Computer Science 455/555
Identity Server
Programming Assignment 2 (100 points)

1 Description

In this assignment, we will implement a RPC/RMI based identity server. The basic idea is that client can connect to this server and submit a new login name request. The server checks its database and responds back either with a Universally Unique ID (UUID) if the new login id hasn't been taken by anyone before or returns back with an error. As part of the request to create a new login name, the client must submit a real user name as well. The server also permits reverse lookups, where a client submits a UUID and ask for the login name and other information associated with that UUID. The server periodically saves its state on to the disk so that it can survive crashes and shutdowns.

2 Motivation

In Linux/Unix systems, the user name is associated with an integer known as user id or `uid`. The integer `uid` is the only value that is actually stored in the file system. The mapping from user name to user id is stored in the file `/etc/passwd`. The nice thing about this is that we can change a user's name anytime without having to change the whole file system (as long as we don't change the associated user id number). This scheme breaks down across several systems since the same user may not have the same user ids on different systems. There are several solutions for maintaining this information in a centralized manner across multiple Linux/Unix machines: Kerberos, Network Information Services and OpenLDAP.

On the other hand, Microsoft Windows has the Active Directory services which plays the same role. Because LDAP-based authentication is supported on the most recent Microsoft systems, including Windows 2000 and XP, and is also supported on Linux and other Unix systems, it makes an excellent choice for a cross-platform authentication system. Note that there are limitations to this. Firstly, the Microsoft clients for Windows 2000 and XP are specific to authenticating against a Microsoft Active Directory server. Although OpenLDAP uses the same LDAP protocol, there are other features of Active Directory (including a modified version of Kerberos with a Microsoft specific mechanism called a "PAC" which means that Active Directory clients will not necessarily be able to authenticate against OpenLDAP.

The second limitation of Active Directory is that clients for it (in LDAP mode, that is) are only available on Windows 2000 and Windows XP. Although Active Directory will work in a "down-level" mode to support older Microsoft clients, users of systems such as Windows NT and Windows ME/98/95 who wish to have full LDAP/Kerberos based authentication

support are forced to upgrade...

So this can be done for multiple platforms but it is not that simple. Our Java based solution will work simply across all platforms that Java runs on, which is virtually everything. Of course, we are implementing a very limited subset of the actual authentication problem.

3 Specifications

3.1 Server

The following lists the salient features of the server.

- The server must be implemented using Remote Method Invocations (or Remote Procedure Calls).
- The server presents a suitable RPC/RMI interface for clients to access the id functions. The server must export remote methods that allow a client to create a new login name, to lookup a login name, to reverse lookup a UUID as well as to modify an existing login name.
- The server maintains a database of login name to UUID mappings in memory. For each newly created login name, the server creates a UUID (use the `java.util.UUID` class for this purpose) and stores it in the mappings database. For each login name, we also want to store the IP address from where the request was sent, the date and time when the request was received and the real user name associated with the login name. Optionally, the server can also store the last change date for each id.
- You may assume that the mappings database is small enough to fit in memory. You must use an efficient data structure for the mappings database.
- Note that a separate thread dispatches every RMI call. The implication for RMI objects is they must be thread safe, since the object may have to handle multiple requests simultaneously. The good thing about this is that your RMI server is already multi-threaded.
- Periodically, the server checkpoints the mapping data structure to the disk. Use Java serialization to accomplish this checkpointing. This makes server resistant to crashes and restarts.

3.2 Client

The client program can be command-line based or GUI based. It should provide for at least the following operations:

- The class containing the main method must be named *IdClient*. The arguments are:

  `java IdClient <serverhost> <query>`

- It must support at least six types of command line queries as follows:

  `--create <loginname> [<real name>] [--password <password>]` With this option, the client contacts the server and attempts to create the new login name. The client optionally provides the real user name along with the request. In Java, we can merely pass the `user.name` property as the user's real name. Use `System.getProperty("user.name")` to obtain this information. If successful, it returns the UUID (or an Account object from the server that the client can then cache if it so desires). Otherwise it returns an error message.

  `--lookup <loginname>` With this option, the client connects with the server and looks up the `loginname` and displays all information found associated with the login name.

  `--reverse-lookup <UUID>` With this option, the client connects with the server and looks up the `UUID` and displays all information found associated with the UUID.

  `--modify <oldloginname> <newloginname> [--password <password>]` The client contacts the server and requests a loginname change. If the new login name is available, the server changes the name (note that the UUID does not ever change, once it has been assigned). If the new login name is taken, then the server returns an error.

  `--get users|uuids|all` The client contacts the server and obtains either a list all login names, lits of all UUIDs or a list of user, UUID and string description all accounts (don't show encrypted passwords in this option).

- The above options can be abbreviated as –c, –l, –r, –m and –g. Note that we can supply only one query at a time.

4 Enhanced Security

4.1 Encrypted Passwords and Encrypted Communications (10 points)

4.1.1 Encrypted passwords

Add handling passwords to the server's functionality. So now, a client specifies a password when they create a new login name. To lookup information, no password is required. To modify the login name, the password is required.

The password should be encrypted by the client before sending to the server. The server should store only the encrypted password in the account database (both in memory and on disk).

Use Java's cryptography extensions. See the package `javax.crypto` and `java.security` as well as other related classes. In particular, look at the classes `javax.crypto.KeyGenerator`, `javax.crypto.Cipher` and `java.security.Key` classes.

4.1.2 Encrypted transmissions

This part requires that the previous part on encrypted passwords be completed first. Now we want to also encrypt all communications over the network. Use RMI over SSL (Secure Sockets Layer) to accomplish this part. With this approach the password can be sent in the clear from the client to the server. The server then encrypts the password and uses it as well as stores it in the encrypted form. This would be lead to more secure password handling since the server doesn't have to protect the encrypted password anymore.

5 Hints on Rmiregistry

You will need to run the rmiregistry in order to use RMI. By default, the rmiregistry service runs on port 1099. This isn't a problem at home. However, in the lab, if more than one student runs rmiregistry, the one started later will fail. You can run multiple rmiregistry daemons on the same system by giving them a different port number. For example:

```
rmiregistry 5113 &
```

starts it listening to port 5113. Now in your call to `Naming.bind()` or `Naming.rebind()`, you must specify the port number. Suppose the name of your server is NiftyIdServer, then you will rebind as follows.

```
MyServer server = new MyServer();
Naming.rebind("//localhost:5113/NiftyIdServer", server);
```

6 Documentation

- Please include a README file that documents clearly how to setup and test your project. Describe clearly how much of the functionality is available. If you did any extra stuff, please document that clearly as
- Use `javadoc` to document your code.

7 Submitting the Assignment

Copy over all the code (and corresponding makefiles, `README` files etc) to a directory on `onyx`, then change your current directory to that directory and invoke submit as follows:

submit amit cs555 2