CS 580 Client-Server Programming
Spring Semester, 2010
Doc 19 SSL & JavaMail
April 19, 2009

References

Transport Layer Security. (2010, April 18). In Wikipedia, The Free Encyclopedia. Retrieved 04:46, April 20, 2010, from http://en.wikipedia.org/w/index.php?title=Transport_Layer_Security&oldid=356880797

Java Networking Programming, 3rd Ed., Harold, O'Reilly, 2005, Chaters 11, 19

SSL & TLS

Secure Socket Layer (SSL)

- SSL1 never release (Netscape)
- SSL2 (1995)
- SSL3 (1996)

Use Public Key encryption
To pass private key

Client checks server certificate

Transport Layer Security (TLS)

- TLS1 (1999)
- TLS1.1 (2006)
- TLS1.2 (2008)

TLS allows server to check client certificate

X.509 Certificates

Pairs public key to a Name

Certificate contents

Version
Serial Number
Algorithm ID
Issuer
Validity
    Not Before
    Not After
Subject
Subject Public Key Info
    Public Key Algorithm
    Subject Public Key
Issuer Unique Identifier (Optional)
Subject Unique Identifier (Optional)
Extensions (Optional)
Certificate Signature Algorithm
Certificate Signature

Certificate Authority (CA)

Trusted companies/agencies that issue certificates

VeriSign (57% of market)

Microsoft Corporation Incident

2001 VeriSign issued certificate named "Microsoft Corporation" to person

Trusted CAs & Web Browsers

Web browsers have a list of trusted CAs

User gets warning if site uses certificate browser can't validate

Root Certificates

Certificates are signed using private key of issuer

Use public key to validate signature

Web browsers contain certificates of CAs (issuers)

Generating a Certificate using Java

```
Al pro 13->keytool -genkey -alias whitney -keystore exampleKeystore
Enter keystore password:
Keystore password is too short - must be at least 6 characters
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  Roger Whitney
What is the name of your organizational unit?
  [Unknown]:  Computer Science
What is the name of your organization?
  [Unknown]:  SDSU
What is the name of your City or Locality?
  [Unknown]:  San Diego
What is the name of your State or Province?
  [Unknown]:  CA
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=Roger Whitney, OU=Computer Science, O=SDSU, L=San Diego, ST=CA, C=US correct?
  [no]:  yes

Enter key password for <whitney>
        (RETURN if same as keystore password):
```

How TLS Works

Handshake

- Client & Server use public/private key system
- Exchange secrete key

Encrypted communication

- Using secrete key encrypt/decrypt communication

How TLS Works

Handshake

Client connect to server, gives list of cipher suites it supports

Server selects strongest cipher suite both support, notifies client

Server sends its certificate to client

Client may verify certificate with CA

Client

- Generates random number,
- Encrypts number with servers public key
- Sends number to server

Server & Client use random number to generate shared secret key

Java Client Side SSL/TLS

```
public static void main(String[] args) throws UnknownHostException,
        IOException {

    int port = 443;
    String hostname = "www.sdsu.edu";
    SocketFactory sslSocketFactory = SSLSocketFactory.getDefault();
    Socket socket =  sslSocketFactory.createSocket(hostname,
            port);

    InputStream in = socket.getInputStream();
    OutputStream out = socket.getOutputStream();

    // Read/Write to server

    in.close();
    out.close();
}
```

Some SSLSocket Extras

getSupportedCipherSuites

getSupportedProtocols

```
SocketFactory sslSocketFactory = SSLSocketFactory.getDefault();
SSLSocket socket = (SSLSocket) sslSocketFactory.createSocket(hostname, port);
for (String ciper : socket.getSupportedCipherSuites())
        System.out.println(ciper);
for (String ciper : socket.getSupportedProtocols())
        System.out.println(ciper);
```

Java JDK Supported Cipher Suites

SSL_RSA_WITH_RC4_128_MD5
SSL_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_RSA_WITH_AES_256_CBC_SHA
TLS_DHE_DSS_WITH_AES_128_CBC_SHA
TLS_DHE_DSS_WITH_AES_256_CBC_SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
SSL_RSA_WITH_DES_CBC_SHA
SSL_DHE_RSA_WITH_DES_CBC_SHA
SSL_DHE_DSS_WITH_DES_CBC_SHA
SSL_RSA_EXPORT_WITH_RC4_40_MD5
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
SSL_RSA_WITH_NULL_MD5
SSL_RSA_WITH_NULL_SHA
SSL_DH_anon_WITH_RC4_128_MD5
TLS_DH_anon_WITH_AES_128_CBC_SHA
TLS_DH_anon_WITH_AES_256_CBC_SHA
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
SSL_DH_anon_WITH_DES_CBC_SHA
SSL_DH_anon_EXPORT_WITH_RC4_40_MD5
SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA
TLS_KRB5_WITH_RC4_128_SHA
TLS_KRB5_WITH_RC4_128_MD5
TLS_KRB5_WITH_3DES_EDE_CBC_SHA
TLS_KRB5_WITH_3DES_EDE_CBC_MD5
TLS_KRB5_WITH_DES_CBC_SHA
TLS_KRB5_WITH_DES_CBC_MD5
TLS_KRB5_EXPORT_WITH_RC4_40_SHA
TLS_KRB5_EXPORT_WITH_RC4_40_MD5
TLS_KRB5_EXPORT_WITH_DES_CBC_40_SHA
TLS_KRB5_EXPORT_WITH_DES_CBC_40_MD5

Java JDK Supported Protocols

SSLv2Hello
SSLv3
TLSv1

Handshake

```
public class ClientSocket implements HandshakeCompletedListener {
	public void handshakeCompleted(HandshakeCompletedEvent event) {
		Socket socket = event.getSocket();
		try {
			InputStream in = socket.getInputStream();
			OutputStream out = socket.getOutputStream();
		// do work here
			in.close();
			out.close();
		} catch (IOException error) { error.printStackTrace(); }
	}

	public void connect() throws UnknownHostException, IOException {
		int port = 443;
		String hostname = "www.sdsu.edu";
		SocketFactory sslSocketFactory = SSLSocketFactory.getDefault();
		SSLSocket socket = (SSLSocket) sslSocketFactory.createSocket(hostname,
port);
		socket.addHandshakeCompletedListener(this);
		socket.startHandshake();
	}
}
```

Session Management

SSL will reuse session keys for multiple connections

- Between same client & server
- Connections made in short time span

Feature can be turned off

Client Mode

setUseClientMode(boolean)

Set client mode to false

Client will try to authenticate itself

setNeedClientAuth(boolean)

Require all clients connecting to server to authenticate themselfs

Server-side SSLSocket

Need to

- Generate public keys & certificates
- Create SSLContext for the algorithm you will use
- Create TrustedManagerFactory for source of certificate material
- Create KeyManagerFactory for type of key material
- Create KeyStore object for key & certificate database
- Fill KeyStore object
- Initialize TrustedManagerFactory & KeyManagerFactory

Then just use the socket

Sending Email
Java

SMTP

Simple Mail Transport Protocol

Common Ports

- 25 Official
- 587 authentication

Rohan

- 25 - no authentication, from campus machines only
- 587 - authentication with username & password
- send mail to rohan users only - no authentication neeed

You need to change

address

host

port

in following examples

Java

Required API

Java Mail

http://java.sun.com/products/javamail/index.jsp

JAF (JavaBeans Activation Framework)

http://java.sun.com/javase/technologies/desktop/javabeans/glasgow/jaf.html

Java Example - no password

```
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

public class SampleEmail {
        public static void main(String[] args) {
                String to = "whitney@rohan.sdsu.edu";
                String from = "whitney@cs.sdsu.edu";
                String host = "cs.sdsu.edu";
                Properties mailSettings = new Properties();
                mailSettings.put("mail.smtp.host", host);
                mailSettings.put("mail.smtp.port", "8025");
                Session session = Session.getDefaultInstance(mailSettings);
                try {
                        MimeMessage msg = new MimeMessage(session);
                        msg.setFrom(new InternetAddress(from));
                        InternetAddress[] address = {new InternetAddress(to)};
                        msg.setRecipients(Message.RecipientType.TO, address);
                        msg.setSubject("Mail Example");
                        msg.setSentDate(new Date());
                        msg.setText("Hello world");
                        Transport.send(msg);
                }
                catch (Exception mailError){
                        mailError.printStackTrace();
                }}}
```

Don't forget to change

to

from

host

port

Java with password

```
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

public class SampleEmail {
        public static void main(String[] args) {
                String to = "whitney@rohan.sdsu.edu";
                String from = "whitney@rohan.sdsu.edu";
                String host = "rohan.sdsu.edu";
                Properties mailSettings = new Properties();
                Session session = Session.getDefaultInstance(mailSettings);
                try {
                        MimeMessage msg = new MimeMessage(session);
                        msg.setFrom(new InternetAddress(from));
                        InternetAddress[] address = {new InternetAddress(to)};
                        msg.setRecipients(Message.RecipientType.TO, address);
                        msg.setSubject("Mail Example");
                        msg.setSentDate(new Date());
                        msg.setText("Hello world");
                        Transport withPassword = session.getTransport("smtp");
                        withPassword.connect(host, 587, "whitney", "password");
                        msg.saveChanges();
                        withPassword.sendMessage(msg, msg.getAllRecipients());
                        withPassword.close();
                        }
                catch (Exception mailError){ mailError.printStackTrace(); } }
}
```

Java - Debugging

```
Session session = Session.getDefaultInstance(mailSettings);
session.setDebug(true);
```

prints out server conversation

Sample Session

client **EHLO Al.local**
250-rohan.sdsu.edu Hello ip68-7-92-191.sd.sd.cox.net [68.7.92.191], pleased to meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-AUTH LOGIN PLAIN
250-DELIVERBY
250 HELP
client **MAIL FROM:<whitney@rohan.sdsu.edu>**
250 2.1.0 <whitney@rohan.sdsu.edu>... Sender ok
client **RCPT TO:<whitney@rohan.sdsu.edu>**
250 2.1.5 <whitney@rohan.sdsu.edu>... Recipient ok
client **DATA**
354 Enter mail, end with "." on a line by itself
client **Message-ID: <849515.01144947917509.JavaMail.whitney@Al.local>**
| **Date: Thu, 13 Apr 2006 10:05:16 -0700 (PDT)**
| **From: whitney@rohan.sdsu.edu**
| **To: whitney@rohan.sdsu.edu**
| **Subject: Mail Example**
| **MIME-Version: 1.0**
| **Content-Type: text/plain; charset=us-ascii**
| **Content-Transfer-Encoding: 7bit**
|
| **Hello world**
client **.**
250 2.0.0 k3DH4MUP027301 Message accepted for delivery
client **QUIT**