Final Project

Steven M. Bellovin
https://www.cs.columbia.edu/~smb

BY NC

Secure Messaging

The goal of the project is to build a secure messaging system, based on TLS and certificates.

Details

- One server that handles all requests
- Several client programs, each with specific functionality
- All communications via HTTPS
- A formal test process

The Server

- One server program that handles all requests
  - You do not need to worry about simultaneous requests
- Authenticates user by either password or client-side certificate
- Issues certificates on request, and stores them
- Stores incoming messages, and delivers them to the client on request
- May need to be split so that some functions are done in a sandbox

Client Programs

`getcert`		*Log in with username and password; the server generates and returns a certificate*
`changepw`		*Log in with username and password, and supply a new password*
`sendmsg`		*Log in with a client-side certificate, send a list of recipient names, and receive a list of certificates. Encrypt the message to those certificates, digitally sign it, and upload it.*
`recvmsg`		*Log in with a client-side certificate and receive a single encrypted message from the server; it is then deleted from the server. Verify the signature and display the message.*

Intermediate Deliverables

- System architecture
  - Crucial security decisions and their rationale
  - Sandboxing decisions
  - File layout and permissions
  - Other security-sensitive decisions
- Test plan
  - What you’re testing, and why
  - Tests must emulate attacks!

Graded mostly on effort, not absolute correctness!

Final Deliverables

- Source code
    - The client and server programs must be in C or C++
    - Test programs may be written in any language you choose
- Revised design and test plan documents
- Test mechanisms and scripts

Don’t!

In the interests of sanity and effort, there are things I do *not* want you to do.

- No GUIs—just simple command-line clients. Writing a GUI takes too much time
- Don’t multithread the server; assume there are no simultaneous requests (yes, this lets you avoid some security issues)
- Do not add other commands or functionality, except in your test mechanisms
- Don’t bother writing code to add users to the system—we’ll supply that list

OpenSSL

- OpenSSL has many, many library routines
- I’ll point to suggested ones

HTTPS

- I will describe a subset of HTTPS that will be used for all communications
- Why HTTPS? It already exists—but it's too complex, so I'll subset it
- You may, if you wish, do some debugging via your browser

Sandboxing

- We’ll be discussing it in a few weeks
- Briefly, it’s a way to restrict the abilities of a process, to limit the damage if it should be compromised
- How you separate the privileged and sandboxed sections is a crucial design decision

Teams

- Teams of approximately four people
- You’re welcome to form your own teams; email XXX@columbia.edu
- We’ll assign teams for everyone else—and we’ll try to match timezones
    - If you haven’t submitted the timezone form, please do so now
    - Feel free to adjust your timezone based on your normal sleep schedule…
- You’re free to organize your team however you choose

Storage

- You are *strongly* encouraged to use Github as a shared repository for your code
- For your documents, you can use Google Docs or (for LaTeX users) Github or overleaf.com

More Documents

- The promised documentation will be uploaded to the website
- Keep watching the entry for for the final project

Questions?