Objective

In this hands-on activity, you will implement attacks on some vulnerable protocols (weak passwords, replay, reflection and XOR encryption). The objective is to develop an intuition about the threats that affect networked systems and the ease with which they can be exploited.

Starter Code

For each vulnerable protocol, you are expected to implement an `attack.py` file that exploits the specific vulnerability. All other code has been implemented for you. For reference, completed `attack.py` files are available for each vulnerable protocol inside the `src/` directory.

Vulnerable Protocols

The following vulnerable protocols have been implemented that you will be expected to attack.

Dictionary Attack

For this attack, you will implement a dictionary attack on the server used in Activity 1. This server is hosted at the URL `http://192.168.0.172:8888`. There is a user called `sashank_narain@uml.edu` with the weak dictionary password of `Acadia`. Use the dictionary available at `/usr/share/dict/words` to crack this password. A sample run of the attack is shown below.

```
$ python attack.py
Password found for sashank_narain@uml.edu -> Acadia
```

Replay Attack

An authentication protocol vulnerable to replay attack has been implemented in the source files `starter/replay/server.py` and `starter/replay/client.py`. Read these source files carefully, understand how the protocol works, understand why it is vulnerable to a replay attack, and then devise an attack for this protocol. A sample run of the attack is shown below.

```
$ python server.py (Terminal 1)

$ python client.py (Terminal 2)
Response: Login successful

$ python attack.py (Terminal 3)
Response: Login successful <- (Even though attacker is unaware of the credentials)
```

Reflection Attack

An authentication protocol vulnerable to reflection attack has been implemented in the source files `starter/reflection/server.py` and `starter/reflection/client.py`. Read these source files carefully, understand how the protocol works, understand why it is vulnerable to a reflection attack, and then devise an attack for this protocol. A sample run of the attack is shown below.

```
$ python server.py (Terminal 1)

$ python client.py (Terminal 2)
Response: Login successful

$ python attack.py (Terminal 3)
Response: Login successful <- (Even though attacker is unaware of the credentials)
```

Weak Encryption (XOR) Attack

For this attack, it is important to understand how XOR works. The authentication protocol has been implemented in the source files `starter/weak/server.py` and `starter/weak/client.py`. There is also `starter/weak/xor.py` that implements the XOR encryption and decryption methods. Again, make sure you understand how the protocol works and then devise an attack for this protocol. A sample run of the attack is shown below.

```
$ python server.py (Terminal 1)

$ python client.py (Terminal 2)
Response: Login successful

$ python attack.py (Terminal 3)
Secret Key: b"\xeb\x13%\xd1\xd3\x136\xc3\x07\x17\xa95\xe4\xb0\xb6\xa1"
Response: Login successful
```

Grading

The grading for this activity is straightforward.

- You will receive 100 points for attempting the attacks with your project group.
- You will receive 0 points if you don't show up or don't attempt the attacks.