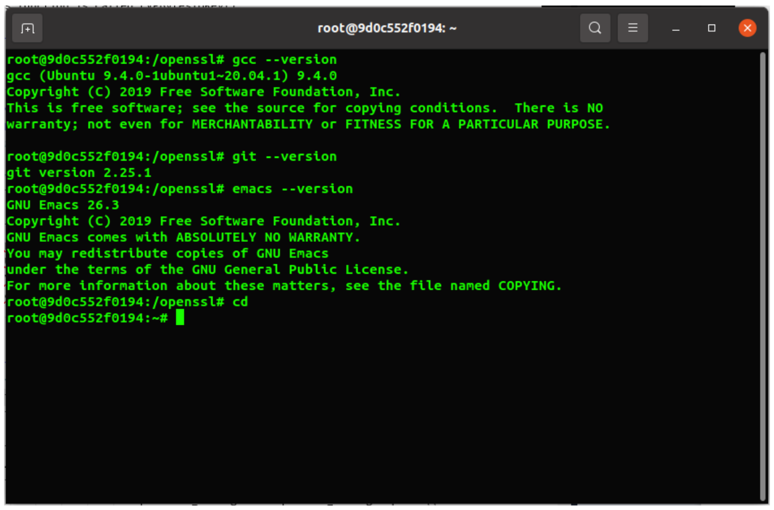
Networking the Physical World: Lab 3

Samir Farhat Dominguez

April 11, 2022

1 Part I: Docker

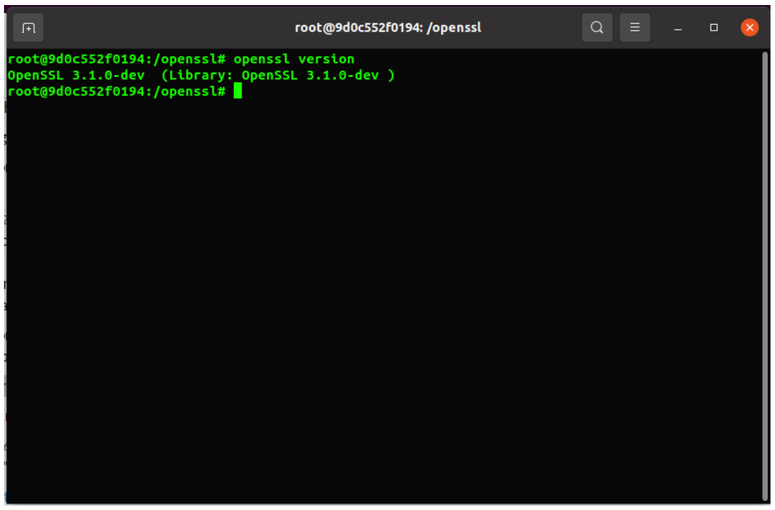


```
root@9d0c552f0194:/openssl# gcc --version
gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

root@9d0c552f0194:/openssl# git --version
git version 2.25.1
root@9d0c552f0194:/openssl# emacs --version
GNU Emacs 26.3
Copyright (C) 2019 Free Software Foundation, Inc.
GNU Emacs comes with ABSOLUTELY NO WARRANTY.
You may redistribute copies of GNU Emacs
under the terms of the GNU General Public License.
For more information about these matters, see the file named COPYING.
root@9d0c552f0194:/openssl# cd
root@9d0c552f0194:~# 
```

Figure 1: Docker container setup with utilities/software

2 Part II: Getting OpenSSL



```
root@9d0c552f0194:/openssl# openssl version
OpenSSL 3.1.0-dev  (Library: OpenSSL 3.1.0-dev )
root@9d0c552f0194:/openssl# 
```

Figure 2: OpenSSL installed with version `3.1.0-dev`

3 Part III: Using OpenSSL to do crypto!

3.1 Rubayat of Omar Khayyam

3.1.1 Hashing

```
root@2570fe3e2e46:~# wget http://www.gutenberg.org/cache/epub/246/pg246.txt
root@2570fe3e2e46:~# openssl dgst -md5 pg246.txt
MD5(pg246.txt)= 52cddbb1433f6b7d498ef07df0fa388b
root@2570fe3e2e46:~# openssl dgst -sha1 pg246.txt
SHA1(pg246.txt)= 93f23cb8ee9e0812e3b874a7fa2cdcb166cbebc1
root@2570fe3e2e46:~# openssl dgst -sha256 pg246.txt
SHA256(pg246.txt)= b80c88302b1c0a9e2a8642b610633662bd809af5e86c9d42ab5eaa6803b219d2
```

3.1.2 Which one (MD5 or SHA256) would you use to hash an important file? What would it mean if you "broke" MD5 or SHA256?

I would use SHA-256 as it is more secure, most obviously due to the output size being twice as long, which means the probability of collisions is lower. To break one of these schemes, would not mean to decrypt or revert the original text in the traditional encryption decryption models. Instead, brute force computations are used to break these hashing functions. Essentially we can attempt to send over every possible sequence of bits(128 for MD5 and 256 for SHA-256) that could correspond to a text, password, or other string hoping for a collision. This sort of attack is computationally intensive, but exponentially more difficult for each additional bit, so SHA-256 is much stronger.

3.2 2048-bit RSA Key-pair

```
root@2570fe3e2e46:~# openssl genrsa -out private.pem 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
................+++++
..............................+++++
e is 65537 (0x010001)
root@2570fe3e2e46:~# openssl rsa -in private.pem -pubout > public.pem
writing RSA key
```

3.3 Random 8 byte Hexadecimal Value

```
root@2570fe3e2e46:~# openssl rand -hex 8 > password
root@2570fe3e2e46:~# cat password
4aa9183b24414aa1
```

3.4 AES-256-CBC mode

```
root@2570fe3e2e46:~# rm plaintext
root@2570fe3e2e46:~# vim plaintext
root@2570fe3e2e46:~# cat plaintext
Hi! I'm Samir Farhat, BUID: U471707119
root@2570fe3e2e46:~# openssl aes-256-cbc -a -in plaintext -out \\
                              ciphertext -pbkdf2 -pass pass:4aa9183b24414aa1
root@2570fe3e2e46:~# cat ciphertext
U2FsdGVkX18BcTBQlx6/2e3Jjj9e09uTKP6HgHefKyFXfIM5jLNr2339PT9ggcA5
+2LHl5LGCuqxhW1domKRCw==
```

3.5 Regarding the Initialization Vector

AES in CBC-mode can only work with the initialization vector(IV). In Cipher Block Chaining(CBC) blocks of the plaintext are XOR'd with the previous block. Since there is not block prior to the first, this specific block is XOR'd with the IV. In the case of OpenSSL, if a password is used the IV does not need to be explicitly outlined. OpenSSL takes the password and mashes it up with an internal key derivation function to produce the key and IV. This function is called EVPBytesToKey().

3.6 Encrypt the Password

```
root@2570fe3e2e46:~# openssl rsautl -in password -out pass.enc -pubin \\
                            -inkey partners_pub.pem -encrypt
root@2570fe3e2e46:~# base64 pass.enc
```

```
Oxqa+cTp9Vm5N5nAdlSjl0NHCDJ+kFGiekHvIkKBHowkWsBUCBzu9pHtRYLUh5OBM9MlZH3BLJLB
HnDeZ1bUZ57DdbOYQ4ZiSd0qaZ46JveOevUFCP7Mw9/6Ku8vVSMbJGTxGxgkoGYP8wgoeozQWKwS
PXcRurIV7tN4DOwvxQxkH126d8lJ+RU1B7TZraslVGRj2UnGFZ0RMQsZXhuJ6tBcPNi9j1Ulb1UE
JaJHnVLeY/w7UezHYjZTBlXIRoKhvTvYvgZgjd+OI8+26/VheIn3BIgPZfxviD5ZD3ePLpAHJzMA
UH9eIx5/QBYDrN0RMSnwKHzAiNEtW7rtaoeYtQ==
```

3.7 Decryption

```
root@2570fe3e2e46:~# cat shared_secret.txt| base64 --decode \\
                        > shared_secret_raw.txt
root@2570fe3e2e46:~# openssl rsautl -in shared_secret_raw.txt -out \\
                            partner_pass.dec -inkey private.pem -decrypt
root@2570fe3e2e46:~# cat partner_pass.dec
2db79cc4386d9c24
root@2570fe3e2e46:~# openssl aes-256-cbc -d -a -pbkdf2 -in \\
                            partners_message -out partner_message -pass \\
                            pass:2db79cc4386d9c24
root@2570fe3e2e46:~# cat partner_message
Hi, this is Nuwapa P. (Prim). Have a good day!
```