CS 6500 – Network Security
Prof. Krishna Sivalingam
July-Nov. 2017
Lab 3: Kerberos Authentication Dialog
Due Date: Sunday, **Oct. 8, 11PM**. On-Line Submission via moodle
NO LATE SUBMISSIONS

1 Project description

The objective of this project is to implement the different steps of the Kerberosv5 authentication dialogue. This requires prior knowledge of socket programming. You may use C/C++, Python or Java.

1.1 Networked Systems

The emulated Kerberos system will contain: (a) one Authentication Server instance (AS) (port 6583); (b) one Ticket Granting Server (TGS) instance (port 8471); and (c) $3 \leq N \leq 9$ number of fileserver instances, on ports numbered from 8681 through 8689, as needed. There will be a total of 10 unique user identifiers, denoted by `user1` through `user10`.

All the servers and the client process will run on the same machine, in different processes, on different port numbers are listed above.

Name your project directory as LAB3 (Note: ALL UPPERCASE). The directory structure under the LAB3 directory is as follows:

- ▷ `AS`, to store all AS related keys
- ▷ `TGS`, to store all TGS related keys
- ▷ `user1`, `user2` up to `user10`, to store all session keys, secret key and data files
- ▷ `FS1`, `FS2` up to `FS9`, to store all session keys, secret key and data files

Each subdirectory, `user1` through `user10`, will contain files that the corresponding user can access. This represents the current working space of that user. There will be one `key` file, that contains the user's K_c secret key (128-bit string, base64 encoded) – assume that this is stored unencrypted. The data files are stored in plaintext form. The $K_{c,tgs}$ values will be stored in files `key_user_i_tgs.txt` base64 format, where i denotes the user ID respectively.

The $K_{c,v}$ values will be stored in files `key_user_i_fs_j.txt` base64 format, where i and j denote the user ID and fileserver ID respectively.

Under each fileserver directory (`FSi`), there will be one subdirectory for each user, `user1` through `user10`. Each subdirectory will contain files that the corresponding user can access. These files can be created ahead of time using standard Unix commands. The data files are stored in plaintext form.

Each fileserver will also have its own `key` file in its subdirectory, that contains the server's K_v value (128-bit string, base64 encoded) – assume that this is stored unencrypted, and the files for the $K_{c,v}$ secret keys.

Assume that all the session keys ($K_{c,tgs}$ and $K_{c,v}$) generated by AS and TGS have a lifetime of three minutes.

1.2 AS

The K_c and K_{tgs} secret keys will be stored in an `AES_128_CBC` encrypted file, called `keys_tgs.txt`, that is accessible by the AS. These secret keys are 128-bit long. You will invoke AS as:

```
% ./as <password> &
```

The AS will be an infinite loop; the only task of the AS is to authenticate the specified user, generate the corresponding ticket for TGS and send it to the user as per the Kerberosv5 dialogue.

1.3 TGS

The K_v (secret key between each fileserver and TGS) keys and K_{tgs} secret keys will be stored in an AES-128-ECB encrypted file, called `keys_tgs.txt` that is accessible by the TGS. These secret keys are 128-bit long. You will invoke TGS as:

```
% ./tgs <password> &
```

The TGS will be an infinite loop; the only task of the TGS is to receive the TGT from the specified user, generate the corresponding ticket for the FS and send it to the user as per the Kerberosv5 dialogue. The $K_{c,tgs}$ files will be stored in the TGS directory files `key_user_i_tgs.txt` base64 format, where i denotes the user ID respectively.

1.4 FS

You will invoke the file servers as:

```
% ./fs <id> &
```

Here, id is 1 through 9. The file server instance will listen on its assigned port (see above). The FS will be an infinite loop, waiting for user commands.

When a fileserver receives a `get filename` message from a user, it first checks if the given file exists; if it does not, it prints an error message. If the file exists, it checks if there is a valid session key ($K_{c,v}$) for the client. If there is no key established, then it prints an appropriate error message. If there is a key established, then it sends an AES-128-ECB plus base64 encrypted version of the data file, using the secret key ($K_{c,v}$).

A similar operation is done for `put filename` message.

2 Client interaction

The main program will be invoked as follows:

```
% ./ker5 -N <fileservers>
```

This will first create the AS, TGS and FS instances (using appropriate “system” commands). It will then wait in a command-line loop:

```
ker5>
```

The commands are:

▷ `get user<k> fileserver<f> <filename>`

▷ `put user<k> fileserver<f> <filename>`

▷ `exit`

For the `get` command, the user k will receive an encrypted file from the fileserver f. This file should be stored locally as filename.enc; the output of the decryption using the corresponding $K_{c,v}$ will be then done and the plaintext file stored in the local directory.

A similar sequence involving the FS and the user is to be done for the `put` command.

For the `exit` command, the main process should shut down all the AS, TGS and FS processes in a safe manner and exit.

3 What to Submit

Name your project directory as LAB3 (Note: ALL UPPERCASE). Once you are ready to submit, change directory to the directory above LAB3, and tar all files in the directory with the command:

```
tar czf Lab3-RollNo.tgz LAB3
```

The directory should contain the following files:

▷ Source Files

▷ Makefile
Typing command 'make' at the UNIX command prompt, should generate all the required executables.

▷ a README file containing instructions to compile, run and test your program. The README should document known error cases and weaknesses with the program.

▷ a COMMENTS file which describes your experience with the project, suggestions for change, and anything else you may wish to say regarding this project. This is your opportunity for feedback, and will be very helpful.

4 Help

1. WARNING ABOUT ACADEMIC DISHONESTY: Do not share or discuss your work with anyone else. The work YOU submit SHOULD be the result of YOUR efforts. Any violation of this policy will result in an automatic ZERO on the project, a potential F in the course, and other academic action.

2. Ask questions EARLY. Do not wait until the week before. This project is quite time-consuming.

3. Implement the solutions, step by step. Trying to write the entire program in one shot, and compiling the program will lead to frustration, more than anything else.

5 Grading

- ▷ AS: 20 points
- ▷ TGS: 20 points
- ▷ FS: 40 points
- ▷ User Interaction: 20 points

No README/COMMENTS: -5 points; No Script File: -10 points; Incomplete Compilation: -10 points