Implementation and Simulation of Secure Sockets Layer (SSL) in Windows Presentation Foundation

Harsh Vachharajani
Master’s Thesis (ISA 799)
Summer 2015
George Mason University
Outline

• Introduction

• Previous Work

• Background of SSL/TLS and Attacks on It

• Implementation

• Limitations and Future Work
Introduction

- Insecure communication between a web server and browser
- Intercepting/sniffing messages transmitted through an insecure channel

without ssl

Information exchanged is insecure

[www.goclio.com]
• Insecure E-mail communication over HTTP between the e-mail server and the user
Previous Work

- Several Open-source programs available for teaching Cryptography
  - CrypTool
    Open source e-learning software to experiment with various cryptographic algorithms and programs in the area of cryptography and cryptanalysis

<table>
<thead>
<tr>
<th>CrypTool 1</th>
<th>CrypTool 2</th>
<th>JcrypTool</th>
<th>CrypTool Online</th>
</tr>
</thead>
<tbody>
<tr>
<td>First version of CrypTool that has simple GUI which demonstrates different cryptographic algorithms</td>
<td>Successor of CrypTool 1 that supports visual programming and has plugin based architecture</td>
<td>Modern, easy-to-use application that allows users to develop their own cryptographic plugins</td>
<td>Online version of CrypTool that allows users to experiment with different algorithms</td>
</tr>
</tbody>
</table>
Previous Work (cont…)

<table>
<thead>
<tr>
<th>CrypTool 1</th>
<th>CrypTool 2</th>
<th>JCryptTool</th>
<th>CrypTool Online</th>
</tr>
</thead>
<tbody>
<tr>
<td>Written in C++, runs on Windows</td>
<td>Written in C# using WPF, runs on Windows</td>
<td>Written in Java, which has Eclipse rich platform and runs on multiple platforms such as MAC OS X, Linux and Windows</td>
<td>Runs in a browser or on a smartphone</td>
</tr>
<tr>
<td>Visualization of several cryptographic algorithms and cryptanalytical methods</td>
<td>Plug’ n’ Play interface enabling workflow and visual programming of the algorithms by vector based GUI</td>
<td>Provides platform to experiment comprehensively with cryptography. Primary purpose is to develop crypto plugins and integrate into JCT</td>
<td>Developed primarily to study fundamentals of historical ciphers. Suitable for working with longer texts and conducting analysis of encrypted messages</td>
</tr>
</tbody>
</table>
Previous Work (cont...)
Previous Work (cont…)

– MAGMA
  • Known as Matrix Algebra on GPU and Multicore Architectures, this freely available software is designed for computations in algebra, geometry, number theory, etc.
  • Provides an environment to work with many different structures such as groups, rings, fields, graphs, etc.
  • Magma relies on features like integration, performance algebraic design, etc.
  • Can be used to encode public key operations used in SSL/TLS
  • Why not?

– SageMath
  • Free open-source mathematics software, which supports teaching in cryptography, algebra, geometry, etc.
  • Uses Python programming language as its base language
  • Features include text-based command line interface, different mathematical and number theory library functions
  • Available for multiple operating systems like Windows, Linux, Solaris, and OSX
  • Why not?
Previous Work (cont…)

– GnuPG
  • Open source implementation of the OpenPGP, designed to operate with PGP - the e-mail encryption program
  • Hybrid encryption software that uses a combination of conventional symmetric-key cryptography for speed and public-key cryptography for an easy secure key exchange
  • Supports various algorithms like:
    – Public key: RSA, ElGamal, DSA
    – Secret key: 3DES, AES-128/192/256, IDEA, etc.
    – Hash: MD5, SHA-1, SHA-256/384/512
  • Supported applications includes:
    – GPG4win
    – GPGMail
    – GPGTools
  • Can be used as a basis for the developed program, however because OpenPGP is different than SSL/TLS and the code is written in a different language, the extension would require substantial effort
Background of SSL/TLS

- What is SSL?
  - It is a cryptographic protocol that provides secure communication over an insecure network.
  - Establishes a link between the server and the client (generally a web server and the browser)
  - Makes use of certificates and asymmetric key cryptography to authenticate each party and to negotiate a symmetric key
  - Includes two sub protocols:
    - The Handshake protocol
      - Used to authenticate server/client and to generate session keys that will be used for the exchange of messages in the record protocol
    - The Record protocol
      - Used to exchange a series of messages between the authenticated server and client by using the cipher suites
Background of SSL/TLS (cont…)

- Difference between SSL/TLS versions:

<table>
<thead>
<tr>
<th>SSL Version 2.0</th>
<th>SSL Version 3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vulnerable to Man-in-the-Middle (MITM) attack</td>
<td>Defends against MITM attack by including the hash of all previous handshake messages in the last handshake message. May be still vulnerable to MITM through the cipher suite rollback attack.</td>
</tr>
<tr>
<td>Uses weak MAC construction</td>
<td>Uses strong MAC construction</td>
</tr>
<tr>
<td>Client can only initiate the handshake at the beginning of the connection</td>
<td>Client can initiate a handshake in the middle of an open session as well</td>
</tr>
<tr>
<td>Restrict the server and client from sending chains of certificates</td>
<td>Allows the server and the client to send chains of certificates</td>
</tr>
</tbody>
</table>
Background of SSL/TLS (cont…)

- **TLS 1.0**
  - An upgrade of SSL 3.0, hence uses standard HMAC compared to an older version of HMAC used by SSL 3.0
  - Key derivation functions and finished messages in the handshake protocol are different

- **TLS 1.1**
  - An upgrade to TLS 1.0
  - Implicit IV replaced with explicit IV for protecting against CBC attacks
  - Change in handling of padding errors

- **TLS 1.2**
  - MD5/SHA-1 combination in the PRFs and in the finished message was replaced by SHA-256
  - Addition of authenticated ciphers, AES-GCM and AES-CCM
  - Modified further to remove backward compatibility, meaning that TLS versions would never negotiate the use of SSL versions
Background of SSL/TLS (cont…)

- SSL 3.0 is no longer considered secure due to its weakness against the POODLE attack.

- BEAST attack can exploit web sites running SSL v3.0 and TLS v1.0, hence TLS v1.1 and TLS v1.2 considered more secure.
• Certification Authority (CA):
  
  – Is a trusted third party between the communicating parties

  – Certificates are digitally signed by the CA’s private key and the opponent verifying party verifies the signature by using CA’s public key; thus authenticating the contents of the certificate

  – Some of the certificate issuing companies include Symantec, VeriSign, Digicert and many more

• In SSL/TLS, the certificates are exchanged during the handshake phase by the communicating parties for authentication purposes
Background of SSL/TLS (cont…)

- Authentication and Key exchange algorithms:
  - Before the client and the server start exchanging any information, they must agree on the cipher suites
  
  - Several key exchange algorithms available for different SSL/TLS versions, where public/private key pairs are generated with
  
  - Public key certificates generated has varied key sizes as decided by the owner
  
  - Currently, 2048 bit public keys are considered much more secure and hence 1024 bit public keys are said to be no longer sufficiently secure
## Background of SSL/TLS (cont...)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>SSL 2.0</th>
<th>SSL 3.0</th>
<th>TLS 1.0</th>
<th>TLS 1.1</th>
<th>TLS 1.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DH-RSA</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DHE-RSA</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ECDH-RSA</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ECDH-ECDSA</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DH-DSS</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DHE-DSS</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Different cipher suites security against known attacks:

<table>
<thead>
<tr>
<th>Cipher</th>
<th>Algorithm</th>
<th>Strength</th>
<th>Protocol Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Algorithm</td>
<td>Strength</td>
<td>SSL 2.0</td>
</tr>
<tr>
<td>Block Cipher with Modes of Operation</td>
<td>AES GCM</td>
<td>256, 128</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>AES CBC</td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Camellia GCM</td>
<td>256, 128</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Camellia CBC</td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>3DES CBC</td>
<td>112</td>
<td>Insecure</td>
</tr>
</tbody>
</table>
### Background of SSL/TLS (cont…)

<table>
<thead>
<tr>
<th>Cipher</th>
<th>Type</th>
<th>Algorithm</th>
<th>Strength</th>
<th>SSL 2.0</th>
<th>SSL 3.0</th>
<th>TLS 1.0</th>
<th>TLS 1.1</th>
<th>TLS 1.2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cipher</strong></td>
<td><strong>Type</strong></td>
<td><strong>Algorithm</strong></td>
<td><strong>Strength</strong></td>
<td><strong>SSL 2.0</strong></td>
<td><strong>SSL 3.0</strong></td>
<td><strong>TLS 1.0</strong></td>
<td><strong>TLS 1.1</strong></td>
<td><strong>TLS 1.2</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>40</td>
<td>Insecure</td>
<td>Insecure</td>
<td>Insecure</td>
<td>Insecure</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>40</td>
<td>Insecure</td>
<td>Insecure</td>
<td>Insecure</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>40</td>
<td>Insecure</td>
<td>Insecure</td>
<td>Insecure</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Background of SSL/TLS (cont…)

- Data Integrity: Different MACs are used to provide data integrity during the data transmission
  - Takes arbitrary length of message to be authenticated as input, along with the secret key and outputs a MAC of fixed size length

- HMAC – Keyed-hash message authentication code:
  - Specific construction for calculating MAC that involves a cryptographic hash (e.g., MD5, SHA-1) in combination with the secret key

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>SSL 2.0</th>
<th>SSL 3.0</th>
<th>TLS 1.0</th>
<th>TLS 1.1</th>
<th>TLS 1.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMAC-MD5</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HMAC-SHA1</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HMAC-SHA256/384</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Background of SSL/TLS (cont…)

- The Handshake protocol:
Background of SSL/TLS (cont…)

– Authenticate the server to the client

– Allow the client and server to select the cryptographic algorithms, or ciphers, that they both support

– Optionally authenticate the client to the server

– Use public-key encryption techniques to generate shared secrets

– Establish an encrypted SSL connection
Background of SSL/TLS (cont…)

• The Record Protocol:
Attacks on SSL/TLS

• There are significant attacks possible on SSL/TLS.

• Cipher Suite Rollback Attack
  – A MITM attacker can alter the ClientHello message, which contains cipher suites supported by the client, strip off the undesirable cipher suites and replace it with the weaker ones
  – The server either rejects the connections if the cipher suites provided by the client (modified by the attacker) are not acceptable, or accepts the weaker ones
  – This attack was mitigated with SSL 3.0

• Version Rollback Attack
  – This attack is possible by modifying the list of cipher suites by the attacker, making a ClientHello message of SSL 3.0 look like a ClientHello message of SSL 2.0
Attacks on SSL/TLS (cont…)

– Forces the server to switch back to a more vulnerable SSL 2.0, thus allowing a way for more attacks to be done with the downgraded version
– As a mitigation, the PreMasterSecret of the ClientKeyExchange message contains the SSL/TLS version

• HeartBleed Attack
  – Attack specific to the implementation of the open source SSL/TLS library called OpenSSL
  – Exploit allowed the attackers to steal private keys from the server and allow anyone to sniff into the memory of the system that used the vulnerable OpenSSL version
  – Buffer Overflow was the flaw in the code, that compromised the private keys, passwords and other sensitive information of the users
  – Was mitigated by correcting the code and with the subsequent release of its next version
Attacks on SSL/TLS (cont…)

• Man-in-the-middle Attack
  – With the anonymous DH key exchange taking place, the attacker could perform MITM attack easily as there is no authentication involved
  – Can get a hold of the public parameters of both communicating parties and act as an intermediary for both of them
  – Neither of the communicating parties would be able to know that they are not communicating with each other and instead with the attacker
  – Successful breach of confidentiality and integrity of the message.

• POODLE, BEAST, CRIME are some of the other types of attacks possible with SSL/TLS
Development Tools and Libraries

- Used the same development platform as CrypTool 2 – C# as the programming language with .NET Framework 4.0
  - Provides a way to develop Windows client applications, XML web services, client-server applications, etc.
  - Provides a great platform to develop GUI rich and client based applications

- Used Visual Studio 2013 as the IDE

- For rich GUI, Windows Presentation Foundation (WPF) has been used
Development Tools and Libraries (cont…)

- WPF provides a platform to develop rich client applications
  - Allows the developers to create an application with vast range of elements like labels, textboxes, radiobuttons, checkboxes, etc.
  - Employs XAML to define interface elements; is responsible for the visual presentation of an application
  - Also supports built-for-user interfaces like 2D/3D rendering, animation and graphics with the elements, audio, video, etc.
  - Served as a perfect source for developing the program
Development Tools and Libraries (cont...)

• OpenSSL

  – Open source cryptography library that provides implementation of SSL/TLS protocols

  – Written in C language, it implements various basic cryptographic algorithms that sets the base for using it to develop different cryptographic applications

  – Implemented by most of the systems across the globe that rely on secure communication

  – Offers a command line interface which is used for the generation of public/private keys, certificates, calculating message digests, etc.
Development Tools and Libraries (cont…)

- MSDN (Microsoft Developer Network):
  - Contains a security library named “System.security.cryptography,” which has APIs, source codes, and other programming information related to all the cryptographic algorithms and other security features
  - Available for free download as a package with Microsoft development tools, like Visual Studio
  - Contains different classes that has constructors, methods and properties allowing the complete functionality of a defined module
Development Tools and Libraries (cont…)

• Tree structure of MSDN library, which defines its hierarchy:
Development Tools and Libraries (cont...)

- Mentalis.org Library
  - Free, open-source library that contains security library called Mentalis.org security library
  - Primary purpose for using this library was that it provided security related functions in C# for .NET development.
  - Supports authentication, cryptography and smartcard framework to connect and communicate with smart cards
Development Tools and Libraries (cont…)

- Library consists of:
  - SecureSocket Library
  - CertificateServices Library
  - Crypto Library
Development Tools and Libraries (cont…)

- Ssl3CipherSuites (Class)
  - GenerateMasterSecret(premaster, client random, server random)
    - Ssl3DeriveBytes(premaster, client random, server random, true)
      - Ssl3DeriveBytes(secret, client random, server random, clientServer)
      - GetNextBytes()
      - GetBytes(int cb)

  (Method in the above mentioned which takes several arguments)

  (Another class whose object is created inside the above defined method and this object is used to get a value of Master Secret)

  (Constructor of the above defined class)
Implementation

• Simulation of SSL protocol – the handshake protocol and the record protocol

• Visualization of its working

• Supported algorithms –
  – Public-Key algorithms: RSA, DH
  – Symmetric Key algorithms - DES, 3DES, RC2, AES-128/256
  – Hash Algorithms - MD5, SHA-1/256/384/512
Demonstration!
Implementation (cont…)

- Use of library function in the developed program

<table>
<thead>
<tr>
<th>Button</th>
<th>Library used</th>
<th>Underlying Class</th>
<th>Function defined under the class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certify Server</td>
<td>MSDN</td>
<td>RSACryptoServiceProvider, RSAParameters</td>
<td>ImportParameters(RSAParameters parameters)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ExportParameters(bool IncludePrivateParameters)</td>
</tr>
<tr>
<td>Verify Server Certificate</td>
<td>MSDN</td>
<td>RSACryptoServiceProvider, RSAParameters</td>
<td>ImportParameters(RSAParameters parameters)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ExportParameters(bool IncludePrivateParameters)</td>
</tr>
</tbody>
</table>
Implementation (cont...)
Limitations and Future Work

• Limitations:
  – Could not directly extend CrypTool 2 because of the code complexity and time constraints

• Library limitations:
  – Several library limitations restrict some major features related to SSL/TLS
  – AES–GCM mode not included in the MSDN library; thus the cipher suite combination with AES-GCM not supported
  – MSDN library is not implemented taking into account particular SSL/TLS versions
Limitations and Future Work

- For symmetric algorithms, RC4 algorithm not included as a part of MSDN
- For asymmetric key algorithms, Diffie-Hellman (not Elliptic Curve Diffie-Hellman) is not included in the MSDN library. Hence it had to be implemented manually

• Conclusions:
  - Successful implementation of the Handshake and Record Layers
  - Use of security libraries and their functions gives built-in functionality to perform required cryptographic operations
  - Better understanding through Simulation
  - Very useful as a learning module and as an educational tool
Future Work

• Implementing the Alert protocol and the change CipherSpec protocol in order to complete the entire SSL implementation

• Developing a laboratory exercise based on this program

• Inclusion of Man-in-the-middle attack as a part of the implementation

• Use of OpenSSL library might enhance the functionality and add many more features to the program
Questions
References


File:Ssl_handshake_with_two_way_authentication_with_certificates.png

[4] SSL/TLS in Detail,

References (cont...)


[15] C# WPF video, https://www.youtube.com/watch?v=krxYDsee2cQ


