DH MITM (Level 2): Write-Up

The Flaw:

Due to the possibility of being able to insert & drop network traffic, an attacker can change both public keys to the number **1**. This will result in the shared secret key being equal to one respectively. This Attack is known as the “Simple Substitution Attack” (*refer to the Resource (Chapter 3.1.2) for more information*).
This Attack allows the attacker (Me) to “bypass” the check performed by the “Trusted Third Party” (TTP) and listen to the messages exchanged between Alice & Bob.

Solutions to mitigate this flaw would be on the one hand, to check if the shared secret is equal to 1. Another solution would be to use public-/private-key certificates (& digital signatures) which I have mentioned in my previous write-up (Level 1).

Solution:

1. I wrote a python script (based on my last script from Level 1) & modified it to fit this use case. This meant replacing the public keys with the integer 1 (encoded in base64).
2. Execute the python script (with the FQDN prefix as the “URL id”). The following messages are of interest:

```
None
{"nonce": "ucu3NHWqY1/+qVbWO4tCvQ==", "ctxt": "u28cZiH3n9pcNvcsNfuFq5O9bYgVz5Dz8LBKrbZT4xNX0wb
xf8N81yV3zY+ychF0nkB5jpTE", "tag": "asJfsg+YfnEDjwRxe0B+xg=="}
None
------------------------------------------------------------
Decoded: Oh come one, a little politeness never hurt anybody :)
------------------------------------------------------------
None
None
None
None
Message: Oh come one, a little politeness never hurt anybody :)
```

```
------------------------------------------------------------
Decoded: Don't be such a poor sport! I'll even make it easy for you and only choose a single w
ord.
------------------------------------------------------------
None
None
None
None
Message: Don't be such a poor sport! I'll even make it easy for you and only choose a single w
ord.
```

An educated guess leaves me to believe the word might be “Please”.

Resource:
https://www.researchgate.net/publication/2401745_Security_Issues_in_the_Diffie-Hellman_Key_Agreement_Protocol

3. Insert the word please as the message to be encoded.

```
241                 # Insert new values
242                 ws_cmd("2", url)
243                 ws_cmd("Alice", url)
244                 ws_cmd("Bob", url)
245                 encoded_msg = AESencrypt(
246                     aes_enc_key, decoded)
247                 ws_cmd(encoded_msg, url)
248                 decoded_i = decoded_i + 1
249             else:
250                 # Insert new values
251                 ws_cmd("2", url)
252                 ws_cmd("Bob", url)
253                 ws_cmd("Alice", url)
254                 encoded_msg = AESencrypt(
255                     aes_enc_key, "Please")
256                 ws_cmd(encoded_msg, url)
257                 decoded_i = decoded_i + 1
258
259
260 if __name__ == "__main__":
261     main()
262
```

4. Restart the Challenge in the web panel & run the script again (with the same “URL id”). The output will be the flag.

```
None
{"nonce": "Ijd3AmrShc3lBGVndBtJlg==", "ctxt": "whyma6UMrTpihOJhOLtUy6dbxJypJ6Tz9GbOLzLhDc+OaowMHW7eO4sRBrrfJvjA+dRk7SLTu8rWcuaXPOsFK1bMYUGF9Fh08t7mtOo9xJRvX+VoptQOkqga0gk=", "tag": "+n82An4aKB90Lo6QVEvePw=="}
None
------------------------------------------------------------
Decoded: Okay, that was way too easy for you. Here is the flag: 6227da62-f360-4d63-a0da-068239008213

------------------------------------------------------------
None
```

Resource:
https://www.researchgate.net/publication/2401745_Security_Issues_in_the_Diffie-Hellman_Key_Agreement_Protocol