Project - Secure File Transfer

Objective

The objective of this project is to introduce you to cryptographic tools via the implementation of a secure file transfer protocol. Your implementation will be similar to and a subset of a very popular tool used in Apple devices, called AirDrop. During the process, you will practice many cryptographic and cybersecurity concepts such as symmetric and asymmetric cryptography, digital certificates, public key infrastructure, mutual authentication, non-repudiation, confidentiality, integrity protection and password security.

Project Team & Milestones

This project is designed to challenge you and practice critical reasoning. To simplify the implementation — (1) you are required to work in a team, and (2) the project is broken down into milestones.

Programming Language

The recommended programming language is Python v3. The reason for choosing Python is because it is one of the easiest programming languages to pick up. Python also has very good support for popular cryptographic libraries such as `PyCrypto` and `cryptography`. Your team is free to choose another programming language (provided all members are willing), however, remember that this is a complex project and you'd need a language that enables fast prototyping and instant feedback.

Team Size

The recommended team size is three students. A consistent team size would make it easier for me to be fair in my grading. Once you have decided a team, please send me an email with the team member names. Only one email per team is required. Make sure that you copy the other members to the email. Remember, you cannot change your team mid-semester so choose wisely.

Project Milestones

The project may appear daunting to some students, especially, if you haven't implemented a large project before. In order to mitigate this problem, the project is broken down into milestones where each milestone can be thought of as a smaller and simpler sub-project. Try completing these milestones in order and the implementation should appear to be much easier. For each milestone, the grading will be a 50/50 split, i.e., you will get 50% of the total points for secure design and 50% of the total points for correct implementation.

Overview

Secure file transfer can be useful in many different scenarios, e.g., uploading source code, transmitting configuration files, transmitting sensitive banking information. For this project, we'll focus on one scenario — how can we securely transfer a file to another person's computer who is in our contact list and on the same local network (wired or wireless)?

The project will be implemented as a command-line tool. We'll call the application `SecureDrop` and the executable `secure_drop`. For the sake of simplicity, the focus will be on the Linux ecosystem. The instructor will also provide Docker containers for testing the application. A sample run of the project is shown below, and the milestones are outlined next. *Note that the lines starting with # are comments that describe what happens in each command.*

```
# The secure_drop command is the entry point of the application. If a user does not exist,
# the registration module will activate. The email address is used as the user identifier.
$ secure_drop
 No users are registered with this client.
 Do you want to register a new user (y/n)? y
```

```
 Enter Full Name: Sashank Narain
 Enter Email Address: sashank_narain@uml.edu
 Enter Password: *********
 Re-enter Password: *********

 Passwords Match.
 User Registered.
 Exiting SecureDrop.
$

# The user registration is a one-time process. Once a user is registered on a client, the
# login module is activated subsequently. After a successful login, a "secure_drop>" shell
# is started.
$ secure_drop
 Enter Email Address: sashank_narain@uml.edu
 Enter Password: ****
 Email and Password Combination Invalid.

 Enter Email Address: sashank_narain@uml.edu
 Enter Password: *********
 Welcome to SecureDrop.
 Type "help" For Commands.

 secure_drop> help
   "add"  -> Add a new contact
   "list" -> List all online contacts
   "send" -> Transfer file to contact
   "exit" -> Exit SecureDrop
 secure_drop>

# The "add" command adds a new contact for the user. If a contact exists, it overwrites
# the existing details. Note that the email address is used as the user identifier.
 secure_drop> add
   Enter Full Name: John Doe
   Enter Email Address: john.doe@gmail.com
   Contact Added.
 secure_drop> add
   Enter Full Name: Tom Bloggs
   Enter Email Address: tom.bloggs@gmail.com
   Contact Added.
 secure_drop>

# The "list" command should show only those contacts that satisfy the following conditions -
# 1. The contact information has been added to this user's contacts.
# 2. The contact has also added this user's information to their contacts.
# 3. The contact is online on the user's local network.
  secure_drop> list
    The following contacts are online:
    * John Doe <john.doe@gmail.com>
  secure_drop>

# The "send" command transfers a file to the contact. Note that the contact must receive the
# following alert and they must approve the transfer. You can save the file to a directory
# of your choice with the same file name as the transmitted file.
```

```
#   Contact 'Sashank Narain <sashank_narain@uml.edu>' is sending a file. Accept (y/n)?
  secure_drop> send john.doe@gmail.com /home/sashank/Documents/john.zip
    Contact has accepted the transfer request.
    File has been successfully transferred.
  secure_drop> exit
$
```

Milestone 1 - User Registration

Keep in mind the following points when implementing the module -

1. You are not expected to implement databases for storing the information, and simple files / JSON structures should work.
2. Implement the module first without any security controls. Once the module has been tested to work correctly, implement the security controls. Focus on simple controls that can protect against traditional password cracking attacks.
3. Write reusable code. The password security implementation can also be reused for the User Login milestone.
4. Use third party APIs. The Python `crypt` module can be very helpful if you plan to implement salted hashes for password security.
5. Think about what information will be required for mutual authentication in the next milestones. Generate and save the information during the registration process. If you plan to use digital certificates for mutual authentication, you can assume that a CA is present on all the clients. Generate a CA manually for testing purposes.

Milestone 2 - User Login

This is a simple milestone that can be implemented quickly. Keep in mind the following points when implementing the module -

1. Write reusable code. Much of the code will be similar to the User Registration milestone. It will save you a lot of time.
2. Think about how you can use information from the login credentials to enhance security in the next milestones. Generate and keep the information in memory. *Remember that keeping the password in memory is not recommended. All other information must be forgotten when the user exits the program.*

Milestone 3 - Adding Contacts

This is also a simple module that can be implemented quickly. Keep in mind the following points when implementing the module -

1. You are not expected to implement databases for storing the information, and simple files / JSON structures should work. You can assume that each user will have limited number of contacts and there is no need to develop algorithms and data structure to efficiently store and retrieve contacts.
2. Think about how you can use the information generated in milestone 2 to ensure the confidentiality and integrity of the contact information. An attacker who gains access to your client should not be able to access the contact information. Any changes made by the attacker should get automatically detected. *A compromise of contact information can easily lead to spam and targeted attacks on those contacts.*

Milestone 4 - Listing Contacts

This is a tricky milestone that will require careful design and implementation. This is the first time the application communicates over the network. As such, the applications's attack surface is significantly increased. Keep in mind the following points when implementing the module -

1. Remember that a contact information should only display if the user has added them as a contact, the contact has also added the user as their contact, and the contact is online on the same network.

2. Implement the module first without any security controls. Once the module has been tested to work correctly, implement the security controls. Do not implement the transport layer protocols and use TCP or UDP. The Python `socket` module contains easy-to-use implementations of transport layer protocols.
3. Write reusable code, at least, for code that performs cryptographic operations. The Python `PyCrypto` and `cryptography` modules can be very helpful in implementing the many different cryptographic operations. These implementations can be reused for the Secure File Transfer milestone.
4. Ensure that the identities of the communicating entities are encrypted. An attacker sniffing packets on your network should not be able to obtain the email addresses of the contacts. As mentioned earlier, a compromise of the contact information can easily lead to spam and targeted attacks on your contacts.
5. Think about how you can mitigate impersonation attacks. The attacker can pose both as a sender or a receiver in the communication. This threat can be avoided by implementing a protocol that performs mutual authentication between the communicating entities.
6. Think about any information that can be exchanged between the entities to enhance the security of the next step, and help avoid re-perfoming the mutual authentication. This information must be unique to the communication and protected from the attacker. Compromise of the information should not result in a compromise of the file transfer.

Milestone 5 - Secure File Transfer

The complexity of this milestone depends on the design of the previous milestone. Keep in mind the following points when implementing the module -

1. Think about how you can efficiently transmit large files without compromising the confidentiality and integrity of the file. The received file must be exactly the same as the transmitted file. The system must ensure this before notifying the user of successful transfer.
2. Think about how you can mitigate replay attacks. The attacker who has recorded a previous communication must not be able to replay the communication to send / receive files. This threat can typically be avoided by using incremental sequence numbers starting with a random seed on every client.

Grading

- 20 points - Code Quality (Readability, Modularity, Documentation, Error Handling)
- 20 points - Successfully implemented Milestone 1 (10 points for security, 10 for implementation)
- 10 points - Successfully implemented Milestone 2 (5 points for security, 5 for implementation)
- 10 points - Successfully implemented Milestone 3 (5 points for security, 5 for implementation)
- 20 points - Successfully implemented Milestone 4 (10 points for security, 10 for implementation)
- 20 points - Successfully implemented Milestone 5 (10 points for security, 10 for implementation)