CS 255: Cryptography and Computer Security, Winter 2000

Programming Project #1

Due: Monday, February 26th, 2001, 11:59pm

Overview

For the programming projects in this course, you will be working in groups of two. Your goal is to increase the security of the provided Vote system. Right now, the Vote system is totally insecure: the votes and vote results are transmitted in the clear.

For project 1, the required security features are:

- A key exchange between the vote client and the vote server when the client connects.
- All network traffic, including vote information, votes and receipts, needs to be encrypted using a block cipher in CBC mode.
- A MAC must be appended to all messages.
- You must devise and implement a measure to combat replay attacks.

We will examine each of these features in detail below.

Key Exchange

You may use any of the key exchanges discussed in class, including the Diffie-Hellman key exchange protocol or any scheme that uses public key encryption (we showed in class that a secure public key system implies a key exchange algorithm). Please no Merkle Puzzles. Note that we have not yet discussed a method to prevent the "person-in-the-middle" attack on key exchange protocols. Consequently, in this project you are only required to secure the key exchange protocol against a passive eavesdropper. We will deal with active attacks on the key exchange protocol in project 2.

You should use the secret obtained during key exchange to generate your block cipher key, your CBC IV and your MAC key. To do this, you should use a hash of the secret as input to a PRNG, and use the output of the generator as your keys and IV.

Message Encryption

Each message transmitted by either a client or the server must be encrypted using a block cipher. You may use any standard block cipher you like, but all messages must be encrypted using CBC mode. As mentioned above, you should generate both the cipher key and the CBC IV from the secret obtained during key exchange.

MACs

Every message that goes over the network should have a MAC to enable the detection of a malicious attacker tampering with messages en route. You should generate the key for this MAC from the secret obtained during key exchange as described above.

Preventing Replay Attacks

Even after you secure votes with encryption and MACs, there is still an obvious replay attack: Marvin captures a vote message en route from the client to the server. He then repeatedly sends that message—which is still a valid vote which will cause the vote tally to reflect this flood of increased votes and not the correct outcome. Your solution should prevent an attacker from replaying a vote.

Security Holes

Even with these four security features, it should be clear that there are still holes in the Vote system. Aside from the "person-in-the-middle" attack on the key exchange protocol (mentioned earlier), the biggest remaining problem is authentication. Anyone can vote, whether or not they are "registered" to do so. In addition, the vote server has no way to identify voters to ensure that the same person does not vote multiple times. Even worse, the vote server sees all user votes - no anonymous voting. We will address many of these problems in programming project 2.

Implementation

As mentioned in class, you will be programming this project in Java. We have provided a substantial amount of starter code. Here is a description of the files we provide for you (files you need to change are in **bold**):

File	Purpose
Makefile	Makefile for the project. Modify this to build any classes you add to the project.
java.policy	Policy file granting network/file permissions. This is necessary for Java to run the Vote system.
Vote/VoteClient.java	Vote Client
Vote/Voter.java	Voter remote interface. This defines methods that may be called over the network by the server.
Vote/VoterImpl.java	Implementation of the Voter interface.
Vote/VoteLoginPanel.java	GUI class for the login screen.
Vote/VoteChoicePanel.java	GUI class for the voting screen.
Vote/VoteReceiptPanel.java	GUI class for the vote receipt screen.
Vote/VoteServer.java	Vote Server remote interface. This defines server methods that may be called over the network by clients.
Vote/VoteServerImpl.java	Implementation of the VoteServer interface.
Vote/SkipConstants.java	Contains constants for the 1024 bit prime p and generator g used for SKIP (Simple Key-Management for Internet Protocols). Use these if you don't have the patience to generate your own.

You should spend some time getting familiar with the provided framework. You will need to copy the `/usr/class/cs255/proj1` directory to your account. You will also need to run `source /usr/class/cs255/setup.csh` to set your path, classpath and java alias correctly (if you are an advanced user, you may want to do this manually.) To build the project, change into the `proj1` directory and type `make`. To run the Vote system, follow these four easy steps:

1. Pick a unique number for your group to use (between 1024 and 65535). We recommend using the last four digits of your phone number. Note this won't work if your last four digits are less than 1024 or your roommate is taking the class.
2. Start the RMI registry. This is part of the networking infrastructure the project uses. To start the registry, type

```
elaine19:~/proj1> hup rmiregistry portNum &
```

If you omit the port number, the registry will start on port 1099, so be careful.

3. Start the Vote Server. Note that the server does not necessarily need to be running on the same machine as the registry. You do, however, have to specify the host and port on which the registry is running. For example:

```
elaine19:~/proj1> java Vote.VoteServerImpl elaine19 portNum &
```

4. Start one or more Vote Clients.

```
elaine19:~/proj1> java Vote.VoteClient &
```

Important note: In the past, we have received complaints from the administrators of the Sweet Hall cluster about CS 255 students who leave the `rmiregistry` and VoteServer processes running after they log out. The `hup` command used to start the rmiregistry above instructs the shell to stop it on exit, but we recommend you explicitly make sure you kill your server processes before logging out anyway.

Crypto Libraries and Documentation

Java's security and cryptography classes are divided into two main packages, `java.security.*` and `javax.crypto.*`. The reason for this division is United States export restrictions. The `javax.crypto.*` package is collectively referred to as the Java Cryptography Extension (JCE). Classes for cryptographic hashing and digital signatures (not required for project #1) can be found in `java.security`, part of the standard JDK, whereas ciphers and MACs are located in the JCE. There is an API user's guide for the JCE that is not published on Sun's site: Although U.S. export restrictions were recently reduced, and all the packages we use in this project (including the JCE itself) are exportable, this documentation file (according to Sun) "contains technical information related to encryption technology that is not approved for world-wide export." We have reproduced this document for educational purposes only, and we hope you do not choose to violate any laws. That being said, you should probably read through the guide first. It can be found at:

href="http://www.stanford.edu/class/cs255/doc/jce/guide/API_users_guide.html

You may also find the following sites useful documentation (we have created local copies of the API documentation on the class Web site for convenience):

Description	URL
Java API	http://www.stanford.edu/class/cs255/doc/jdk/api/
JCE API	http://www.stanford.edu/class/cs255/doc/jce/apidoc/
Java Tutorial	http://java.sun.com/docs/books/tutorial/
Chapter 6 from *Java Cryptography*, by Jonathan Knudsen	http://www.oreilly.com/catalog/javacrypt/chapter/ch06.html

Some classes you may want to take a look at:

```
javax.crypto.KeyAgreement
javax.crypto.KeyGenerator
javax.crypto.IvParameterSpec
javax.crypto.Mac
javax.crypto.Cipher
javax.crypto.SecretKeyFactory

java.security.MessageDigest java.security.KeyPairGenerator java.security.SecureRandom
java.math.BigInteger
```

A Word About Networking

The Vote System runs over the network, as it would be rather boring to construct an electronic voting scheme that was limited to one computer. However, we do not expect everyone interested in cryptography to have network programming experience. For that reason, our programming projects will use Java RMI for all network communications.

RMI stands for Remote Method Invocation, and it is a standard part of the Java library. RMI enables you to declare objects as "remote", making the methods of that object accessible to Java programs running on different machines.

All of the RMI setup has already been done for you. For instance, in `VoteClient.connect`, you will find the following code:

```
_server = (VoteServer)Naming.lookup("//" + host + ":" +
                                    port + "/VoteServer");
```

This simply sets the variable `_server` to a reference to an object called "VoteServer" which has been registered with the RMI registry (the `rmiregistry` program) running on the designated host and port. In `VoteClient.sendVote`, you see the code:

```
_server.postVote(voteNum, choice, _clientID);
```

This has the effect of sending `voteNum`, `choice` and `_clientID` over the wire to the machine where the VoteServer is running, and calling the server's `postVote` method with `voteNum`, `choice` and `clientID` as parameters.

If you want to change the parameters to `postMessage` (we hope you do!) or add a new remote method, you need to do so in two places: You ened to modify the `VoteServer` remote interface in `VoteServer.java`, and the actual method definition in `VoteServerImpl.java`. The same is true for the `Voter` class.

Help

- We strongly encourage you do use the class newsgroup, `su.class.cs255`, as your first line of defense for the programming projects. TAs will be monitoring the newsgroup daily, and your question may have already been answered there.
- In addition to the newsgroup, we will be maintaining a Programming Assignment FAQ on the course Web page. Be sure to check it frequently.
- We will be moving some of our office hours to Sweet Hall to help you with programming questions. Check the Web page for up-to-the-minute office hour locations.
- If all other avenues have been exhausted, you can email the course staff at cs255ta@cs.stanford.edu.

Submission

In addition to your well-decomposed, well-commented solution to the assignment, you should submit a README containing the names, Leland usernames and Stanford ID numbers of the people in your group as well as a description of the design choices you made in implementing each of the required security features.

When you are ready to submit, make sure you are in your `proj1` directory and type `/usr/class/cs255/bin/submit`.

Last modified: Fri Feb 9 07:26:50 PST 2001