ELEC5616

Project 1

Group 17

1. Diffie-Hellman Key Exchange

The first step of this project, we use Diffie-Hellman Algorithm to exchange cryptographic keys securely in some public and unsafe channels. First of all, we chose 4096 bits safe prime because amount of computation required to break any number of MODP Diffie-Hellman key exchanges is very close to the amount required to break just one of them, as long as they all use the same parameters. To avoid being broken by adversary, it's necessary to choose a large prime number in our security system. But 8192 bits safe prime is not a very appropriate choice because it will cost a lot of resource and time to compute. We set the generator to 2 which is hard enough to break. Next, we create some parameters of Diffie-Hellman :

- Prime number of 4096 bits: choose from RFC3526
- Generator = 2
- Private Key: generated by the random.randint() which is a number between 2 and prime-1.
- Public Key: $g^{my\_private\_key}\ mod\ Prime$

```
33  def create_dh_key():
34      generator = 2
35
36      # get random my_private_key 2 < g < prime-1
37      my_private_key = random.randint(2, prime - 1)
38      # calculate (g^m) mod prime
39      my_public_key = pow(generator, my_private_key, prime)
40      # Returns (public, private)
41      return (my_public_key, my_private_key)
42
```

2. Confidentiality

To ensure the confidentiality, we encrypted sending message by AES. As of today, no practicable attack against AES exists. Therefore, AES remains the preferred encryption standard for governments, banks and high security systems around the world. And we chose the CFB mode of AES because it makes a block cipher into a self-synchronizing stream cipher. And in this mode, encrypting process is

done for a small block of data. Besides, CFB mode can also avoid padding because the size of the blocks. It's normally chosen to fit the data unit to be encrypted.

And the process to define the AES is showing as follow:

```
31          # Key of AES
32          key = bytes.fromhex(self.shared_hash)
33          # initialisation vector is previous 16 bytes shared_hash
34          iv = self.shared_hash[:16]
35          # Create AES object with CFB mode
36          self.cipher = AES.new(key, AES.MODE_CFB,iv)
```

3. Integrity

To prevent attacks from tampering in the transit, we have to ensure the integrity and authentication of the original data. HMAC can help us to avoid such attack. First, we let the shared_hash be the key of HMAC and use this key to hash our message. Then we will get the authentication code which can be attached to messages send across the channel. By this way, we can verify the authenticity of messages received by comparing the original HMAC code and the one sends across.

When the message is sent:

```
50          #set up new HMAC object
51          self.hmac=HMAC.new(bytes(self.shared_hash[:64],encoding='ascii'),None)
52          data = data + ('\0' * add_zero).encode()
53          encrypted_data = self.cipher.encrypt(data)
54          self.hmac.update(encrypted_data)
55          message_hmac = self.hmac.hexdigest().encode()
56          encrypted_data = encrypted_data + message_hmac
57
```

The code to compare two HMAC codes:

```
92      if self.shared_hash:
93          #re-create the HMAC object
94          self.hmac = HMAC.new(bytes(self.shared_hash[:64], encoding='ascii'), None)
95          #the received HMAC
96          receive_hmac = encrypted_data[pkt_len -32:pkt_len]
97          self.hmac.update(encrypted_data[:-32])
98          #get the original HMAC
99          original_hmac = self.hmac.hexdigest().encode()
100         #check whether oringinal HMAC equals to received HMAC
101         if receive_hmac != original_hmac:
102             print("Tampering Attack!! closing connection.")
103             self.conn.close()
```

4. Preventing Replay

This program uses a timestamp to prevent replay attack. A replay attack is an attack that the eavesdropper pretends to be one of the participants by re-transmit the sniffed data. An unclear identified network connection between participants is the main reason of this attack. Create session, one-time passwords and timestamp can effectively prevent replay attack. For the act of eavesdropping on data, even if an eavesdropper gets the data, it will take extra time to resend the data to the target participant. The timestamp is mainly to limit the time difference between sending and receiving data between users and users and get the result of reducing the probability of replay attacks.

5. Peer-To-Peer file transfer between bots

This Skynet is built based on peer-to-peer networking (P2P), and P2P has a property that is resistant to attack and highly fault-tolerant ability. In a P2P network, even if a computer is attacked by someone, it will not have a big impact on other nodes. Moreover, the connection between nodes can automatically be adjusted to ensure that the entire network is availability. P2P networks also have a good privacy protection. The transmission of files does not need to pass the specific node, and the probability of eavesdropping on information is reduced.

A central web server:

advantages:

- simple structure, easy to control
- low latency, fast to transmit files

disadvantages:

- high-burden of central server
- easy be controlled by hacker

6. Botnet in real world

The most significant threat to the real world of botnet is denial of service attacks (DDOS). The botnet begins to continuously access the target computer in a short period of time via the controlled bots to form the attack. This kind of attack is simple, dangerous, and difficult to defend. In addition, botnet also could send spam emails, mining bitcoins etc. Today, there are some mature measures to detect botnet. The first is honeypot technology. This method uses the honeypot to obtain a sample of the bot program, analyses it, and then uses different methods to defend botnet over the characteristics of these bot programs. Secondly, botnet can be discovered through network traffic analysis, in a

network where a botnet is deployed, network flow is different from a common network. The botnets can be discovered through research on network flow changes.