Writeup for phase 2

Jiahao Cai

Oct 7, 2018

0. Overview

I have to admit that I have some problems with phase 2, mostly because I am not familiar with multithread programming and Cryptography. Also, I have much coursework and research assignment to do, so time is very limited. I have implemented most part of the server, tried to show a good taste in my code. I hope you wouldn't think that's a over-engineering.

1. Things that I have implemented

- 1. A web server configured using command line args which can receive packets,
- 2. a log tool which can write error logs to file without blocking the main thread, I choose to use a daemon thread. It has an advantage and a disadvantage as far as I know:
  - pro: the server only need to create one thread for logging, not as many as the number of errors,
  - con: when the server quit abruptly, the daemon thread will quit immediately as well, which will lead to potential log loss, but since the serve is supposed to run for a long time, this will be a rare scenario, so I think it will not matter a lot.
    - While writing this writeup, I just realize that my design is not good, because the log values when something happens, e.g. a malicious packet makes the server crash, but using a daemon thread will lead to log loss in this scenario, which is very bad.

2. Things that I have not implemented

I met some problems and I didn't solve all of them in my limited time…

- 1. The server cannot check if the signature is correct. Actually I know to do it (see snippet 1 & 2 below), but I cannot import the raw RSA key into Python (see snippet 1), which is very very annoying.

```
# snippet 1
def load_pubkeys(self, keys_dict):
    """
    Args:
        keys_dict: (packet id, raw binary public key)
    Returns:
        (packet id, RSA public key obj)
    """
    """
    ** Not implemented: **
        Failed to import RSA public key
    """
    for pkt_id in keys_dict:
        # keys_dict[pkt_id] = RSA.importKey(pkt_id) # Not working!!!
        pass
    return keys_dict

# snippet 2
def signature_check(self, pkt_id, pkt_seq_num, content, signature, pubkey):
    """
        Return true if the signatrue is valid
        ** Not fully implemented: **
            because failed to import RSA key from file, see utils.load_pubkeys()
    """
    """
    # The code should be like this:
    expected_hash = pubkey.decrypt(signature)
    received_hash = sha256(content).hexdigest()
    if expected_hash != received_hash:
        self.verify_log_queue.put( # put the error log into queue
            "%s\n%s\n%s\n%s" %
            (self.utils.bytes_2_hex_str(pkt_id),
            self.utils.bytes_2_dec_str(pkt_seq_num),
            received_hash,
            expected_hash)
        )
        return False
    else:
        return True
```

- 2. The server cannot check if the checksum is correct because I am not sure how to calculate it. I have implemented a prototype of how I am going to do when I know it.

```
def checksum_check(self, pkt_id, pkt_seq_num, pkt_checksum_idx):
    """
        Return true if the checksums match
        ** Not fully implemented: **
            because haven't figured out how checksum works in this scenario
    """
    # The code should be like this:
    received_crc32 = "I don't know how to calulate it"
    expected_crc32 = "I don't know how to calulate it as well"
    if received_crc32 != expected_crc32:
        self.checksum_log_queue.put(
            "%s\n%s\n%s\n%s\n%s\n" %
            (self.utils.bytes_2_hex_str(pkt_id),
            self.utils.bytes_2_dec_str(pkt_seq_num),
            self.utils.bytes_2_dec_str(pkt_checksum_idx),
            received_crc32,
            expected_crc32)
        return False
    else:
        return True
    )
```

- 3. I didn't have time to figure out how to use the XOR cipher, all I know about XOR is approximately `a ^ b ^ b = a`.
- 4. As I didn't implemented the features above, the server didn't use the adorable cat as well.