Course Project Specification

CSCI-4341-04-Fall 2025: Foundations of Intelligent Security Systems

Instructor: Dr. Sergei Chuprov, sergei.chuprov@utrgv.edu

THIS PROJECT DESCRIPTION IS SUBJECT TO CHANGES. YOU WILL BE INFORMED IF ANY CHANGES ARE INTRODUCED.

Project Description:

The course project is a significant component of this class, accounting for **45% of the final grade**. Students may choose between two options for their project:

1. **General Project:** Students can propose a topic of their choice, aligned with the course themes, such as AI, Security, and Data Management. The topic must be approved by the instructor.
2. **Instructor Research-Based Project:** Students can work on a project related to Federated Learning and based on the instructor's research. This option requires a higher standard of rigor and includes producing a report close in quality to a research publication. **If you select this type of project, you will receive a 7% bonus to your final grade.**

Section 1 outlines the core requirements and milestones, applicable to both types of projects.

1 General Requirements and Milestones

Key Requirements

1. **Research and Practical Implementation:** All projects must combine a research component with a practical implementation component. Students are required to:
   - Conduct a literature review and research relevant to their chosen topic.
   - Develop a software prototype or solution as part of the project.
2. **Submission Format:**

   All submissions are done through Brightspace. The appropriate sections will be set up.

   - Reports must follow standard **IEEE** (https://www.ieee.org/conferences/publishing/templates.html) or **ACM conference templates** (https://www.acm.org/publications/proceedings-template), available in LaTeX or Word, and be submitted in **PDF format**. **The use of these templates is strictly enforced. Reports not using the correct template will receive a 30% grade penalty for the report portion.**
   - Slides for all presentations must be in **PDF format** and submitted at least **one hour before class starts**.
   - Any software and code implementations **should be submitted in an archive with an included README file** describing how to execute it and how to reproduce the results. The implementation can also be shared in a repository (e.g., GitHub) and provided as a link.

3. **Group Size:**
    - Projects may be completed individually or in groups of up to **3 members**.
    - Groups must self-enroll by the enrollment deadline (see timeline).
4. **Late Submission and Attendance Policy:**
    - **Late Reports:** A penalty of **10% per day** will be applied to the report grade for any late submission.
    - **Absence on Presentation Day:** If a presenter is not attending a presentation in person without a valid, documented reason, they will receive a **grade of 0** for that specific presentation milestone.

Project Milestones and Grading

The project will progress through the following milestones, with deliverables and grading outlined below:

1. **Project Proposal Presentation (4% of final grade):** Teams will present their project idea, plan, design, and proposed implementation and evaluation strategy.
    - **Submission:** A **three-page proposal summary in PDF format**, using the specified template.
    - Presentation Duration: **5 minutes + 3 minutes Q&A**.
2. **Progress Presentation 1 (7% of final grade):** Teams will present their progress, including dataset selection, tool choices, and research question formulation.
    - **Submission:** A **three-page project update in PDF format**, using the specified template.
    - Presentation Duration: **5 minutes + 3 minutes Q&A**.
3. **Progress Presentation 2 (7% of final grade):** Teams will present further progress, including methodology, experimental design, and preliminary results.
    - **Submission:** A **three-page project update in PDF format**, using the specified template.
    - Presentation Duration: **5 minutes + 3 minutes Q&A**.
4. **Final Presentation and Project Submission (27% of final grade):** Teams will present their finalized project and submit all final materials. This milestone is graded in two parts:
    - **Final Presentation (9%):** Present results, conclusions, and a demonstration. Presentation Duration: **20 minutes + 10 minutes Q&A.**
    - **Final Report and Code (18%):** Submit the comprehensive final report and all supporting code.
    - **Submissions:**
        - The **final presentation slides** in PDF format.

- The **final report (minimum 6 pages)** in PDF format, using the specified template.
- The **complete project code**, submitted either as a single archive file (e.g., .zip) or as a link to a public repository (e.g., GitHub). The code must include a `README` file with clear instructions on how to run it and reproduce your results.

Timeline

Week	**Milestone**	**Submission Deadline**
Week 2	Project groups enrollment deadline.	9/12 @ 11:59 PM
Week 4	Project Proposal Presentations.	Three-page proposal due 9/22 @ 11:59 PM
Week 6	Project Milestone 1 Presentations.	Three-page project updates due 10/13 @ 11:59 PM
Week 9	Project Milestone 2 Presentations.	Three-page project updates due 11/3 @ 11:59 PM
Week 15	Final Project Presentations & Submission.	Final report and code due 12/10 @ 11:59 PM

2 Example General Project: Intrusion Detection System Design and Evaluation

Note: This is only an example of a project you may attempt to do. Please, discuss with the instructor more specifics on your project and your plans.

Project Topic: Intrusion Detection System (IDS) Design and Evaluation

Project Learning Outcomes

As a result of this project, you should be able to:

- Demonstrate an understanding of the key principles involved in IDS operation and design.
- Demonstrate knowledge of basic IDS design techniques.
- Design and implement a software engineering project related to IDS.
- Conduct research and development in AI and security domains, including:
    1. Applying simulation and design tools to test IDS systems empirically.
    2. Performing statistical analysis and evaluation of the results.
    3. Writing a report and delivering a presentation.

2.1 General Timeline of the Project

- **Milestone 1:** Research on tool choice, data preparation, statistical-based intrusion detection design.
- **Milestone 2:** Artificial intelligence and machine learning methods in IDS design.
- **Milestone 3:** Results analysis and presentation.

Project Description

In this project, you will use various methods and tools to design an IDS, which must be AI/ML-based, meaning that intelligent systems need to be used for the detection. You will compare them based on their performance and resource consumption. As the example of the performance metrics, you may evaluate your IDS design using the following ones:

- **False Positives Ratio (FPR):** The ratio of cases where the IDS signals an attack when there is none.
- **False Negatives Ratio (FNR):** The ratio of cases where the IDS fails to detect an attack when one exists.

You may use any design tools provided by programming languages or environments. Decisions on tool choice and implementation must be detailed in your **project proposal**.

Background Information on IDS Design

Intrusion Detection is a key monitoring technique in computer security that detects breaches that cannot easily be prevented by access controls or information flow techniques. Such breaches may result from software bugs, authentication failures, or system mismanagement. Intrusion detection is typically divided into two categories: anomaly detection and misuse detection.

- **Anomaly Detection:** Detects intrusions by identifying deviations from normal system behavior, such as unusual CPU usage or I/O activity. By maintaining profiles of normal activity for users or programs, large variations can be flagged as potential intrusions.
- **Misuse Detection:** Identifies intrusions by matching known patterns of abuse, such as exploiting vulnerabilities in software. Techniques include rule-based systems, state machines, and expert systems.

Statistical-Based Intrusion Detection (SBID) systems analyze audit trail data by comparing it to expected profiles to detect potential security violations. Historical systems include Haystack, IDES, and MIDAS, each with unique contributions to IDS technology.

General Content

1. Design and test an IDS system. Prepare data compiled from real-life experiments for IDS development and testing.
2. If necessary, develop a software tool for data selection and preprocessing, performing tasks like feature extraction or re-formatting records.
3. Select an IDS tool, ensuring compatibility with your operating system and input file format.
4. Prepare data sufficient for developing an IDS capable of detecting at least five attacks and normal conditions, with randomized records for training and testing.
5. If you are training the model, divide the data into:
    - **Training Data:** Two-thirds of the dataset for IDS design.
    - **Testing Data:** One-third of the dataset for performance evaluation.
6. Design and develop an IDS performing both misuse and anomaly intrusion detections using a machine learning model.
7. **Misuse-Based IDS:**
    (a) Train the IDS to derive attack patterns.
    (b) Design a classification procedure and test it for at least five attacks.
    (c) Evaluate false positive and false negative ratios, adjust the procedure, and iterate to improve results.
    (d) Measure resource consumption (time, memory) during IDS testing.
8. **Anomaly-Based IDS:**

(a) Derive patterns for normal behavior and use them for anomaly detection.

(b) Design a classification procedure for binary detection (attack/normal) and test it.

(c) Evaluate FPR and FNR, adjust the procedure, and iterate to improve results.

(d) Measure resource consumption (time, memory) during IDS testing.

9. Document the project:

   - Research, decisions, and data preparation processes.
   - Description of the IDS tool and its application.
   - Testing procedures and evaluation methods.
   - For group projects, document workload distribution and coordination efforts.

Data Selection

To conduct the experiments, you need a proper set of data. You have two options:

1. Use the benchmark data of the International Knowledge Discovery and Data Mining group (KDD).
2. Find data on the web. Please note that you need to find records of network traffic with at least five attacks present, as well as normal traffic.
3. Collect your own network traffic data with attacks simulated **(you will receive 5% bonus for this to your project grade)**

Option 1: KDD Data

These data are based on the benchmark of the Defense Advanced Research Projects Agency (DARPA), collected by the Lincoln Laboratory of the Massachusetts Institute of Technology in 1998. This was the first initiative to provide designers of Intrusion Detection Systems with a benchmark to evaluate different methodologies. For further details, see *DARPA, Intrusion Detection Evaluation*, MIT Lincoln Laboratory, 1998 (http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html) and other publications on this website.

To collect these data, a simulation was made of a fictitious military network consisting of three "target" machines running various operating systems and services. Additional machines were used to spoof different IP addresses, generating traffic between various IPs. Finally, a sniffer was used to record all network traffic in the TCP dump format. The total simulated period was seven weeks.

Details of the Data

- **Normal Connections:** Simulated to profile expected behavior in a military network.
- **Attack Types:** Attacks were categorized into one of five classes:
  - User to Root (U2R)
  - Remote to Local (R2L)

  - Denial of Service (DOS)
  - Data
  - Probe

- **Connection Definition:** A connection is defined as a sequence of TCP packets starting and ending at well-defined times, during which data flows between a source IP address and a target IP address under a specified protocol.

In 1999, the original TCP dump files were preprocessed for utilization in the IDS benchmark of the International Knowledge Discovery and Data Mining Tools Competitions. For further details, see *KDD Cups 99 - Intrusion Detection Contest, 1999* (http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html).

Original Data Format for KDD Data

The dataset includes a description of connections using the following features:

1. Duration of the connection.
2. Protocol type (e.g., TCP, UDP, ICMP).
3. Service type (e.g., FTP, HTTP, Telnet).
4. Status flag.
5. Total bytes sent to the destination host.
6. Total bytes sent to the source host.
7. Whether source and destination addresses are the same.
8. Number of wrong fragments.
9. Number of urgent packets.

Derived Features

In addition to the basic features, each connection is also described by 32 derived features, divided into three categories:

- **Content Features:** Domain knowledge is applied to analyze the payload of the original TCP packets (e.g., number of failed login attempts).
- **Time-Based Traffic Features:** These capture properties that develop over a 2-second temporal window (e.g., number of connections to the same host within 2 seconds).
- **Host-Based Traffic Features:** These assess attacks spanning longer intervals by analyzing the last 100 connections instead of time windows.

Each record in the dataset contains **41 attributes** and a **target value**, which indicates the attack name. For further reading, see:

- Lee, W., S. Stolfo, and K. Mok, *Mining in a Data-Flow Environment: Experience in Network Intrusion Detection.* Proceedings of the 5th ACM SIGKDD, 1999.
- Lee, W., S.J. Stolfo, and K.W. Mok, *A Data Mining Framework for Building Intrusion Detection Models.* IEEE Symposium on Security and Privacy, 1999.

Strategy for Model Design to Detect Attacks

In this project, students have the flexibility to choose any programming language or tool they prefer for implementing their IDS. This allows them to leverage their existing expertise and comfort with specific development environments.

Implementation Guidelines

If students choose to use a programming tool for the project, they must ensure that their solution fulfills the following requirements:

- **Implementation of an IDS:** The IDS must be implemented as a software solution that integrates the following components:
  1. **Training the ML Model:** Develop and train a machine learning model using the prepared dataset. The model should be capable of identifying at least five different attack types along with normal network traffic.
  2. **Testing the Model:** Apply the trained model to a separate testing dataset to evaluate its performance.
  3. **Result Evaluation:** Implement functionality to compute and report key performance metrics, such as FPR, FNR, and overall accuracy.
- **Documentation:** Provide the documentation describing the software's design, usage instructions, and the results obtained during testing.

Tool Options

Students may select from a wide range of tools, including but not limited to:

- General-purpose programming languages such as Python, Java, C++, or MATLAB.
- Specialized machine learning platforms or libraries, such as TensorFlow, PyTorch, or Scikit-learn.

Evaluation Metrics

The performance of the IDS should be assessed based on its ability to accurately detect attacks and distinguish them from normal behavior. Key metrics to include are:

- FPR
- FNR

- Overall Accuracy
- Resource Consumption

3 Project Based on Instructor's Research

Research Project: Federated Learning and Privacy-Preserving Machine Learning

Note: This is only an example of a project you may attempt to do. Please, discuss with the instructor more specifics on your project and your plans.

The objective of this research project is to explore the concept, methodologies, and applications of **Federated Learning (FL)** with a focus on its role in privacy-preserving machine learning. Students will examine the challenges, solutions, and potential of FL in various domains, and understand how it contrasts with traditional centralized machine learning models.

You can find more background on FL in Attachment 1 in the end of this document.

Topics to Explore

During this course, students will explore the following topics:

1. Exploration of FL with literature review, Machine Learning model definition, and dataset preparation.
2. Implementation of a Conventional FL Model and evaluation of the setup against security threats.
3. Implementation of FL with a Proportional-Integral-Derivative (PID) approach and evaluation of the setup against security threats.
4. Presentation and Report.

Project Learning Outcomes

As a result of this project, students should be able to:

- Demonstrate an understanding of the key principles involved in FL operation and design.
- Demonstrate a good knowledge of basic ML and FL techniques.
- Demonstrate an ability to design and implement a software engineering project to conduct the research and development activities stated above.
- Demonstrate skills in conducting research and development in artificial intelligence and security domains.

Project Workflow

The workflow of the project includes:

- Learning the problem and understanding FL principles.

- Potentially collecting data or benchmarks with known results from other researchers.
- Conducting experiments to evaluate the chosen models and methodologies.

The project will include the following milestones.

3.1 Milestone 1: Research and Data Preparation

Your Milestone 1 consists of three parts:

1. Bibliography and patent search and review
2. Data source analysis, preparation, and pre-processing
3. Formulation of the initial (foundational) model

Section A: Bibliography and Patent Search and Review

In Section A, you must conduct a bibliography and patent search, analyze your findings, and write a concise review. Your report should cover at least two of the following topics:

1. **FL Feature Analysis:**
   (a) Analyze and compare key aggregation methods and algorithms (e.g., Federated Averaging) and recent advancements.
   (b) Analyze the differences between Federated Learning and traditional centralized machine learning, particularly in terms of privacy, data security, and scalability.
2. **Privacy and Security in Federated Learning:**
   (a) Explore how Federated Learning addresses privacy concerns in machine learning. Investigate techniques such as differential privacy, secure multi-party computation, and homomorphic encryption.
   (b) Discuss the vulnerabilities of Federated Learning systems, including potential threats such as model poisoning and inference attacks. Provide examples and case studies.
3. **Challenges and Future Directions:**
   (a) Identify the key challenges facing the broader adoption of Federated Learning, such as communication efficiency, model heterogeneity, and the lack of labeled data.
   (b) Discuss potential future research directions in Federated Learning, including advancements in communication protocols, the introduction of metacognition, and improved strategies for addressing data heterogeneity.

Section B: Data Source Analysis, Preparation, and Pre-processing

In Section B, you have to choose which dataset you will be using for your project and the problem you will be investigating. You will have three options:

1. **Option 1:** Choose the dataset provided by the instructor.
    - Instructor will provide you with a subset of the Federated Extended MNIST (FEMNIST) dataset (https://leaf.cmu.edu/). The subset represents handwritten digits distributed among clients in IID (Independent and Identically Distributed) manner.
2. **Option 2:** Use the Medical MNIST (MedMNIST) dataset (https://medmnist.com/).
3. **Option 3:** Choose another dataset and problem on your own. Please refer to Attachment 2 at the end of this document for the description of common datasets for FL.

Data Preparation Strategy

The Data Preparation Strategy is the process of preparing data for the specific task that you aim to solve with your ML model. Existing datasets are often organized in such a way that they can be used for a broad range of setups. Some datasets, such as the FEMNIST dataset, are specifically prepared for use in FL. Others, like MNIST, are intended for centralized use, so you will need to simulate the distribution of data across clients.

If you choose Option 2 or Option 3, you will have to implement the Data Preparation Strategy, which may include the following steps (not all of them may need to be implemented for your project):

1. **Data Cleaning:**
    - Duplicate Removal: Identify and remove duplicate samples to ensure data quality.
    - Noise Reduction: Apply techniques such as denoising filters or thresholding to reduce noise in the images, improving model performance.
2. **Data Normalization:**
    - Normalize the pixel values to a standard range (e.g., [0, 1]) to facilitate faster convergence during model training.
3. **Data Augmentation:**
    - Purpose: To increase the diversity of the training data and mitigate overfitting.
    - Techniques: Implement random rotations, translations, zooming, and horizontal flipping of images to augment the dataset.
4. **Data Partitioning:**
    - Client Simulation: Partition the dataset by writer, where each writer's data is treated as a distinct client in the federated learning setup.
    - Non-IID Data Distribution: Maintain the natural non-IID distribution to reflect real-world federated learning scenarios.

5. **Data Labeling (For Milestone 2: Conventional FL):**
    - Ensure that all images are correctly labeled and that the labeling format is consistent across the dataset. This step is crucial for the correct functioning of the federated learning algorithm.
6. **Data Corruption (For Milestone 3, FL with PID):**
    - Corrupt labeling on a number of clients according to your group number.

Section C: Formulation of the Initial (Foundational) Model

In Section C, you must formulate an initial (foundational) model. In FL, the initial model is typically a model architecture that is trained either from scratch or pre-trained on a central dataset before being distributed to clients. When choosing an ML model, consider that it may be easier to use models supplied with the framework you are using. I suggest using the Flower Framework with PyTorch (https://flower.ai/). Other options include TensorFlow Federated Learning, PySyft, or any other framework of your choice.

Below are the options for creating the initial model:

1. **Training from Scratch:**
    - Define the Model Architecture: Design a neural network or another type of machine learning model suitable for your task. This architecture will be consistent across all clients.
    - Centralized Training (Optional): If available, train the model on a central dataset. This can be a small representative subset that gives the model a good starting point before federated learning begins.
    - Initialize Weights Randomly: If you don't have a central dataset or choose not to pre-train, initialize the model weights randomly.
2. **Using a Pre-trained Model:**
    - Select a Pre-trained Model: Choose a model pre-trained on a related task or dataset, such as a model pre-trained on ImageNet for image classification.
    - Adapt the Model: Modify the final layers to suit the specific task or dataset (e.g., replace the output layer to match the number of classes in your dataset).
    - Fine-tune (Optional): Fine-tune the pre-trained model on a small representative subset of data to adapt it to the features of your federated learning task.
3. **Distributing the Initial Model:**
    - Model Broadcasting: Once the initial model is ready, it is broadcasted to all clients participating in the federated learning process. Each client receives the same model architecture and initial weights.
    - Local Training on Clients: Each client trains the model on its local data and updates the model weights based on their specific dataset.
4. **Considerations for Initial Model Selection:**

- Model Complexity: Ensure that the model is not too complex for the devices running it, as federated learning often involves edge devices with limited computational power.
- Generalization Ability: The initial model should generalize well to diverse and potentially non-IID data across different clients.
- Communication Efficiency: The size of the model affects communication costs in FL. Larger models may increase communication overhead, so balance model complexity with communication efficiency.

Example of Initial Model Creation: For instance, using the FEMNIST dataset:

- Define a Convolutional Neural Network (CNN) suitable for handwritten character recognition.
- Initialize weights using Xavier initialization.
- Broadcast the CNN model to all clients.
- Clients train the model on their local FEMNIST data.

The model definition example for the FEMNIST dataset is provided by the authors: https://github.com/TalwalkarLab/leaf/blob/master/models/femnist/cnn.py

3.2 Milestone 2: Implementation and Evaluation of the Conventional FL

Overview of Federated Learning Approach

Federated Learning (FL) has attracted significant attention in the research community since its inception in 2016. The most prominent advantage of FL over centralized learning is the preservation of training data privacy. Higher data privacy is achieved because the training data is never transferred over a network; instead, the Machine Learning (ML) model training is conducted locally at each participating client. The essence of the difference with centralized learning can be explained as follows:

- **Centralized Learning:** Move the data to the computation.
- **Federated Learning:** Move the computation to the data.

By doing so, FL enables machine learning (and other data science approaches) in areas where it wasn't possible before. For example, we can now train excellent medical AI models by enabling different hospitals to work together. We can solve financial fraud by training AI models on data from different financial institutions. We can also build novel privacy-enhancing applications (such as secure messaging) that have better built-in AI than their non-privacy-enhancing alternatives. These are just a few examples, and as FL is deployed, more areas will be reinvented with access to vast amounts of previously inaccessible data.

Typical Steps in Federated Learning Model Training

1. **Initialize Global Model:** We start by initializing the model on the server, similar to centralized learning. The model parameters are initialized, either randomly or from a previously saved checkpoint.

2. **Send Model to Client Nodes:** The parameters of the global model are sent to the connected client nodes (e.g., smartphones, IoT devices, or organization servers). These client nodes will use the same model parameters to start their local training. Only a few selected client nodes are used to avoid diminishing returns with more nodes.

3. **Train Model Locally:** Each client node trains the model on its local dataset. The training process doesn't need to converge fully; it only trains for a limited time, often for a few steps or mini-batches.

4. **Return Model Updates to the Server:** After local training, each client node sends back its model updates, which may include full model parameters or gradients accumulated during local training.

5. **Aggregate Model Updates:** The server aggregates the model updates received from the client nodes. This aggregation process typically uses Federated Averaging (FedAvg), which computes a weighted average of the model updates based on the number of examples each client used for training.

6. **Repeat Steps 2 to 5 Until Convergence:** The process of sending model updates, training locally, and aggregating updates is repeated multiple times until the global model converges. Each round of federated learning trains the model on data from all participating client nodes, although each client only trains the model for a short period.

You can see this brief video to better visualize how FL works: https://www.youtube.com/watch?v=X8YYWunttOY

Attack Scenarios in Federated Learning

While FL provides a higher level of security by eliminating the transfer of data and preserving it locally, there are still potential attacks that can target the ML model training process. These attacks can be classified into two main categories:

- **Data Poisoning Attacks:** These occur when the data used in an ML system becomes corrupted due to adversarial attacks or unintentional harmful conditions (e.g., poor network connection, storage damage). This process is referred to as *Data Quality (DQ) degradation*.

- **Model Poisoning Attacks:** These threats occur when an attacker intentionally manipulates the ML model itself, causing the acquired model to become invalid. This can happen if an attacker gains access to a local training node and replaces the local model.

In this course project, you will perform Data Poisoning attacks on certain clients' data in the FL simulation process. Your goal is to analyze how the global model training degrades during such an attack. The metrics used to measure the degradation will be described in the following sections.

Implementation of Conventional Federated Learning Setting

Conventional Federated Learning refers to FL without any incorporated attack mitigation techniques. In this setting, if any participating clients are compromised (e.g., intruders gain access to a model training on a smartphone or IoT device), they can negatively influence the global model training by sending irrelevant model updates to the server.

In the implementation phase of this project, you may choose your tools, programming languages, and frameworks to design the FL system. However, we suggest using the Flower Framework with Python. You are required to implement an FL setup with at least 10 participating clients.

Federated Learning Workflow

Your workflow when implementing Conventional FL should be as follows:

1. **Research existing FL frameworks and make a decision on which framework you will use.** When considering FL frameworks, keep in mind that you need to collect evaluation accuracy and loss metrics (per client, both) after each aggregation round in order to evaluate the training process. Additionally, the framework should allow the implementation of custom aggregation strategies to incorporate the PID approach. If you don't have a specific reason for choosing your own framework, we suggest using **Flower**.
2. **Download and install the Flower Framework (https://flower.ai/) (or another framework of your choice).**
3. **Implement the FL setup:**
   - Establish centralized model initialization and client-server communication.
   - Implement the ML model suitable for the given dataset and task.
4. **Verify that your setup is working properly:**
   - Execute simulation on the given dataset without data poisoning.
   - Collect and analyze metrics during the runtime.
   - Implement Data Poisoning on the specified number of clients.
   - Execute simulation on the given dataset with data poisoning.
   - Collect and analyze metrics during the runtime.
5. **Note:** We do not require you to implement the FL simulation across physically separate machines with real-world network communication. The simulation should be conducted within one physical machine. However, if you would like to explore a more realistic setup involving computations across separate machines or virtual machines and network communication, please let us know, and we will guide you through this.

Evaluation of Conventional Federated Learning Setting

Description of the Dataset

You will work with the dataset that you have chosen in Part 1.

Description of Metrics to Collect During Each Evaluation Phase

In order to perform the evaluation of the FL performance, you must collect the following metrics during the FL simulation after each aggregation round:

- **Loss per client:** The loss (error) should decrease with each aggregation round. The Flower Framework calculates the loss value after each aggregation round (refer to the Flower documentation).
- **Average loss among clients:** Can be calculated using the collected per-client loss values.
- **Evaluation accuracy per client:** Accuracy metrics are collected during the evaluation phase after each aggregation round. After each round, each client's model accuracy is evaluated on its evaluation subset. The accuracy should increase as the model trains. The Flower Framework calculates the accuracy metrics.
- **Average evaluation accuracy among clients:** Can be calculated using the collected per-client accuracy values.

You need to save the collected metrics to a CSV (or other suitable format) so that they can be accessed for further analysis, plotting, and comparison of strategies with and without incorporated defense mechanisms.

Evaluation Without an Attack

During the evaluation of your FL setup without an attack, if the model is designed correctly to comply with your dataset, you should see that the evaluation accuracy increases as rounds go by. Simultaneously, the loss should decrease with each aggregation round.

Based on the metrics you collect during the simulation, you will plot the following graphs:

1. History of loss per client over rounds.
2. History of average loss among clients over rounds.
3. History of evaluation accuracy per client over rounds.
4. History of average evaluation accuracy among clients over rounds.

Evaluation With an Attack

During the simulation with implemented attack scenarios, you will see that the accuracy of each malicious client will not increase with each round. The average loss, at the same time, may not increase as fast as it did without an attack or, in some cases, may even start to decrease. This would indicate that the attack works, and the centralized model cannot converge.

If on your per-client accuracy and per-client loss graphs you cannot distinguish benign clients from poisoned ones, your setup is incorrect. Possible steps for troubleshooting:

- Your plotting functions may be incorrect.

- Your poisoning might not be effective (this can happen if you poison in the runtime, try creating the poisoned dataset in the file system and load it directly).
- You may be evaluating incorrectly (ensure that you are evaluating on the local poisoned client's subset).

Based on the metrics you collect during the simulation, you will plot the following graphs:

1. History of loss per client over rounds.
2. History of average loss among clients over rounds.
3. History of evaluation accuracy per client over rounds.
4. History of average evaluation accuracy among clients over rounds.

Thus, after both evaluations with and without an attack, you should produce 8 graphs in total, along with the corresponding files containing the collected metrics.

3.3 Milestone 3: Implementation and Evaluation of Federated Learning with PID

Description of the PID Concept

Detection of model anomalies can be based on the calculation of various metrics using the model weights that each client supplies to the aggregation server after each training round, as the model weights are the only information that the server receives from each client. One possible approach is inspired by the Proportional-Integral-Derivative (PID) controller (https://en.wikipedia.org/wiki/Proportionalintegralderivative_controller). While the classic PID controller is given as:

$$u(t) = K_p e(t) + K_I \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt},$$

our adaptation utilizes the distance of each client's model to the centroid of all client models at each aggregation round:

$$u(t) = K_p D(t, \mu) + K_I \sum_{i=0}^{t-1} D(i, \mu) + K_d (D(t, \mu) - D(t-1, \mu))$$

where $D(t)$ is the Euclidean distance of the weights from the centroid μ during round t. The coefficients K_p, K_i, and K_d determine the weights of the proportional, integral, and derivative parts, respectively, and must be investigated empirically for a particular setting.

The algorithm can be described as follows:

Input: client-provided local models
Output: aggregated global model

Clients execute:

- Receive global model from the server
- For each local epoch do:
  - Execute training algorithm (e.g., SGD)
- Send the local model to the server

Server executes:

- For each aggregation round (1, 2, . . .):
  - Receive weights from clients
  - Compute PID
  - For each client do:
    - If client PID $<$ PID threshold, then exclude client from aggregation
  - Perform aggregation based on FedAvg
  - Distribute aggregated global model back to clients

Calculation of PID Step-by-Step

PID scores for each client are calculated based on the weights received from clients. Each client's PID is calculated independently from other clients.

The process of calculating the PID scores is as follows:

1. Calculate the normalized 2-distance from the center (mean) of the model parameters to each client's model.
2. Calculate the initial client's PID at round 1 using only the first part of the formula – the distance to the centroid.
3. In the next aggregation round, calculate PID using the entire formula.
4. Compare the PID of each client participating in the aggregation round to the established threshold, and discard the clients whose PID is greater than the threshold from the aggregation, global model distribution, and further communication.

Please note that if you set the variables K_p, K_i, and K_d to 1, 0, and 0, respectively, only the distance will be considered during the calculation of the PID score. The latter two components of the formula are responsible for incorporating the historical behavior of each client. These need to be adjusted for a particular setting. For the FEMNIST subset, you can start with the values of $K_p = 1$, $K_i = 0.05$, and $K_d = 0.5$.

Evaluation with an Attack

When evaluating the implemented PID algorithm, you should be able to see that malicious clients will be removed during the aggregation if the algorithm is implemented correctly. Invent a way to

demonstrate the client removal dynamics and implement it in your code. Include the client removal dynamics in your report.

Like for the Conventional FL, you will plot the following graphs, along with the client removal dynamics:

1. History of loss per client over rounds.
2. History of average loss among clients over rounds.
3. History of evaluation accuracy per client over rounds.
4. History of average evaluation accuracy among clients over rounds.

Results Discussion

You will discuss the results achieved during Conventional FL and FL with PID. Ideally, if you implemented the algorithm correctly, the algorithm will exclude the malicious clients from the simulation. After exclusion, you should observe that the average loss decreases faster, as the incorrect models are no longer provided. Include the following plots in your report along with your results discussion:

1. Average loss over rounds with the attack in Conventional FL vs Average loss over rounds with the attack in FL with PID – put these metrics on one plot to better showcase the difference.
2. Average accuracy over rounds with the attack in Conventional FL vs Average accuracy over rounds with the attack in FL with PID – put these metrics on one plot to better showcase the difference.

Elaborate on the results that your plots demonstrate. In your presentation, try to identify the most interesting aspects of your project work and results, and spend most of your time presenting and discussing them.

Attachment 1: FL Background Information

The evolution of Communications, Data Science, and Machine Learning (ML) fields, as well as growing security and privacy protection requirements, have resulted in the emergence of the Federated Learning (FL) paradigm (1). First introduced by Google to improve smartphone keyboard query suggestions (2), FL enables privacy and security enhancement by keeping training data locally and improving data communication efficiency by not transferring huge amounts of data to the aggregation unit (3). The FL goal of privacy protection is accomplished mainly by communicating only the local models instead of the original data to the aggregation units. Other techniques for further security and privacy enhancement have been proposed and incorporated into the FL process, such as:

- Employing more robust aggregation functions, e.g., Geometric Median (4) or Trimmed Mean (5) instead of FedAvg (1).
- Federated Learning Based on Dynamic Regularization (6).
- Personalized Federated Learning by Pruning (7).
- Employing a root dataset to verify local models (8).
- Selecting a global model from the set of local models based on the square-distance score (9).
- Differential privacy approaches (10).
- Secure aggregation approaches based on updates verification using cryptographic primitives (11; 12; 13).

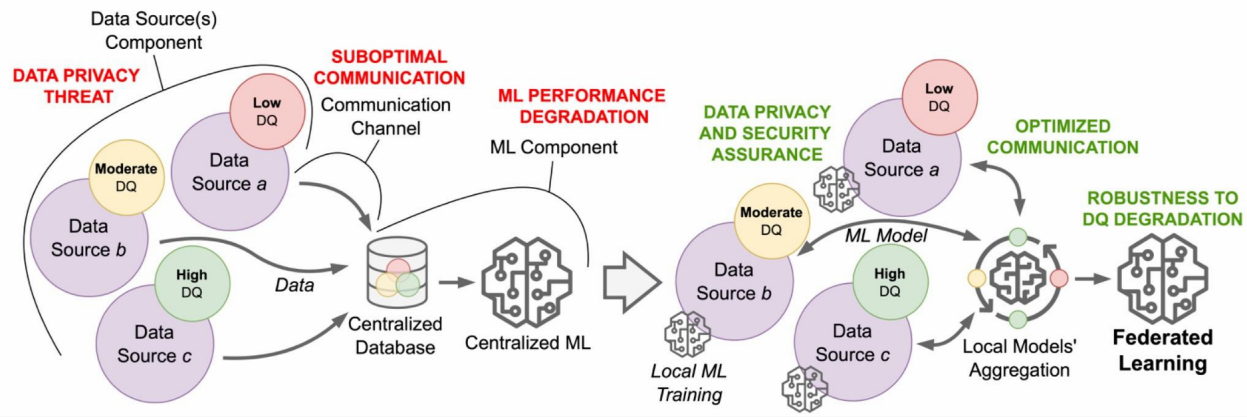

Figure 1: High-level architectures of both centralized ML and conventional FL, with major weaknesses of the former architecture, and the inherent advantages of the latter one

In Figure 1, we show the high-level architectures of both centralized ML and conventional FL, indicate major weaknesses of the former architecture, and depict the inherent advantages of the latter one. Despite having been intensively researched and improved, conventional FL still demonstrates its vulnerabilities. The conventional FL process (14) includes the action of sending back the global model updates from the aggregation unit to local units in each aggregation round without considering the possibility that these local units might already be compromised by an adversary, resulting in

a major privacy violation. In this case, the adversary might gain access to these global updates and apply data inference attacks to derive sensitive information. This information leakage may violate the privacy of local units participating in the FL system. These attacks might be employed to infer the data stored on the local unit, which poses obvious privacy concerns (15). Various studies have addressed malicious update detection and mitigation (16; 9; 17). However, the conventional FL operation flow (14) does not envisage the possibility of excluding compromised local units from the global updates distribution and requires additional solutions on top of it. This challenge needs to be addressed to enhance the trustworthiness of the FL process from the end-user's perspective and the prospects of its applications in industry.

Although FL has some default security and privacy mechanisms, recent research has demonstrated that these mechanisms are insufficient against some advanced attack strategies aimed at distilling sensitive information about the training data by analyzing the ML model only (18; 19; 20; 21). Generally, when applied in the FL environment, membership attacks allow the adversary to employ the ML model parameters to reconstruct the models trained by the local units to derive information on the data they were trained on (22; 23; 24). An alternative approach for the attacker, who is able to reconstruct the ML model trained at the local unit, is to supply a set of inputs to this model to detect if these inputs are part of the collection used during the learning stage (25). In general, the attack model assumes the adversary has at least partial knowledge of the training collection samples. However, Suri et al. (25) demonstrated that, in some cases, membership can be inferred even without any knowledge of the training data by using samples of a similar distribution, which can be referred to as subject membership. This makes such attacks even riskier in terms of individual privacy, as there is no requirement that the samples employed by an adversary to infer the membership must be present in the training collection. The leaked gradient updates and other ML model parameters might be used by the adversary to partially or fully reconstruct the samples from the training collection (26; 27; 28). In their recent research, Geng et al. (29) successfully reconstructed the original image samples from the CIFAR-10 (30) and ImageNet (31) datasets using a zero-shot batch label restoration approach. They also showed that the reconstruction accuracy depends on how long the model was trained on the local unit, and the integration of ML model parameters from multiple local units can effectively improve the image recovery. Hence, additional security measures are needed to identify the malicious attackers in FL and exclude them from obtaining the model updates.

Attachment 2: Federated Learning Data Collections

Several datasets have been specifically designed or adapted for research in Federated Learning (FL). These datasets often cater to the decentralized and heterogeneous nature of FL environments. Here are some notable data collections that are widely used in FL research:

1. **LEAF (A Benchmark for Federated Settings)**
    - **Description:** LEAF provides a suite of benchmark datasets tailored for federated learning research. It offers datasets that are naturally partitioned in a way that reflects real-world FL scenarios, such as non-IID (non-identically distributed) data across clients.
    - **Link:** https://github.com/TalwalkarLab/leaf
    - **Datasets Included:**
        - Sent140 (Twitter Sentiment): A federated version of sentiment analysis on tweets, split by user.

  - Shakespeare: A dataset based on the works of Shakespeare, split by character.
  - CelebA: A large-scale face attributes dataset, split by identity.
- **Use Case:** Ideal for benchmarking FL algorithms on tasks like image classification, sentiment analysis, and language modeling.

2. **Google's Federated Learning Datasets**

- **Description:** Google has made several datasets available for FL research, designed to mimic real-world applications.
- **Datasets Included:**
  - FEMNIST: An extended version of the MNIST dataset, partitioned to simulate a federated environment.
  - StackOverflow: A dataset for next-word prediction and tag prediction, split by user activity.
  - Landmarks User Data: A dataset for image classification tasks in FL, representing different landmarks.
- **Use Case:** Suitable for studies on user personalization, model heterogeneity, and communication efficiency.

3. **Medical Imaging Datasets**

- **Federated Tumor Segmentation (FeTS) Challenge:** The FeTS initiative provides a federated learning framework for the segmentation of brain tumors across different institutions.
- **Medical Segmentation Decathlon:** This dataset includes ten different medical imaging tasks such as brain, liver, and lung segmentation, distributed across institutions.
- **TCIA (The Cancer Imaging Archive):** TCIA provides a wealth of medical imaging datasets, often associated with different institutions, making them suitable for distributed training.
- **Federated Learning for Melanoma Detection:** The ISIC dataset for melanoma detection has been adapted for federated learning, where data is distributed across different dermatology centers.

4. **CIFAR-10/100 Federated Version:**

- **Description:** CIFAR-10 and CIFAR-100 are popular datasets for image classification tasks. Federated versions of these datasets are often used to simulate non-IID scenarios.
- **Link:** https://github.com/TalwalkarLab/leaf
- **Use Case:** Useful for benchmarking FL algorithms in image recognition tasks with varying degrees of data heterogeneity.

5. **FLAIR: Federated Learning Annotated Image Repository**

- **Description:** A challenging large-scale annotated image dataset for multi-label classification suitable for federated learning.
- **Link:** https://machinelearning.apple.com/research/flair
- **Use Case:** Annotated Image Repository.

These datasets provide a diverse range of use cases for Federated Learning research, from image classification to natural language processing and personalized recommendation systems. They are often accompanied by tools and frameworks that help researchers simulate FL environments, making them ideal for developing and testing FL algorithms.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics. PMLR*, 2017, pp. 1273–1282.

[2] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," arXiv preprint arXiv:1812.02903, 2018.

[3] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140 699–140 725, 2020.

[4] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," arXiv preprint arXiv:1912.13445, 2019.

[5] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning. PMLR*, 2018, pp. 5650–5659.

[6] D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," arXiv preprint arXiv:2111.04263, 2021.

[7] S. Vahidian, M. Morafah, and B. Lin, "Personalized federated learning by structured and unstructured pruning under data heterogeneity," in 2021 IEEE 41st International Conference on Distributed Computing Systems Workshops (ICDCSW). IEEE, 2021, pp. 27–34.

[8] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," arXiv preprint arXiv:2012.13995, 2020.

[9] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[10] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, "Protection against reconstruction and its applications in private federated learning," arXiv preprint arXiv:1812.00984, 2018.

[11] A. Roy Chowdhury, C. Guo, S. Jha, and L. van der Maaten, "Eiffel: Ensuring integrity for federated learning," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 2535–2549.

[12] H. Lycklama, L. Burkhalter, A. Viand, N. Küchler, and A. Hithnawi, "Rofl: Robustness of secure federated learning," in 2023 IEEE Symposium on Security and Privacy (SP). IEEE, 2023, pp. 453–476.

[13] M. Rathee, C. Shen, S. Wagh, and R. A. Popa, "Elsa: Secure aggregation for federated learning with malicious actors," in 2023 IEEE Symposium on Security and Privacy (SP). IEEE, 2023, pp. 1961–1979.

[14] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," arXiv preprint arXiv:1610.05492, 2016.

[15] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in 2019 IEEE symposium on security and privacy (SP). IEEE, 2019, pp. 739–753.

[16] D. Cao, S. Chang, Z. Lin, G. Liu, and D. Sun, "Understanding distributed poisoning attack in federated learning," in 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 2019, pp. 233–239.

[17] S. Shen, S. Tople, and P. Saxena, "Auror: Defending against poisoning attacks in collaborative deep learning systems," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, 2016, pp. 508–519.

[18] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "ML-leaks: Model and data independent membership inference attacks and defenses on machine learning models," arXiv preprint arXiv:1806.01246, 2018.

[19] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in neural information processing systems*, vol. 32, 2019.

[20] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 603–618.

[21] F. Boenisch, A. Dziedzic, R. Schuster, A. S. Shamsabadi, I. Shumailov, and N. Papernot, "When the curious abandon honesty: Federated learning is not private," in 2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P). IEEE, 2023, pp. 175–199.

[22] W. Wei, L. Liu, M. Loper, K.-H. Chow, M. E. Gursoy, S. Truex, and Y. Wu, "A framework for evaluating gradient leakage attacks in federated learning," arXiv preprint arXiv:2004.10397, 2020.

[23] X. Xu, J. Wu, M. Yang, T. Luo, X. Duan, W. Li, Y. Wu, and B. Wu, "Information leakage by model weights on federated learning," in *Proceedings of the 2020 workshop on privacy-preserving machine learning in practice*, 2020, pp. 31–36.

[24] A. Pustozerova and R. Mayer, "Information leaks in federated learning," in *Proceedings of the Network and Distributed System Security Symposium*, vol. 10, 2020, p. 122.

[25] A. Suri, P. Kanani, V. J. Marathe, and D. W. Peterson, "Subject membership inference attacks in federated learning," arXiv preprint arXiv:2206.03317, 2022.

[26] B. Zhao, K. R. Mopuri, and H. Bilen, "IDLG: Improved deep leakage from gradients," arXiv preprint arXiv:2001.02610, 2020.

[27] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients: How easy is it to break privacy in federated learning?" *Advances in Neural Information Processing Systems*, vol. 33, pp. 6937–6947, 2020.

[28] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, "See through gradients: Image batch recovery via gradinversion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16,337–16,346.

[29] J. Geng, Y. Mou, F. Li, Q. Li, O. Beyan, S. Decker, and C. Rong, "Towards general deep leakage in federated learning," arXiv preprint arXiv:2110.09074, 2021.

[30] A. Krizhevsky, V. Nair, and G. Hinton, "CIFAR-10 (Canadian Institute for Advanced Research)," accessed on July 30, 2023. Available: http://www.cs.toronto.edu/~kriz/cifar.html.

[31] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, IEEE, 2009, pp. 248–255.