Computer Security Project: Secure(?) Chat

Due: Monday, October 21st @ 11:59pm

Synopsis

Write a chat program in C that provides:

- Authentication of correspondents
- Message secrecy (encryption)
- Message integrity (MACs)

Given that this program processes formatted input from a network, you should naturally focus on software security as well.

Goals for the student

- Learn to effectively use off-the-shelf cryptographic libraries (`openssl`).
- Understand issues in protocol design and network programming.
- How to avoid or mitigate common software security issues.

Important Notes

If you'd like, feel free to collaborate in small groups (≤ 3 members). If you do collaborate in a group, please **use git**. This ought to help you organize, but it will also be useful for me to make sure everyone was contributing to the project. If you have not collaborated with git much, I have some maybe helpful notes here.

Details

I've given you a skeleton which does very basic chat stuff: Depending on the invocation, it will listen for connections, or make one with another host. Beyond that, it just sends and receives text, displaying each message in a log window. It will be up to you to:

- Write some sort of handshake protocol to setup ephemeral keys (your protocol should have perfect forward secrecy!).
- Mutual authentication, using public key cryptography.
- After authentication, each message should be encrypted and tagged with a message authentication code. You may also want to take measures to prevent replay attacks.

I think SSH will be a good model on which to base your protocol. In particular, don't use PKI (public-key infrastructure, with certificates and such), and instead assume that communicating parties have already exchanged public keys. However, implementing deniable authentication would be a nice touch (and is something SSH does not provide). If you want to use 3DH, you can find an example in `dh-example.c`.

Compiling the skeleton

You will need:

- gtk3 and the header files. If you are on linux/BSD, you might have to get a package like `gtk+3-devel` or similar, although some distributions (e.g. Arch Linux) will include header files in the normal package (no `-devel` needed).
- openssl and headers (`openssl-devel`).
- gmp and its header files (`gmp-devel`).

Running `make` should just work on most linux or BSD systems if you have all the above installed, but let me know. I'm confident you could also get this working just fine on a mac via homebrew. You should be able to get it working on Windows as well, but it might be easier to just do it in a virtual machine. If you do get it working natively on Windows, I'd be interested, so please let me know what steps were needed.

Once you do have the skeleton compiled, you can run `./chat -h` and see a list of options. You should be able to test it out like this:

```
$ ./chat -l & sleep 1 && ./chat -c localhost &
```

Two windows should appear in a moment, connected over the loopback interface.

Other notes

There is a directory `openssl-examples` that demonstrates how to get most of the functionality you'll need from `openssl`. However, your professor decided to write his own Diffie-Hellman key exchange, as the openssl version was somehow even more obfuscated and confusing than usual. You can see the Diffie-Hellman stuff in files `dh.h`,`dh.c`, and you can see some example usage in `dh-example.c`. Note that the function `dhFinal(...)` will also do key derivation for you (transforming the Diffie-Hellman value into pseudorandom bits that you can use as keys for encryption and MACs).

You might also find the following links helpful.

- network programming guide
- If you ever need to manipulate `mpz_t` types, read `info gmp`. Alternatively, you can read the manual online.

Submission Procedure

Have one of your group members send me your repository. If you have it hosted somewhere, you can just send a link, but if you've done things on your own servers, just make me an archive like this:

```
$ cd /path/to/your/chat/../
$ tar -czf chat.tgz chat/
```

Importantly, there should be a `.git/` folder in there containing the commit history.