Project 3: Intel Software Guard Extensions (SGX)

This project is due on **February 27, 2018** at **11:59pm CST**. We strongly recommend that you get started early. You will get 80% of your total points if you turn in up to 72 hours after the due date. Late work will not be accepted after 72 hours past the due date.

The project is split into two parts. Checkpoint 1 helps you to get familiar with the language and tools you will be using for this project. The recommended deadline for checkpoint 1 is **February 20, 2018**. We strongly recommend that you finish the first checkpoint before the recommended deadline. However, you do NOT need to make a separate submission for checkpoint 1. That is, you need to submit your answers for both checkpoints in a single folder before the project due date on **February 27, 2018** at **11:59pm CST**. Detailed submission requirements are listed at the end of the document.

This is a group project; you SHOULD work in a group of **2 or 3 students**.

The code and other answers you submit must be entirely your own work, and you are bound by the Student Code. You MAY consult with other students about the conceptualization of the project and the meaning of the questions, but you MUST NOT look at any part of someone else's solution or collaborate with anyone else. You may consult published references, provided that you appropriately cite them (e.g., with program comments), as you would in an academic paper.

Solutions MUST be submitted electronically in the svn directory of **one team member per team**, following the submission checklist given at the end of each checkpoint. Details on the filename and submission guideline is listed at the end of the document.

ONE submission per group is sufficient.

Release date: February 13, 2018
Checkpoint 1 Recommended Due date: February 20, 2018
Checkpoint 2 Due date: February 27, 2018 at 11:59pm CST

Course SVN: `https://subversion.ews.illinois.edu/svn/sp18-cs463`

Introduction

From previous machine problem, we learned how to implement a secure multi-party computation using the homomorphic encryption. While a homomorphic encryption method provides a strong security guarantee, these algorithms are often unpractical due to a large performance overhead. As an alternative, recent multi-party computation literatures often introduce hardware-based solutions. In this machine problem, we will learn about one of hardware solutions to implement a multi-party machine learning (ML) platform.
Each group represents one of three data owners, each with disjoint 30 samples of Iris plants. The samples have four numeric, predictive attributes and a class label. These three data owners are willing to collaborate and produce a ML model that classifies whether each Iris plant belongs to Iris Virginica class or not. Meanwhile, you do not wish to share your dataset with other collaborators. Your task is to perform a multi-party machine learning computation by training the logistic regression model with three samples without leaking sample information to other entities (including the server). To accomplish this task, you will use a new hardware protection named the Intel SGX which allows programmers to create a secure enclave and to protect code and data inside the trusted enclave.

Please read the assignment carefully before starting the implementing.

Objectives

- Write an enclave application in C/C++.
- Gain familiarity with the Intel SGX APIs.
- Find vulnerabilities of an enclave application.

Checkpoints

This machine problem is split into 2 checkpoints. Checkpoint 1 would help you write untrusted region of an enclave application and a client associated with the enclave server. Your client can run on any machine and can be written in any language, but your enclave application is expected to run in one of our boxes which have Intel SGX compatibilities. In Checkpoint 2, you will implement a ML training algorithm which runs on a trusted enclave. You will test your server and client with *at least two other groups with different datasets* and answer a few questions about your results.

Implementations

You are provided with a hello-enclave template in (`mp3/mp3_server`) to build your server application. We will guide you through the implementation of client and server side of the protocol on this document.

Protocol

This section explains the protocol between an enclave server and clients to perform a multi-party ML computation. The protocol is following.

1. Each client P_i establishes a connection with a server S.
2. Once connection is established, client sends his/her dataset D_i in plaintext to the server.
3. Once server receives all three datasets, server creates an enclave and runs logistic regression training algorithm inside the enclave.
4. Server returns the model M trained on the datasets to each client.

Logistic Regression

This section serves to provide a pseudo-code for the Logistic Regression algorithm for your server implementation. We are given a Iris plant data with four features (`SepalLengthCm,SepalWidthCm,PetalLengthCm,PetalWidthCm`) and two classes: 1 if Iris plans is Virginica and 0 otherwise. If this dataset can be separated by two given classes depending on the features, then we can use logistic regression to train our model with samples and predict whether an unknown Iris plant is Virginica or not. Once our server receives three disjoint datasets from three clients, we train our model using the pseudo-code below.

```
"""
Inputs:
features: a 90 by 4 matrix of Iris training samples
labels: a 90 by 1 matrix of binary label for each sample
lr: learning rate (use 0.01 for this exercise)
iters: number of iterations (use 50 for this exercise)
"""

def train(features, labels, lr, iters):
    initialize weights to a zero vector

    for i in range(iters):
        #1. Get Predictions
        z = dot(features, weights) # (90,4) * (4,1) = (90,1)
        for each z[i] in z:
            predictions[i] = 1.0 / (1 + exp(-z))

        #2. a dot product of transposed features and (predictions - labels)
        #this represents the aggregate slope of the cost
        #function across all observations
        gradient = dot(features.T,  predictions - labels)
        #3. Take the average cost derivative for each feature
        gradient /= 90

        #4. Multiply the gradient by our learning rate
        gradient *= lr

        #5. Subtract from our weights to minimize cost
        weights -= gradient

    return weights
```

Your task in Checkpoint 2 is to need to write this algorithm in C/C++ and complete the training. The vector of weights returned from this pseudo code represents a model M, a weight vector, returned by a server. To understand the algorithm in depth, please refer to the link below.

Reference

ML Cheatsheet - Logistic Regression
`http://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html`

3.1 Checkpoint 1 (20 points)

This machine problem is split into 2 checkpoints. would help you write untrusted region of an enclave server and a client associated with the enclave application. You will test your client code on your own machine and test server code on our SGX boxes.

3.1.1 Inform Course Staff (0 points)

Once you formed a group to complete this machine problem, one person from your group must contact the course staff via email (`lee559@illinois.edu`) to receive:

1. the Iris Plant dataset which is composed of 30 samples
2. address of an enclave box to run your server application on
3. port number for your server application

Please include [CS463 MP3] in your email title. This process is necessary for us to ensure that we allocate resources evenly to each group. Please complete this procedure as soon as possible. You SHOULD use remote authentication methods such as `ssh` to use the SGX machine.

What to submit No submission required.

3.1.2 Implement the TCP Client (10 points)

In this section, you need to complete the implementation of the client and test it on your own machine. We have not provided any skeleton code for this portion as your client does not need SGX functionality for the exercise. You may write your client in any language as long as the client can:

1. Establish a connection with the server
2. Send your dataset in plaintext to the server. In other words, you simply need to read your csv dataset file in a buffer and send the string to the server via a socket.
3. Receive the model computed by the server as a reply (a weight vector composed of a comma-separated string with four values).

When you submit your client code, you must provide a README file that explains how to compile and run the client. You can use any TCP client code template from the web as long as you cite the reference in the README file. You MUST compress your client code and the associated README file to `mp3_client.tar.gz`.

What to submit

- Create a directory called `mp3` under your subversion repository.
- `mp3/mp3_client.tar.gz` (contains client code and a README file)

3.1.3 Implement Untrusted Code (10 points)

In this section, you will need to implement untrusted portion of an enclave application. You are provided with a hello-enclave template in (`mp3/mp3_server`) to build your application. The template is composed of two folders, `App` and `Enclave`.
To start understanding the template, read `App.cpp` which contains the `main()` function. All untrusted code of this template (including OCALLs) is in the `App` folder, so you will only implement code inside this folder. After the implementation, your server is expected to:

1. Open a TCP server on a designated port
2. Receive the csv string from client and parse the dataset.

Note: Before you compile your server code with the `Makefile`, run following command on your terminal. You may want to add this line to `.bashrc`.

```
source /opt/intel/sgxsdk/environment
```

What to submit

- `mp3/mp3_server`

3.2 Checkpoint 2 (80 points)

For this checkpoint, you'll need to implement trusted code segment (`mp3/mp3_server/Enclave`) of your server. After the complete implementation, you test your server with two other groups with different datasets like you did for the MP2. You CANNOT test it within your own group.

3.2.1 Find Your Partners (10 points)

Please find at least 2 groups that you want to work with. Three groups should all have different datasets for this exercise. Please create a txt file called `groups.txt` under `mp3`.

- On the first line of the file, please put the netids of YOUR OWN group members, separated by space.
- On the second line of the file, please put the netids of the first group you plan to work with, separated by space.
- On the third line of the file, please put the netids of the second group you plan to work with, separated by space.

What to submit

- Place `groups.txt` in `mp3`

3.2.2 Implement Trusted Code (40 points)

In this section, you will need to implement trusted portion of an enclave application. Note that all trusted portion of this template (ECALLs) is in `mp3/mp3_server/Enclave`, so you must implement logistic regression algorithm in this directory. We expect you to implement one ECALL which is called by server after all three datasets are collected. This ECALL allows the application to run a logistic regression training algorithm on the dataset. For implementation of this algorithm, please refer to a section above on the Logistic Regression.

Tips:

- When you run your code in Enclave, you cannot use some library functions. These are listed in the Appendix of the Intel Software Guard Extensions Developer Reference.
`https://download.01.org/intel-sgx/linux-2.1/docs/Intel_SGX_Developer_Reference_Linux_2.1_Open_Source.pdf`
- Intel has provided some sample Enclave applications when they released the SDK. These codes will help you to understand on syntax format for defining and calling the ECALLs and OCALLs.
`https://github.com/intel/linux-sgx/tree/master/SampleCode`

- Among sample applications, you SHOULD read `SampleEnclave` application which is a producer-consumer application. This application shows how to write a multi-threaded enclave application.
- Do not forget to define your OCALLs and ECALLs in `Enclave/Enclave.edl`.

What to submit

- `mp3/mp3_server`

3.2.3 Report (30 points)

Please write a 1-page report as follows:

- Please put the names and netids of ALL the group members at the top of the report. (5 points)
- What is the accuracy of label prediction for the model computed by your server? (10 points)
- Find and explain a vulnerability of this protocol. (10 points)
- Share your experience in Enclave programming. (5 points)

Please name the report as `report_[netid1]_[nedid2]_[netid3].pdf`.

What to submit

- Place `report_[netid1]_[nedid2]_[netid3].pdf` in `mp3`.

3.3 Extra Credit

You may have realized that this multi-party computation is not secure. For extra credit, you may propose and implement a countermeasure to patch a vulnerability that you found. We will award an extra credit of upto 10 percent of total MP grade for your work. However, your total MP grade cannot exceed 100 percent. Thus, this extra credit can only be used to cover the points you have lost on other MPs.
Note: You may need a `sudo` privilege to install necessary packages, so contact course staff if necessary.

What to submit

- Place your new server and/or client in `mp3/ec`.
- `mp3/ec/README` [a paragraph of what you did]

Submission Checklist

Place the following files in `mp3` in your subversion repository.

- `mp3/report_[netid1]_[nedid2]_[netid3].pdf` [Report for Section 3.2]
- `mp3/groups.txt` [list of group members and partner groups]
- `mp3/mp3_client.tar.gz` [Implementation of the client]
- `mp3/mp3_server` [Implementation of the server]
- `mp3/ec` [Implementation of a countermeasure (not required)]