Victoria Carlos
Aneil Singh
11/26/18
Dr. Christensen

Port Knocking Project

Message Design

Format

Next, we had decided on focusing on a shared key that was hashed from the client to server. However this opened up the opportunity for playback attack. We want to implement an additional layer of security and decrease chances of successful eavesdropping from an intruder. To address this we use timestamp.

The final message format is as follow:

hashed_secret_knock:hashed_secret_key*

**with timestamp*

This way, there are 2 layers of security the intruder would have to bypass and know. Since its a hash, ideally, a intruder would not know the order of the knock or key or the significance of the colon

Encoding

knockClient.py

```
currTime = str(datetime.datetime.now().time().replace(microsecond=0))
combo = currTime + config.SECRET_KEY
seqHash = hashlib.sha256(config.SECRET_KNOCK.encode()).hexdigest()
clientHashKnock = hashlib.sha256(combo.encode()).hexdigest()
concatKnock = seqHash + ":" + clientHashKnock
clientSock.sendto(concatKnock.encode(), (config.UDP_IP, config.UDP_PORT))
```

- When the knockClient program is called, it grabs the current time (omitting microseconds)
- We concatenate the string of the current time + our shared secret into the var ‘combo’
- Next, we hash our correct port knock sequence
- Hash the combo
- Concatenate the 2 hashes, separated by a colon
- Send to UDP server

knockServer.py

```
data, addr = serverSock.recvfrom(1024)
currTime = str(datetime.datetime.now().time().replace(microsecond=0))
dataDec = data.decode()
```

```
dataSplit = dataDec.split(":")

serverHashKnock = hashPW(currTime+config.SECRET_KEY)

if dataSplit[0] == hashPW(config.SECRET_KNOCK): #if a match is found
       if dataSplit[1] == serverHashKnock:
              if addr[0] not in config.KNOWN_IP:
                     config.KNOWN_IP.append(addr[0]) #remember this host as a known ip
                     serverHashKnock = None
                     weblite.enableWeblite()
                     .
                     .
                     .
```

- This falls under the ‘while True’ loop when the server socket is listening
- Server receives the data and address of UDP packet
- Whenever a server receives a packet, the current time (from the server side) is called and stored
- Decode the received data to string
- Split the received data by colon
- Hash the time it received a packet + secret key
- Check if it passes sequence then key match cases
- If it passes, start a new *thread* and enable weblite for 10 sec
- Also, add this IP address as a known and trusted IP
- For all other failed knock cases/DOS attack prevention, we disable the server socket for 10 sec and boot up again

Rules

Assumptions made throughout the project:

- Values such as web port number, knock sequence, secret key are known to both the client and server. Such shared variables are all stored in config.py
- On the server side, the value of the hash secret key is never permanent, only created and used at that current instance
- There is some threshold of error at times regarding the time. Normally, the error of packages sent and received is off by milliseconds. When debugging together, at times, it was off by 1 second.