Project 2 Extra Credit Handout
CS 6262: Network Security, Fall 2024

Release Date: September 25 2024, 00.00

Due Date: October 07 2024, 23.59

1 TLS Certificates (2 Points)

Certificates are used to bind a domain name (e.g. https://google.com) to an entity (Google, the company). A certificate authority (CA) is a trusted third-party that verifies that an entity controls access to its website. When you access a website on a browser, your browser checks for a valid TLS certificate. If one is unavailable or expired, it informs users that the connection is not safe. In this assignment, you will implement a part of this functionality.

Notes:

1. You will write a Python script using `pyOpenSSL` to fetch certificates from a website served via HTTPS.
2. You need to complete all of the stubbed functions to get full credit for this question.
3. Do not remove any functions or classes.
4. Use `pyOpenSSL` and built-in `socket` module to connect to a server, perform a TLS handshake, retrieve the certificate and extract information such as the start and end dates of the validity period, the certificate authority (CA), and public key. You do not need any other libraries.
5. Make sure the file is named `tls_certificate_grabber.py` when uploading it to Gradescope.
6. When doing handshakes with your TLS context manager, don't forget to set the server domain, because many large websites use Server Name Indication to provide different certificates based on the domain.

1.1 Starter Code

```
class TLSCertificateGrabber:
    # The hostname and port you are connecting to.
    def __init__(self, hostname, port):
        pass

    # This will be an <OpenSSL.SSL.Connection object>
    def connect_and_handshake(self) -> SSL.Connection:
        pass

    # This will be an <OpenSSL.crypto.X509 object>
    def get_certificate(self, ssl_sock: SSL.Connection) -> crypto.X509:
        pass

    # Format: "YYYYMMDDHHMMSSZ" (ASN.1 GeneralizedTime format)
    def get_validity_start(self, cert: crypto.X509) -> str:
        pass

    # Format: "YYYYMMDDHHMMSSZ" (ASN.1 GeneralizedTime format)
    def get_validity_end(self, cert: crypto.X509) -> str:
        pass

    # We only want the common name (CN)
    # If the Certificate Authority's Distinguished Name
    # in string format is "/C=US/O=Let's Encrypt/CN=R3",
    # return "R3"
    def get_certificate_authority(self, cert: crypto.X509) -> str:
        pass

    # Format: a PEM-encoded string, such as
    # "-----BEGIN PUBLIC KEY-----\nMIIBIjANBgkh...QIDAQAB\n-----END PUBLIC KEY-----\n"
    def get_public_key(self, cert: crypto.X509) -> str:
        pass

    # Ensure the function returns the existing values in
    # the original order
    def dump_certificate(self) -> (crypto.X509, str, str, str, str):
        ssl_sock = self.connect_and_handshake()
        cert = self.get_certificate(ssl_sock)
        not_before = self.get_validity_start(cert)
        not_after = self.get_validity_end(cert)
        issuer = self.get_certificate_authority(cert)
        public_key = self.get_public_key(cert)
        return cert, issuer, not_before, not_after, public_key
```

1.2 Resources

- DigiCert TLS/SSL Certificates Overview
- Python 3 socket module documentation
- Instructions for installing PyOpenSSL
- PyOpenSSL examples