**TNE30024**

Deploying Secure Engineering Applications Online

**Tutorial 5 (week 6)**

**Secure Communications in Practise**

1. Why is the TLS layer typically implemented as a Library at the Application Layer?

TLS is typically implemented as a library at the Application Layer for several reasons:

a) Flexibility: Implementing TLS as a library allows applications to easily integrate secure communication without having to implement the complex cryptographic protocols themselves.

b) Ease of updates: Security protocols often need updates to address vulnerabilities or implement new features. A library implementation makes it easier to update the TLS functionality across multiple applications.

c) Abstraction: The library provides a clean abstraction between the application logic and the security layer, allowing developers to focus on their application's core functionality.

d) Cross-platform compatibility: Libraries can be ported to different operating systems and environments, ensuring consistent TLS implementation across platforms.

1. Within the TLS libraries, how is a server verified by the client?

Server verification in TLS libraries typically involves the following steps:

a) Certificate chain validation: The client verifies the server's certificate chain up to a trusted root certificate authority (CA).

b) Hostname verification: The client checks that the server's certificate is issued for the domain name the client is connecting to.

c) Certificate expiration check: The client ensures the server's certificate is not expired or revoked.

d) Digital signature verification: The client verifies the digital signature on the server's certificate using the public key of the issuing CA.

1. In terms of certificate information, what does the server need to provide to the client in order to allow a connection to be established?

I'd be happy to answer your questions about TLS (Transport Layer Security) implementation and certificate handling. Let me address each of your questions in order:

1. Why is the TLS layer typically implemented as a Library at the Application Layer?

TLS is typically implemented as a library at the Application Layer for several reasons:

a) Flexibility: Implementing TLS as a library allows applications to easily integrate secure communication without having to implement the complex cryptographic protocols themselves.

b) Ease of updates: Security protocols often need updates to address vulnerabilities or implement new features. A library implementation makes it easier to update the TLS functionality across multiple applications.

c) Abstraction: The library provides a clean abstraction between the application logic and the security layer, allowing developers to focus on their application's core functionality.

d) Cross-platform compatibility: Libraries can be ported to different operating systems and environments, ensuring consistent TLS implementation across platforms.

1. Within the TLS libraries, how is a server verified by the client?

Server verification in TLS libraries typically involves the following steps:

a) Certificate chain validation: The client verifies the server's certificate chain up to a trusted root certificate authority (CA).

b) Hostname verification: The client checks that the server's certificate is issued for the domain name the client is connecting to.

c) Certificate expiration check: The client ensures the server's certificate is not expired or revoked.

d) Digital signature verification: The client verifies the digital signature on the server's certificate using the public key of the issuing CA.

1. In terms of certificate information, what does the server need to provide to the client in order to allow a connection to be established?

To establish a TLS connection, the server typically needs to provide:

a) Server certificate: Contains the server's public key, domain name, and other identifying information.

b) Certificate chain: Includes intermediate certificates linking the server certificate to a trusted root CA.

c) Server's public key: Used for key exchange and establishing the secure connection.

d) Supported cipher suites: A list of encryption algorithms the server supports.

e) TLS protocol version: The version of TLS the server is using.

1. When developing a software solution to use a certificate, what information/files does the program need to provide to the TLS libraries?

When developing software to use certificates with TLS libraries, the program typically needs to provide:

a) Trusted root CA certificates: A list of trusted root certificate authorities, often in a file called a "trust store" or "CA bundle".

b) Client certificate (if using client authentication): The client's own certificate for mutual TLS authentication.

c) Client private key (if using client authentication): The private key corresponding to the client certificate.

d) Server certificate and private key (for server-side implementations): The server's own certificate and corresponding private key.

e) Configuration settings: Such as preferred cipher suites, TLS version, and certificate verification options.

f) Hostname (for client-side implementations): The expected hostname of the server for certificate validation.

# Socket Programming

For the second half of the tutorial, we are going to write a very short python program to download a web page from a HTTPS server that is encrypted via a certificate.

***Note: Here I am going to use a sample code to demonstrate it. This will be similar to the one you are going to use in the lab.***

Purpose of the Programming section:

1. Familiarize with python code, library functions and their syntaxes
2. Understand that for a valid connection, the server certificate is authenticated against the stored central repository.

1

1. Understand that having an invalid connection will cause the TLS library and code to fail

1. Understand that a complete solution (one that you come up with by yourself) would: - catch the TLS errors to allow the program to fail nicely rather than crash

- allow provision of alternate root certificates to accept non-verified connections

**Step 1: Create a socket, wrap it with TLS Context and Verifying Connection**

**Step 2: Downloading a page**

**Step 3: Connect and Download page from Rule201**



2