CS558 Course Project

- **If you choose to do a presentation project, please email me your preferences for 3 project topics no later than Nov. 1st, 2023 (Wed.), indicating your first, second, and third choices. Alternatively, if you wish to propose your own cybersecurity-related topic, please email me your chosen topic by Nov. 1st, 2023.**
- **If you choose to do a systems project or a K-12/community outreach project, please email me your project topic by Nov. 1st, 2023 (Wed.)**
- **If you choose to do a programming project, there is NO need to send me an email.**

Please note:

1. If you are a graduate student and would like to have this course count as a long programming course, you must choose the programming project.
2. The use of any AI tool in this project must be approved by the instructor.

Presentation Project

In-class presentation: Nov. 28 – Dec. 7

Presentation slides submission (brightspace): 11:59pm Dec. 7 (Thursday)

The presentation project needs to be completed individually. In the presentation project, you will give a presentation in the class on one of the topics given below. You will need to make PowerPoint slides for your presentation. Each presentation should last 15-20 minutes. **The content covered in your presentation should not have significant overlap with the material covered in this course.** The presentation will be scheduled between Nov. 28 and Dec.7 (You will receive an email from me at least two weeks prior to your scheduled presentation date).

1. Security/privacy issues related to generative AI (you can also demonstrate some of the generative AI tools during the presentation)
2. Identity Theft
3. Social Engineering
4. Some of the recent attacks/scams
5. DeepFake (you can also demonstrate some of the deepfake tools during the presentation)
6. Cyber ethics
7. Social media security/scams
8. Firewall
9. Cyberbullying
10. AI/ML-based security
11. Cybersecurity topics that are not covered in this course

Guideline (ppt slides)

- Motivation: E.g., Why role-based access control?
- Background: E.g., Syntax of xml
- Technical details
- Use examples/pictures/demonstrations to illustrate technical details.

Guideline (presentation)

- Present slowly and clearly.
- Do not directly read from the slides.
- Do not read from the presentation notes.
- Use examples to illustrate technical details.

Programming Project

No presentation

Submission deadline: 11:59pm, Dec. 7 (Thursday)

The programming project can be completed either individually or by a group of 2 students. You can use the existing implementations of RSA and AES/3DES algorithms such as those provided in java.security, openssl, etc. For this project, you can use C, C++, Java, or Python. You are not required to implement a graphical user interface. **If you choose to do the programming project on your own, you will get 10 points extra credits.**

In this project, you will implement an ***iterative*** secure banking system consisting of a bank server and multiple clients (i.e., the ATMs). Each bank user can use the ATM to transfer money to other users and view their account balance. The bank server manages a file "password" that stores user IDs and associated passwords, as shown below. In this project, you can create the file "password" manually.

```
chris 1234
fey 5678
joe 9012
```

The bank server also manages a **file "balance**" that keeps track of the savings and checking account balances for each user. The file "balance" has the following format:

```
<user-id> <saving-balance> <checking-balance>
```

Here, <user-id> is the ID of the user. <saving-balance> and <checking-balance> represent the balances in the user's savings and checking accounts, respectively.

Initially, the file "balance" contains the following data, indicating $10,000 in the savings account and $1,000 in the checking account for each user.

```
chris   10000   1000
fey     10000   1000
joe     10000   1000
```

Both public-key encryption and symmetric-key encryption methods are used for security. Let **Kpub** and K**prb** denote the public and private key of the bank server, respectively. Assume that all clients (i.e., ATMs) have the bank's public key. The public and private keys can be manually generated and stored on the disk.

The client is invoked as:

atm <*Bank server's domain name*> <*Bank server's port number*>

The bank server is invoked as:

bank <*Bank server's port number*>

The detailed steps are given below:

S1: The ATM establishes a connection with the bank server.
S2: The ATM prompts the user to enter their ID and password.
S3: The ATM generates a symmetric key K, sends E(Kpub, K) and E(K, ID||password) to the bank server, where ID and password are the user's ID and password entered, respectively.
S4: The bank decrypts the E(Kpub, K) using Kprb to obtain the symmetric key K. The bank then decrypts E(K, ID||password) using K and obtains the user's ID and the password. Next, the bank compares the ID and the password against the one stored in file "password". If both the ID and the password are correct, then the server sends "ID and password are correct" to the ATM; otherwise, the server sends "ID or password is incorrect" to the ATM. The ATM then displays the message received from the server to the user.

S5: If the ID or the password is incorrect, the ATM prompts the user to re-enter the ID and password. Otherwise, the ATM displays the following **main menu**:

Please select one of the following actions (enter 1, 2, or 3):

1. Transfer money
2. Check account balance
3. Exit

S6: If the user selects option 1, then they are prompted to select between transferring money from the savings account or the checking account:

Please select an account (enter 1 or 2):

1. Savings
2. Checking

If the user enters an input other than 1 and 2, the ATM displays "incorrect input" and asks the user to select either the savings or checking account again. Otherwise, the ATM prompts the user to provide the recipient's ID and the amount to be transferred.

Next, the ATM sends the account (1/2), the recipient's ID, and the transfer amount to the server. The server checks if the recipient's ID exist. If not, the server sends the message "the recipient's ID does not exist" to the ATM. Otherwise, if the user's account has sufficient funds, then the server updates the "balance" file, and sends "your transaction is successful" to the ATM. If there are insufficient funds, the server sends "Your account does not have enough funds" to the ATM. In the above cases, the ATM displays the message received from the server to the user and returns to the main menu.

S7. If the user selects the option 2, then the ATM sends "2" to the server. The server then responds by sending the balances of both the savings and checking accounts back to the ATM. Subsequently, the ATM displays the balances in the following format:

Your savings account balance: <amount>
Your checking account balance: <amount>

The ATM then displays the main menu.

S8. If the user selects option 3, then the ATM sends "3" to the server, and both the ATM and the server close the connection socket, and the server continues listening for connection (i.e., the server should keep the listening socket open).

S9. If the user enters any input other than 1, 2, and 3, the ATM displays "incorrect input" and returns to the main menu.

Submission guideline

- **If you work in a group of 2 students, only ONE group member should submit the project.**
- Please hand in your **source code, public/private keys, file "password", file "balance" that contains the initial balance,** and a **Makefile (C/C++/Java)** electronically (**do not submit .o or**

executable code). Please make sure that your code compiles and runs correctly on remote.cs.binghamton.edu.

- Write **a README** file (text file, do not submit a .doc file) which contains
  - The name and email address of the group members.
  - The programming language you use (C/C++/Java/Python)
  - Code for performing encryption/decryption
  - Whether your code was tested on remote.cs.binghamton.edu.
  - How to execute your program.
  - Anything special about your submission that the TA/grader should take note of.
- Place all your files under one directory with a unique name (such as proj-[userid] for project, e.g. proj-pyang).
- Tar the contents of this directory using the following command. **tar –cvf [directory_name].tar [directory_name]**
- E.g. tar -cvf proj-pyang.tar proj-pyang/
- Use brightspace.binghamton.edu to upload the tared file you created above.

Grading Guideline

- Readme: 5'
- Makefile (C/C++/Java): 5'
- Encryption/decryption: 20'
- Other functionality (C/C++/Java): 60'
- Other functionality (Python): 65'
- Extra-credits (work alone): 10'

Systems Project

In-class presentation + demo: Dec. 7
Submission deadline: 11:59pm Dec. 7 (Thursday)

Students will give presentations and show demos (if applicable). Students will also need to submit presentation slides or videos by 11:59pm on Dec. 7.

1. Buffer Overflow Attack

This project can be done individually (10 points extra credits) or in a group of two students. In this project, you will execute and demonstrate a buffer overflow attack. You are free to choose any example to demonstrate this attack.

2. Kernel Rootkit

This project should be done individually. In this project, you will download a windows rootkit that allows attackers to hide files or processes, and demonstrate how to do it. To carry out this project, you will need to first set up a virtual machine (e.g. virtualbox) and then execute the rootkit inside the virtual machine. It is important to ensure that your computer or laptop is disconnected from the internet during the execution of the.

3. Virus (language: C)

This project can be completed individually (up to 15 points extra credits) or by a group of two students. Please note that this is a challenging project.

In this project, you will design and implement a virus that is capable of infecting all executable files within a specific directory. Note that the program can still execute after the infection. When the infected program executes, the virus will execute first, followed by the program.

4.Capture the Flag (CTF) Challenge

For this project, you need to solve one CTF challenge from https://w3challs.com/, and then demonstrate and explain how to solve the challenge in the class. This project can be completed individually (up to 10 points extra credits) or by a group of two students.

K-12/Community Outreach Project

Submission deadline: 11:59pm Dec. 7 (Thursday)

1. K-12 Education video

This project is intended for individual completion. In this project, you will need to create 1-3 educational videos (totaling approximately 15 minutes), focused on K-12 cybersecurity education. These videos should be age-appropriated (e.g., middle school or high school) and engaging. Sample topics include simple encryption/decryption techniques, online safety, cyberbullying, introduction to computer security, history of cybersecurity, common attacks and countermeasures, cybersecurity career, etc.. The videos should be age-appropriate and engaging.

To submit the video, please upload the video to your youtube account and then email me the youtube link to the video.

2. K-12/Community Outreach Activities

This project is intended for individual completion. In-class presentation + demo will be scheduled sometime between Nov. 28 and Dec. 7.

In this project, you need to brainstorm two ideas for community and K-12 cybersecurity outreach activities (e.g., cybersecurity activities for K-12 students in school and on the community day of the engineering week). These activities should be designed to be fun and engaging.

3. K-12 Online Cybersecurity Games

This project can be completed individually (10 points extra credits) or by a group of two students. In-class presentation + demo will be scheduled sometime between Nov. 28 and Dec. 7.

In this project, you will develop an online cybersecurity game to teach K-12 students cybersecurity concepts. Sample topics include password guessing, encryption/decryption game, cybersecurity quiz, cybersecurity education game, etc. You have the flexibility to choose any programming language for this project.