SD3 Secure Web Application Development

Secure Socket Layer

Transport Layer Security

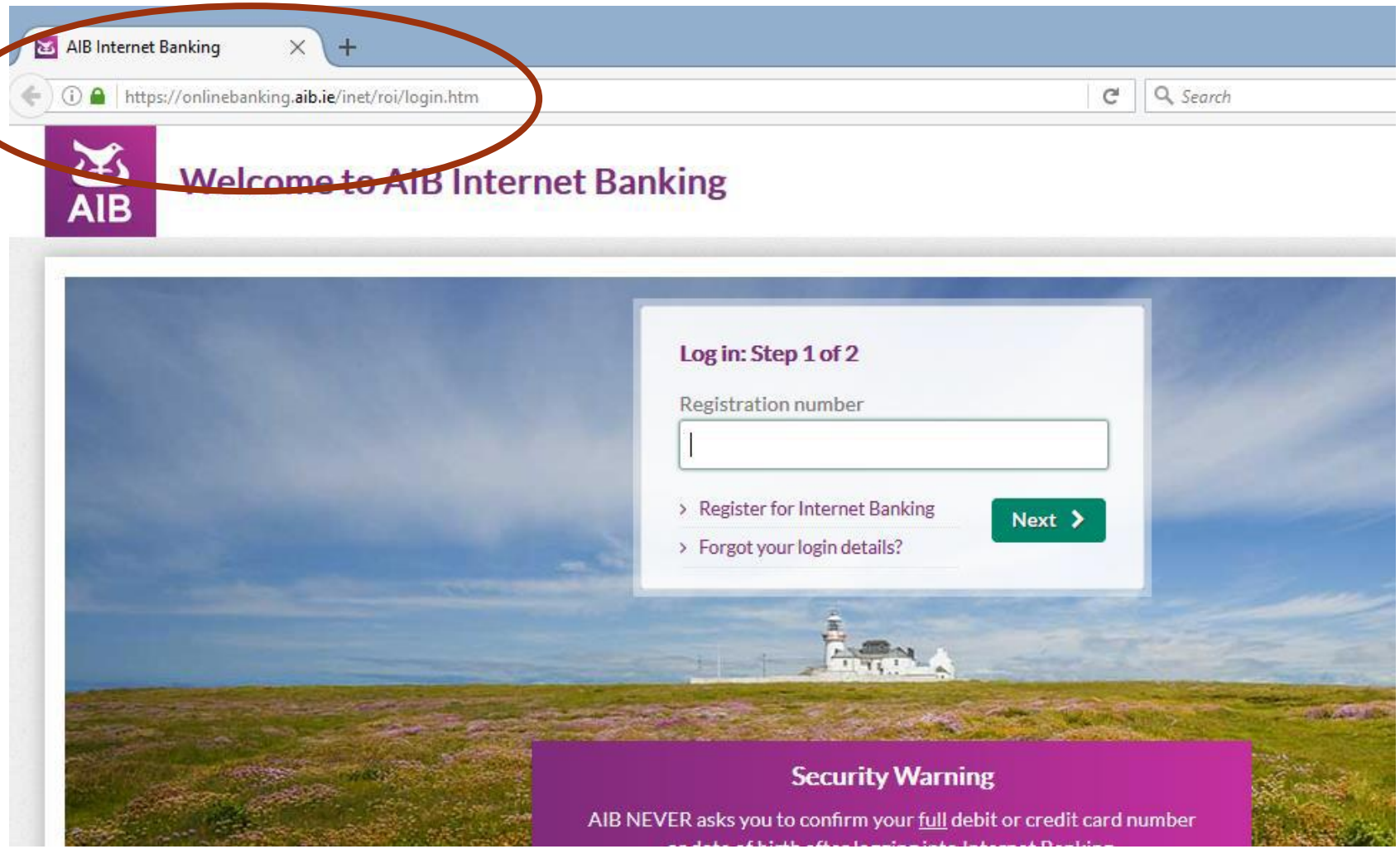
Introduction

- If your application works with sensitive data you should use a secure connection when you send data between the client and sever.
  - Otherwise its vulnerable to eavesdropping.
- SSL (Secure Sockets Layer) and Transport Layer Security (TLS) are protocols that lets you transfer data securely. TLS is a successor to SSL
- Two purposes of SSL/TLS.
  1. Verifying that you are talking directly to the server that you think you are talking to
  2. Ensuring that only the server can read what you send it and only you can read what it sends back

Introduction

- Even if someone were to intercept the message transmitted from the browser to the server, if they're encrypted they wont be able to make sense of any of the actual data you send.
  - They can roughly estimate how much data you're sending, but that's about it.
- There is now a trend amongst a number of websites to conduct conversations entirely using HTTPS.
  - HTTP traffic is vulnerable.
- SSL/TLS only protects data in transit from the browser -> server. It doesn't (obviously) protect your site from SQLi or XSS etc.

Introduction



- The user can infer this because HTTPS is being used rather that HTTP and also the browser displays a padlock.

Introduction

- With SSL/TSL the data is encrypted before it is transmitted between the browser and the server.
- If a hacker intercepts the data it is useless to them unless they can break the encryption code.
- As a user its hard to tell whether you are using TLS or SSL.
- Sending data over a secure connection is noticeably slower than regular HTTP connections.
  - Why?
    - Encryption/Decryption overhead

Introduction

- Another advantage of using SSL/TSL is that you can determine if data has been tampered with during transit.
- You can also verify that a client or a server is who they claim to be.

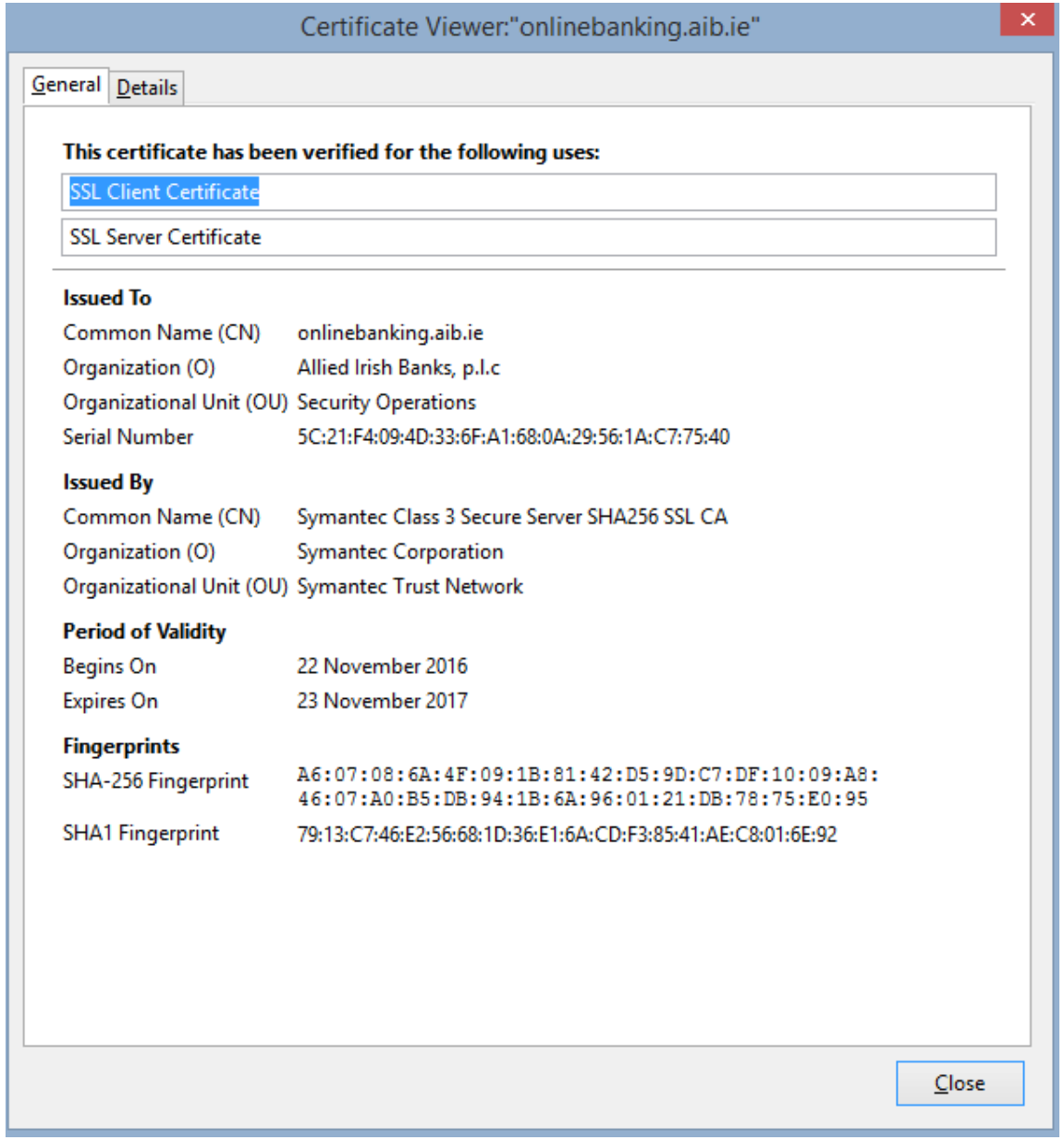
How SSL Works

- When a browser attempts to access a website that is secured by SSL, the browser & web server establish an SSL connection using a process called an “SSL Handshake”.
- To use SSL the client and server must provide authentication.
  - That way the client and server can accept or reject the secure connection.

How SSL Works

- Before a secure connection is established, the server uses SSL server to authenticate itself.
  - It does this by providing a digital secure certificate to the browser.
  - If the browser deems that the certificate hasn't come from a trusted source it informs the user.
  - The user can decide in this case if they want to trust the certificate or not.
  - If the user chooses to accept the certificate a secure connection is established.

Sample SSL Digital Certificate



How SSL Works

- There are two types of certificates:

1. **Server Certificate:** Issued to trusted servers so client computers can connect to them using secure connections.
2. **Client Certificate:** Issued to trusted clients so server computers can confirm their identity.

How To Get a Certificate

- If you want to establish a secure connection with your clients, you must get a digital certificate from a Certification Authority (CA).
  - The CA will run a background check on the company to verify them.
  - They will then issue the digital secure certificate.
- Digital certificates aren't free and the cost will depend on many factors such as the level of security.
  - You will need to decide on the level of strength (of encryption) you want your connection to support.

How To Get a Certificate

- In the early days of web programming, many web servers used certificates with 40-bit or 56-bit SSL strength.
  - At this strength its possible for a hackers to break the encryption code.
  - These strengths are appropriate for some sites.
- Today most browsers use 128-bit or higher SSL strength.
- Once you purchase a certificate you typically send it to your web host and they install it on your site.
- Once installed, clients can send data over a secure connection.

SSL Strengths

Strength	Pros and Cons
40-bit	Most browsers support it, but it's relatively easy to crack the encryption code.
56-bit	It's thousands of times stronger than 40-bit strength and most browsers support it, but it's still possible to crack the encryption code.
128-bit	It's over a trillion times a trillion times stronger than 40-bit strength, which makes it extremely difficult to crack the encryption code, but it's more expensive.
256-bit	It's virtually impossible to crack the encryption code, but it's more expensive and not all browsers support it.

Where can you get a certificate?

- You can create you own self-signed cert.
  - Used primarily in a testing environment.
- "Proper" certs can be obtained here.
  - https://www.websecurity.symantec.com/ssl-certificate
    - Both TSL & SSL Certificates
  - https://www.godaddy.com/web-security/ssl-certificate
  - https://www.globalsign.com
    - Both TSL & SSL Certificates
  - https://letsencrypt.org/ (free and backed by Mozilla)
    - Both TSL & SSL Certificates

Transport Layer Security (TLS)

- TLS is a cryptographic protocol that provides end-to-end communications security over networks and is widely used for internet communications and online transactions.
- It is an IETF standard intended to prevent eavesdropping, tampering and message forgery.
- Common applications that employ TLS include Web browsers, instant messaging, e-mail and voice over IP.
- Many businesses use TLS to secure all communications between their Web servers and browsers regardless of whether sensitive data is being transmitted.

TLS vs SSL

- TLS is more efficient and secure than SSL as it has stronger message authentication, key-material generation and other encryption algorithms.
- For example, TLS supports pre-shared keys, secure remote passwords, elliptical-curve keys and Kerberos whereas SSL does not.
- TLS and SSL are not interoperable, but TLS does offer backward compatibility for older devices still using SSL.

How TLS Works

- The TLS protocol specification defines two layers.
- The TLS record protocol provides connection security, and the TLS handshake protocol enables the client and server to authenticate each other and to negotiate security keys before any data is transmitted.
- The TLS handshake is a multi-step process. A basic TLS handshake involves the client and server sending “hello” messages, and the exchange of keys, cipher message and a finish message.
- The multi-step process is what makes TLS flexible enough to use in different applications because the format and order of exchange can be modified.

TLS – Current Version 1.3

- In March 2018 IETF released TLS 1.3 with “major improvements in the areas of security, performance and privacy”. The biggest change is that TLS 1.3 makes it significantly more difficult for attackers to decrypt HTTPS-encrypted traffic and therefore better protect privacy.
- Version 1.3 also makes the handshake process faster by speeding up the encryption process. This has a security benefit, but it should also improve performance of secure web applications. With TLS 1.2, the handshake process involved several round trips. With 1.3 only one round is required, and all the information is passed at that time.

Transport Layer Security

- TLS consists of two parts:
  1. The **TLS handshake layer** manages which cipher will be used, the authentication (using a certificate specific to your domain name and organization), and the key exchange (based on the public-private key pair from the certificate). The handshake process is performed only once to establish a secure network connection for both parties.
  2. The **TLS record layer** gets data from the user applications, encrypts it, fragments it to an appropriate size (as determined by the cipher), and sends it to the network transport layer.
- TLS establishes an encrypted, bidirectional network tunnel for arbitrary data to travel between two hosts. TLS is most often used in conjunction with other Internet protocols such as HTTPS, SSH, FTPS, and secure email.

HTTP vs HTTPS

- HTTPS is the HTTP protocol embedded within the SSL/TLS protocols. HTTP takes care of all the web surfing mechanics, and SSL/TLS takes care of encrypting the data sent over the network and verifying the identity of the server host using a certificate.
- More and more web servers are also going HTTPS-only, not just for security reasons, but for other practical arguments:
- **Some browser vendors now require HTTPS for certain browser features** (e.g., geo-location). And Google and Firefox intend to phase out non-encrypted HTTP in their browsers. So, the browser community is pushing for HTTPS as the standard.
- **Users expect a trust- and safety-indicating URL bar** (e.g., the padlock icon) without any security warnings, especially on eCommerce sites and other sites with privacy-sensitive data
- It may increase search engine ranking, too, though this has yet to be confirmed by Google.