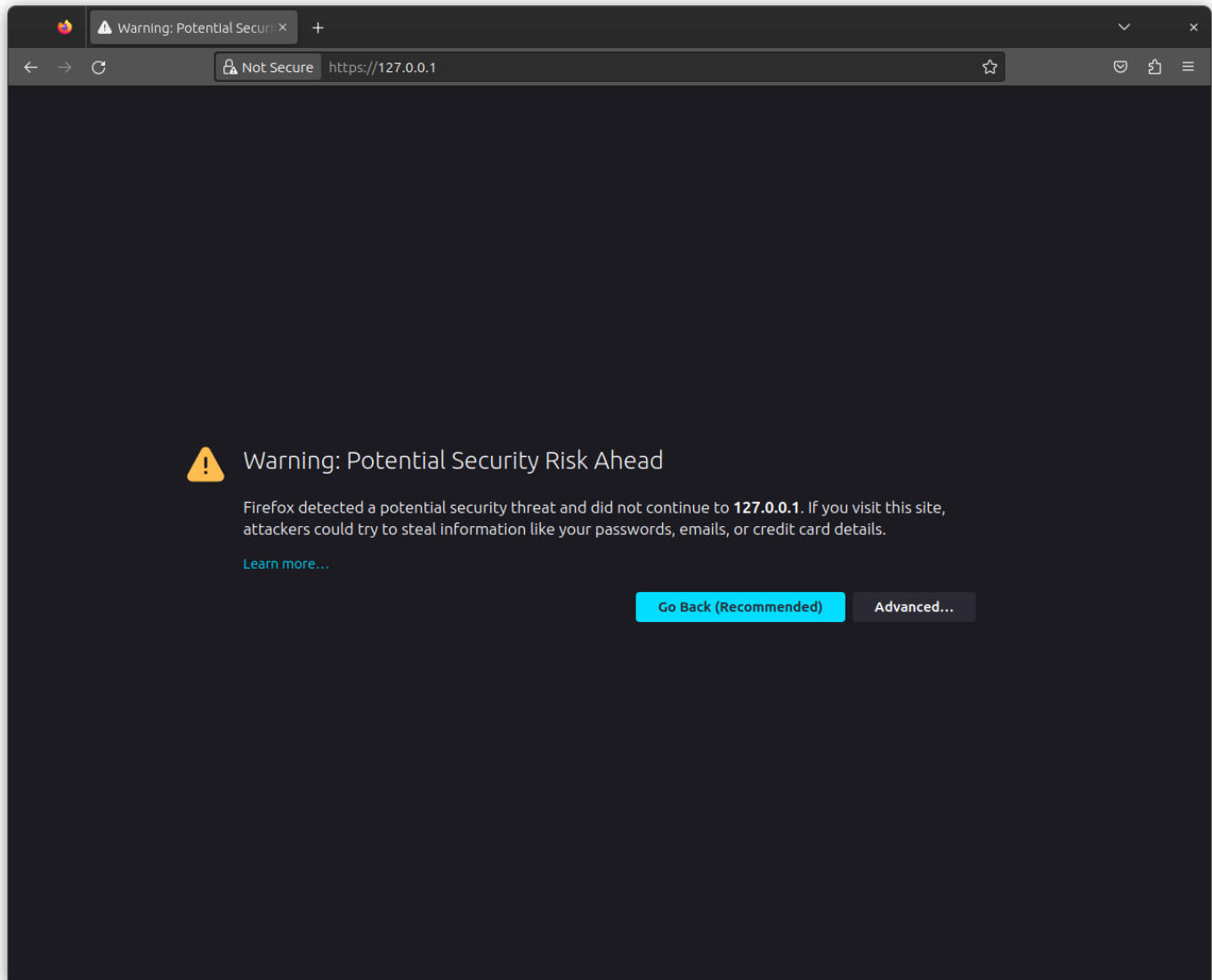
Exercise 10

Figure 19: Firefox warning

- a)
- b) The command connects to the server using OpenSSL and displays detailed information about the SSL/TLS connection, indicating the symmetric encryption algorithm being used for the current session. The “Protocol” field in the output indicates the version of the SSL/TLS protocol that is being used for the current session. For example, the protocol field may show “TLSv1.2” or “TLSv1.3”, and finally the “Certificate” field in the output indicates the SSL/TLS certificate that is being used for the current session.
- c) The three steps in the handshake are:
1. Client Hello: the client sends a “Client Hello” message to the server with information about the SSL/TLS version supported by the client, a list of supported cipher suites, and a random number generated by the client. This information is used by the server to find the best cipher suite and SSL/TLS version that both the client and server support.
 2. Server Hello: The server responds with a “Server Hello” message with information about the SSL/TLS version and cipher suite selected by the server, a random number generated by the server, and the server’s SSL/TLS certificate. The client uses the server’s SSL/TLS certificate to authenticate the server and establish the connection.

3. Client Key Exchange and Change Cipher Spec: The client sends a “Client Key Exchange” message to the server with the client’s public key, which is used to encrypt the session key. The client also sends a “Change Cipher Spec” message to the server saying that all subsequent messages will be encrypted using the agreed cipher suite and session key. Finally, the server responds with a “Change Cipher Spec” message of its own, saying that it is also ready.

June 14th, 2023