Assignment #4 – 'PGP and Hashing'

Q1> Using PGP.

CTF Username: AyushShrivastava
Flag: 4386053827758958853302006748855

Verifying that the given public key is valid :

For email address abhishek.b@iitgn.ac.in, the OpenPGP fingerprint is 8074 FEB6 C3A0 C887 6E30 1878 3281 6904 E431 CB9A. We navigate to mailvelope keyServer to get the public key for the email address abhishek.b@iitgn.ac.in and store it in the file pgp.txt. We run the command,

```
gpg --show-key pgp
```

We get the below output proving that the public key belongs to <abhishek.b@iitgn.ac.in>.

```
pub     ed25519 2023-03-01   [SC]      [expires: 2025-02-28]
                8074FEB6C3A0C8876E30187832816904E431CB9A
uid     Abhishek Bichhawat    <abhishek.b@iitgn.ac.in>
sub     cv25519 2023-03-01   [E]       [expires: 2025-02-28]
```

Creating my own OpenPGP key on mailvelope :

For creating an OpenPGP key on mailvelope, we run the command `gpg --gen-key.`
We will be prompted for an email id and password. After providing the IITGN email id and some appropriate password, we run the following command.

```
gpg --output ./mypgp --armor --export shrivastavaayush@iitgn.ac.in
```

This will provide you with the PGP public key block, which we will upload to the mailvelope server. Upon which we will receive a mail from mailvelope to verify our email id.

PGP Chatbot Challenge :

Email id: Shrivastavaayush@iitgn.ac.in
Email id fingerprint: E4D96AB62A4C3E064FCAB88F20E2C84115EA2D9C

Step 1 :
We have made a txt file (Q1.txt) that contains my email address – Shrivastavaayush@iitgn.ac.in, and then we will run the command `gpg --clear-sign Q1.txt`

This gives the PGP signed message in the output file Q1.txt.ASC. We convert this obtained file to base64 using the online converter. We now plug this base64 value into the CTF server.

Step 2:
Target JSON to send: {"command":"get_flag"}
We have made a txt file (Q1_command) that contains the command "{"command": "get_flag"}" and then we will run the command `gpg --clear-sign Q1_command.`

This gives the PGP signed message in the output file Q1_command.ASC. We convert this obtained file to base64 using the online converter. We now plug this base64 value into the server. This will give us the personal flag as a PGP message. We will store this flag in a file flag.txt.

We will decrypt the flag using the command.

```
gpg --decrypt flag.txt
```

The Flag obtained is 43860538277589588533020067488555.

Relevant Files:

All obtained relevant files are present in the GitHub repo folder.

Q2> Hash Extension.

CTF Username: AyushShrivastava

The flag is: 199df8220aceba468bff497edfae209a

Explanation:
A length extension attack or Hash extension attack is a type of attack where an attacker can use Hash(message1) and the length of message1 to calculate Hash(message1 || message2) for an attacker-controlled message2 without needing to know the content of message1.

This is problematic when the hash is used as a message authentication code with the construction Hash(secret || message), and the message and the length of the secret are known because an attacker can include extra information at the end of the message and produce a valid hash without knowing the secret. Algorithms like MD5, SHA-1 and most of SHA-2 based on the Merkle–Damgård construction is susceptible to this such attacks.

Solution:
We get the original cookie and Hash value from the CTF server.
Original cookie (C): “username=nekomusume&groups=students,users,”
Length of the Cookie (C) = 42 bytes.

The secret key (SK) on the CTF server is 16 bytes long, which is unknown to us.
Let us assume SK is “xxxxxxxxxxxxxxxx.” An arbitrary unknown string.

Message packet = concat (SecretKey , Cookie) = SK||C
Length of message packet = 42 + 16 bytes = 58 bytes (464 bits or 0x1d0 bits)

To obtain SHA, we must pad this message packet to fit into a block size of 64 bytes. Following proper padding rules, the final message packet comes out to be of length 128 bytes.

Padded Message = b'xxxxxxxxxxxxxxxxusername=nekomusume&groups=students,users,\x80\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x01\xd0'

SHA obtained from the CTF server for the above padded message is:
“b54cb1fd4be14d5d36fdd2cdbda8c5b2b5bb8a621814ae2f6de3b42d5a824df7”

We want to append additional Malicious Information (MI) of “admins” into the above message.
So now our new message becomes,
New Cookie = b'xxxxxxxxxxxxxxxxusername=nekomusume&groups=students,users,\x80\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
01\xd0admins'

We now obtain the SHA value for this new cookie by initializing the older SHA value as the initial vectors h0,h1,h2,h3….h7. The new SHA value comes out to be,

New SHA = 519f788dcf2224debe3a6614802a98f7acff6dc0db44cc2e4325796bfd805517.

We plug the New Cookie and New SHA into the server and obtain the flag.

Code:
A very elaborate code explaining the whole process is available in the Jupyter file.

Additional Files and Code.

i. Link to the folder containing all the below files: (In case any of the link changes)
https://github.com/AYUSHs799/IITGN_CS431

ii. Link to Jupyter file for Hash Extension:
https://github.com/AYUSHs799/IITGN_CS431/blob/main/Assignment_4%20PGP%26Hashing/HashExtension.ipynb

iii. All relevant files for the PGP question are available in the GitHub repo folder
IITGN_CS431/Assignment_4 PGP&Hashing/PGP_files at main · AYUSHs799/IITGN_CS431 (github.com)

References.

i. https://textbook.cs161.org/crypto/hashes.html

ii. https://textbook.cs161.org/crypto/macs.html

iii. https://en.wikipedia.org/wiki/Length_extension_attack

iv. Mailvelope Key Server

v. http://www.gnupg.org/

vi. https://www.devdungeon.com/content/gpg-tutorial

vii. https://base64.guru/converter/encode/text