Data Analytics & Cyber Security

Team Project & Group Dynamics Module

Network Intrusion Detection System

Technical Review & Design Document

to be submitted on: 23/11/2022

Ingrid Melin – K00258766

Patryk Kaiser – K00263702

Daniel Mackey – K00263905

Project Supervisor(s): Tom Davis and Aileen O'Mara

Contents

Objective

Our system is being built to demonstrate a proof of concept Network Intrusion Detection System (IDS).

We aim to have a system which, when deployed in a network environment, can detect, and classify cyber-attacks.

To perform detection, we will use a trained Machine Learning (ML) algorithm.

The objective is to develop a packet/network flow capture program, a trained ML model, and finally a front-end website which will display alerts and network traffic information.

We are building this system because we believe that data analytics and cyber security together can create systems which will keep business and individual assets safe from malicious parties.

Through early detection of scans that usually precede full scale attacks our system will be able to alert network administrators to a potential impending attack.

Requirements

Functional Requirements

- A monitoring system which will store network flow information. Figures 1 and 2 show the use case in two different levels of complexity.
- A prediction program using our Machine Learning model which will generate alerts on detected threats. Figure 2 shows the use case for the prediction program working together with the monitoring program.
- A database used to hold alerts issued by the prediction program.
- A web front end to display alerts generated by the prediction program. The use case for this requirement can be seen in figure 3.
- We want our program to run on Windows, Linux, and Mac.

Non-Functional requirements

- Machine Learning prediction accuracy of >= 90%.
- Eliminating false positives is impossible so we aim to keep false positives to an acceptable level. With an accuracy of ~90% we hope false positives will not be a huge problem. We aim to help stop false positives by supplementing the benign labelled data with normal traffic flow data from our own home networks. This will hopefully give the program a diverse and modern set of data to train on.
- Sanitizing inputs into the front end.
- Hashing and salting user login information for the front end.
- The monitoring system should not impede network performance.
- The prediction program should not hugely affect hardware performance.

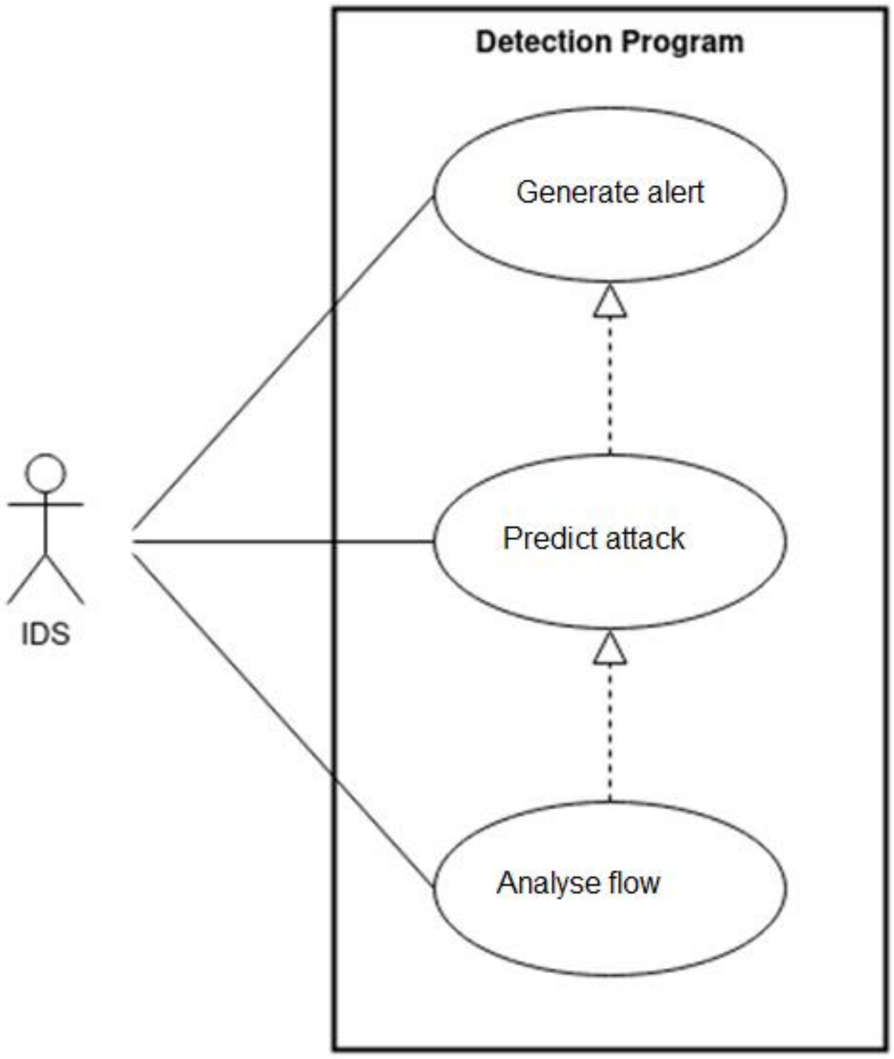

Figure 1: The use case diagram for the detection program. This use case needed to be updated to fit in with us splitting the program up into separate parts. We have decided to leave this use case in to show the functionality of the program as one whole piece.

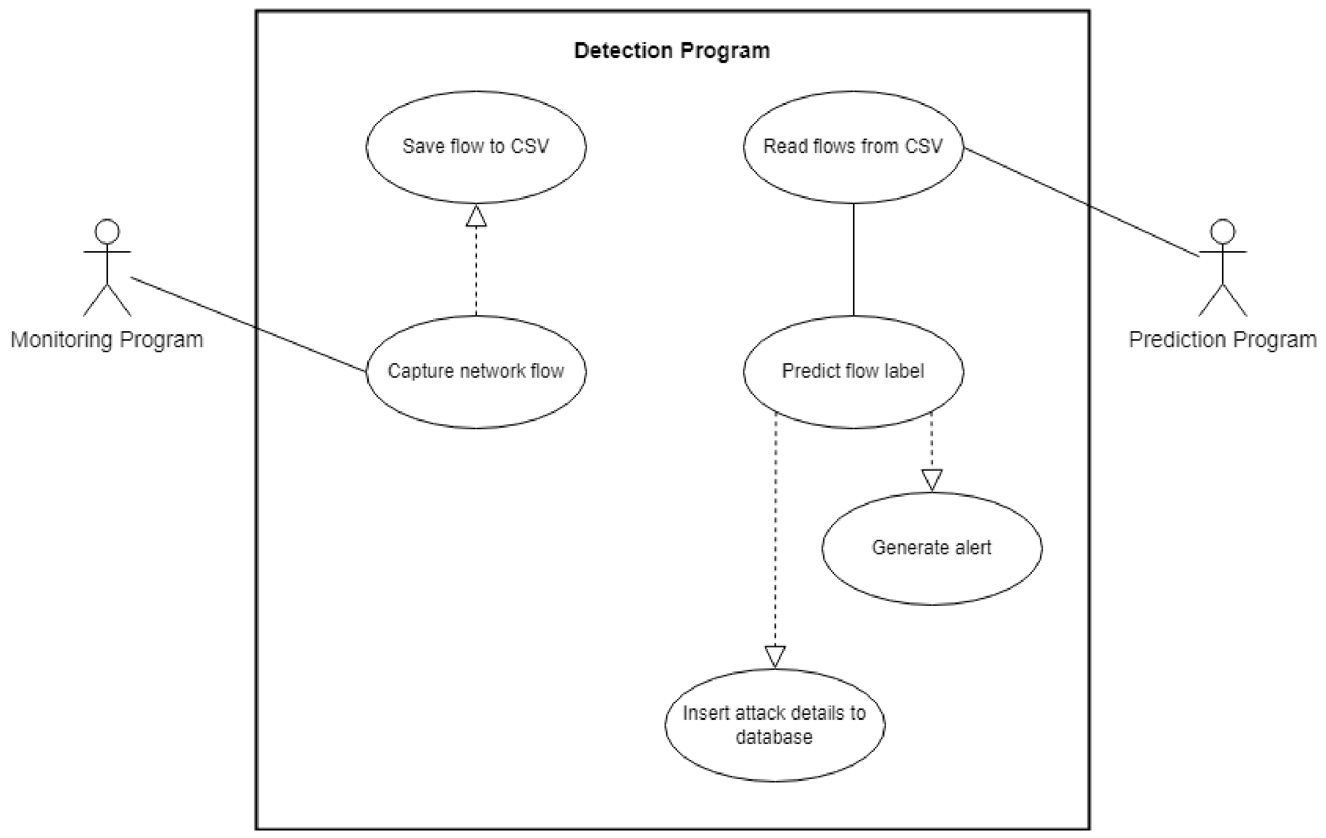

Figure 2: This use case shows the two programs (one for monitoring/capturing network flow and another for running predictions on the data) working together to provide the intrusion detection functionality.

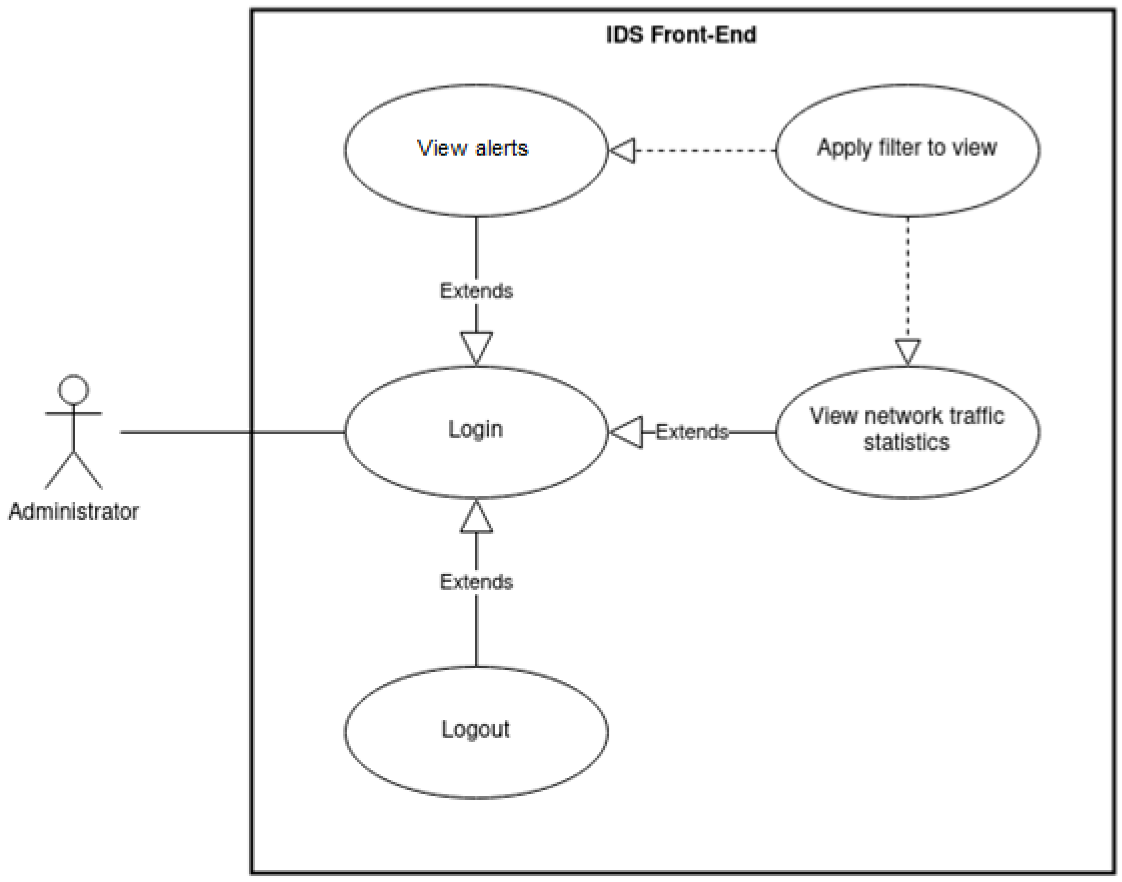

Figure 3: The front-end use case. This use case shows the functionality a network administrator can get from using the front-end.

Architectural Design

Figure 1 shows the flowchart for the monitoring and prediction programs working together.

Figure 2 shows the flowchart for the Supervised Machine Learning program.

Figure 3 shows the proposed location for the deployment of our system on a network.

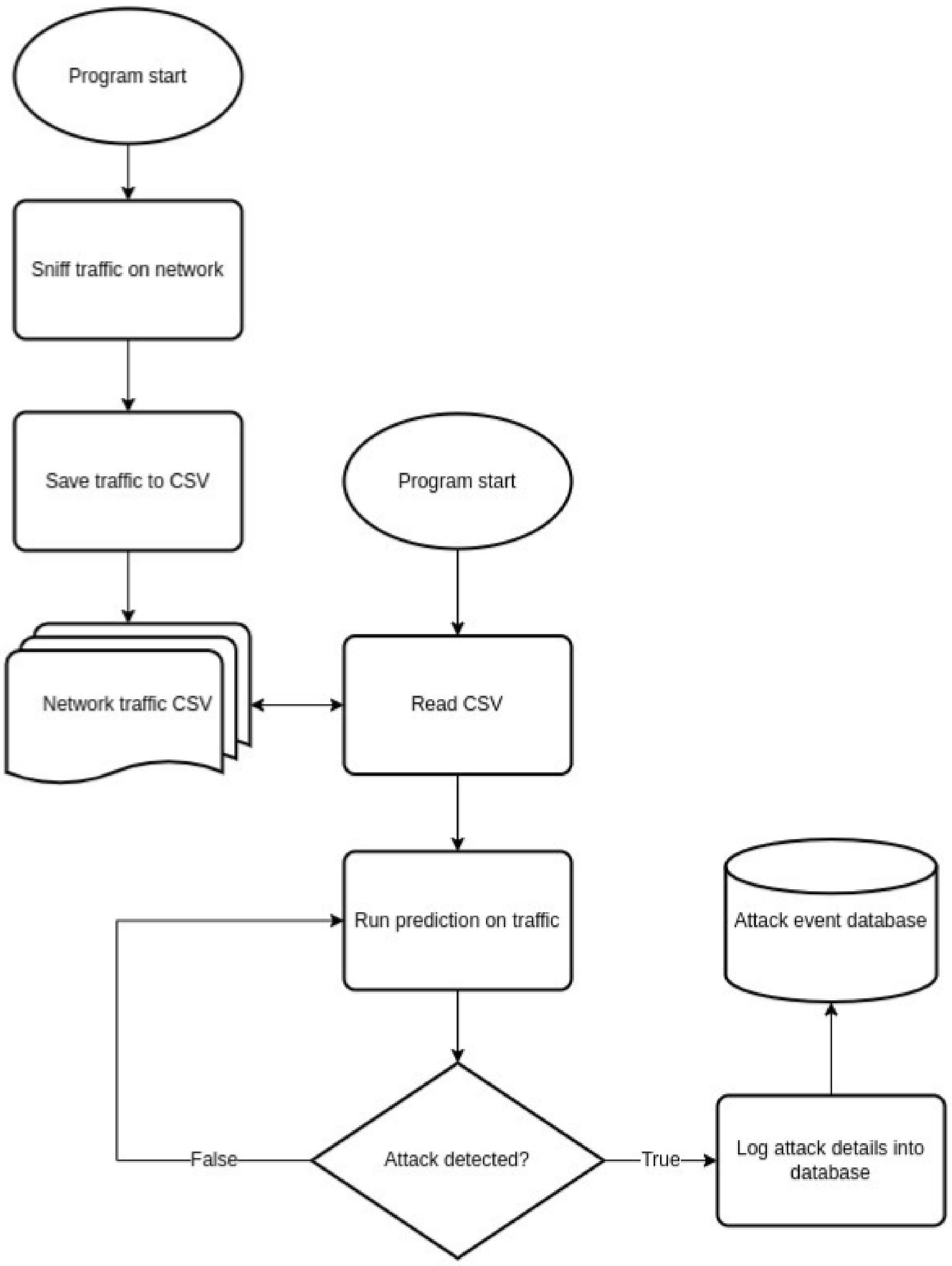

Figure 4: Flowchart showing the monitoring program and prediction program working together to predict attacks. The monitoring program will save traffic flow to a CSV which will then be read by the prediction program. The prediction program will send the details of the detected attack to the attack event database which will be used to generate an alert.

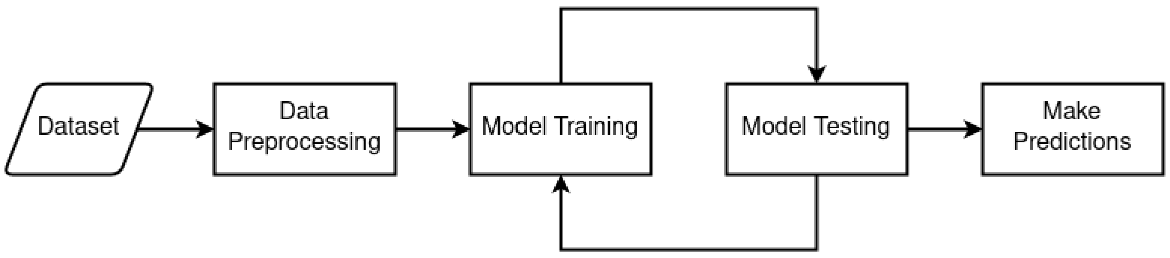

Figure 5: The Supervised Machine Learning program flowchart. First we pre-process the dataset and then use it to train and test the program. Once the program reaches a satisfactory level of accuracy we can use it in the real world to make predictions on live network traffic flow.

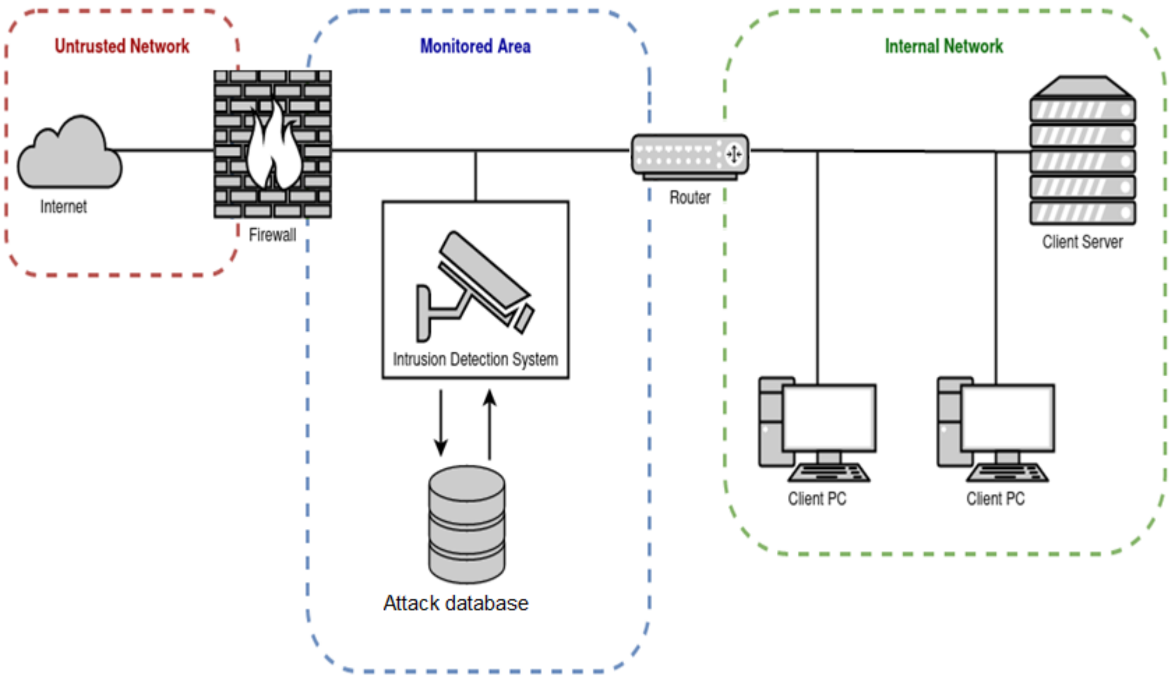

Figure 6: The proposed location of our system on a network. Our research has shown that the system will perform best when placed between the firewall and router. The firewall will block and allow traffic based on the rules provided by the network administrator. If anything manages to sneak past the firewall it will be detected by the intrusion detection system. This will hopefully also minimize false positives by feeding already filtered data into the prediction program.

3 Tier Architecture

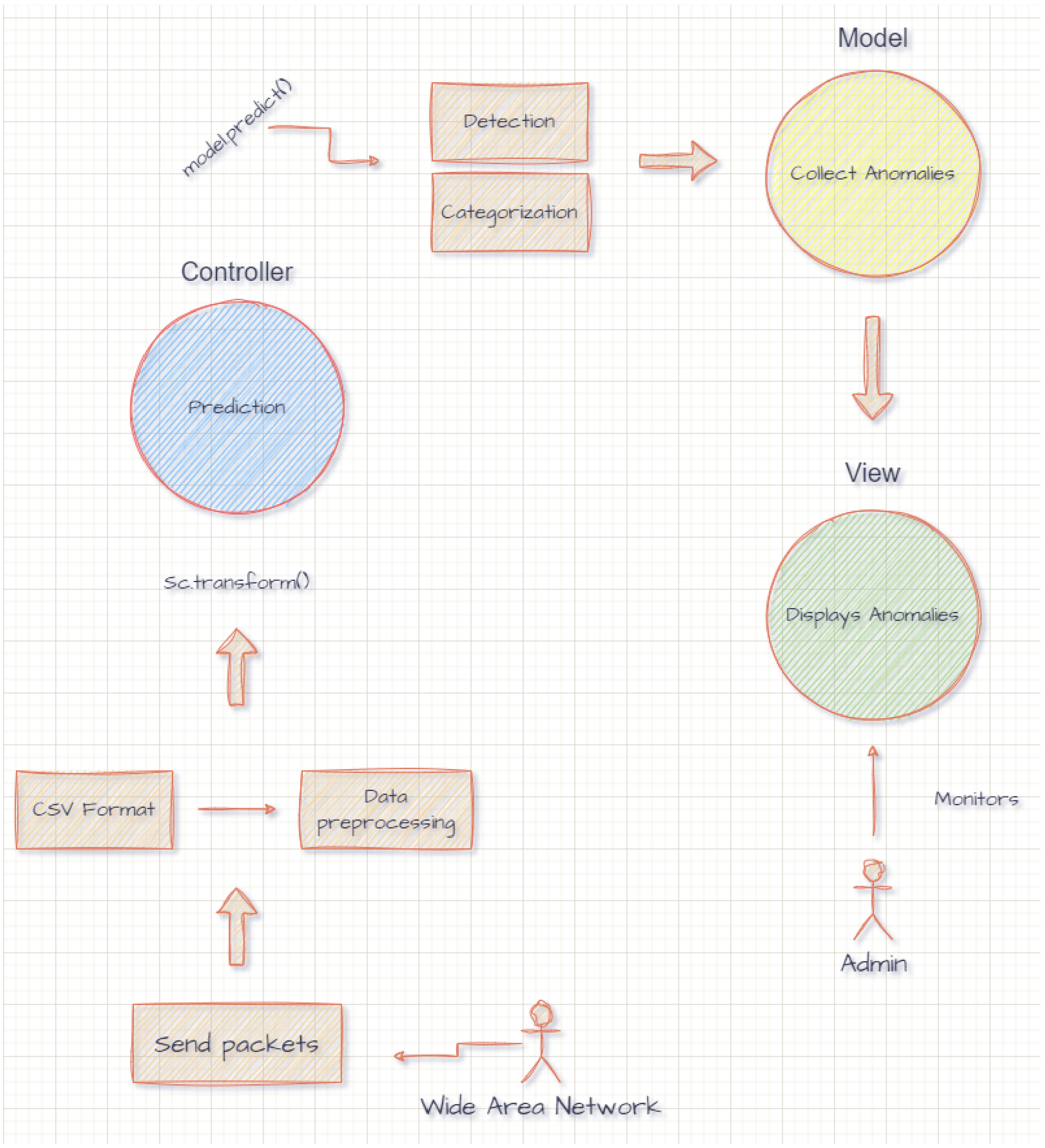

Figure 7: MVC (Model – View – Controller) Architecture

3 interconnected components that show the important detail, with straight forward interaction between the backend and frontend functionality of our project and how they come together to accomplish our goals.

The controller is responsible for being the brain, unlike the view and model. The controller must be able to execute the sequential algorithm, scaling and fitting the sniffed wide area network data to create a desired SML model to classify future detections into their relevant categories for the model architecture component to handle and store, which the view component shows to the administrator on duty via the local area network and its hosted user interface.

User Interface

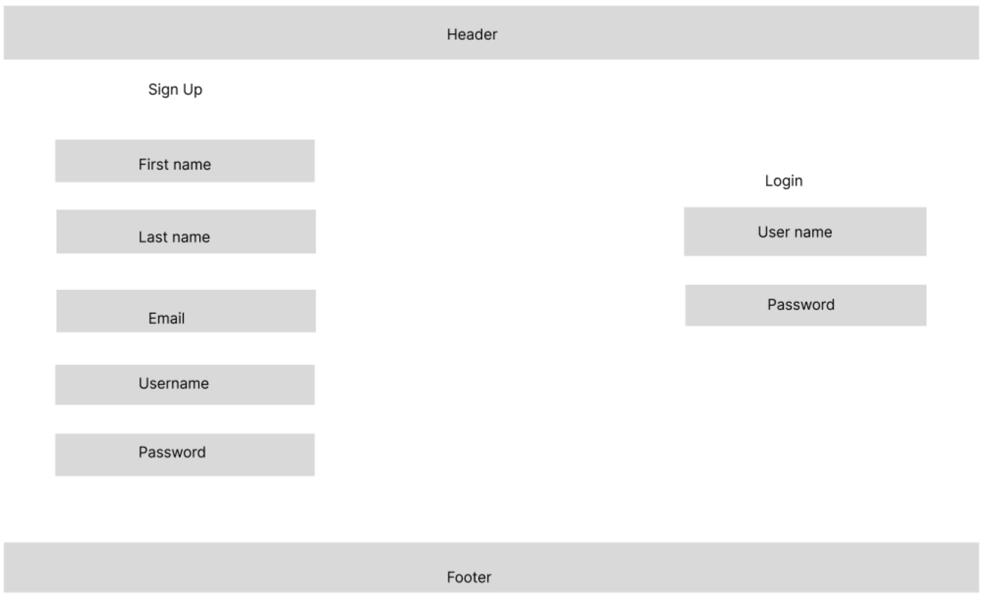

Figure 8: Wireframe for the sign up and login screen.

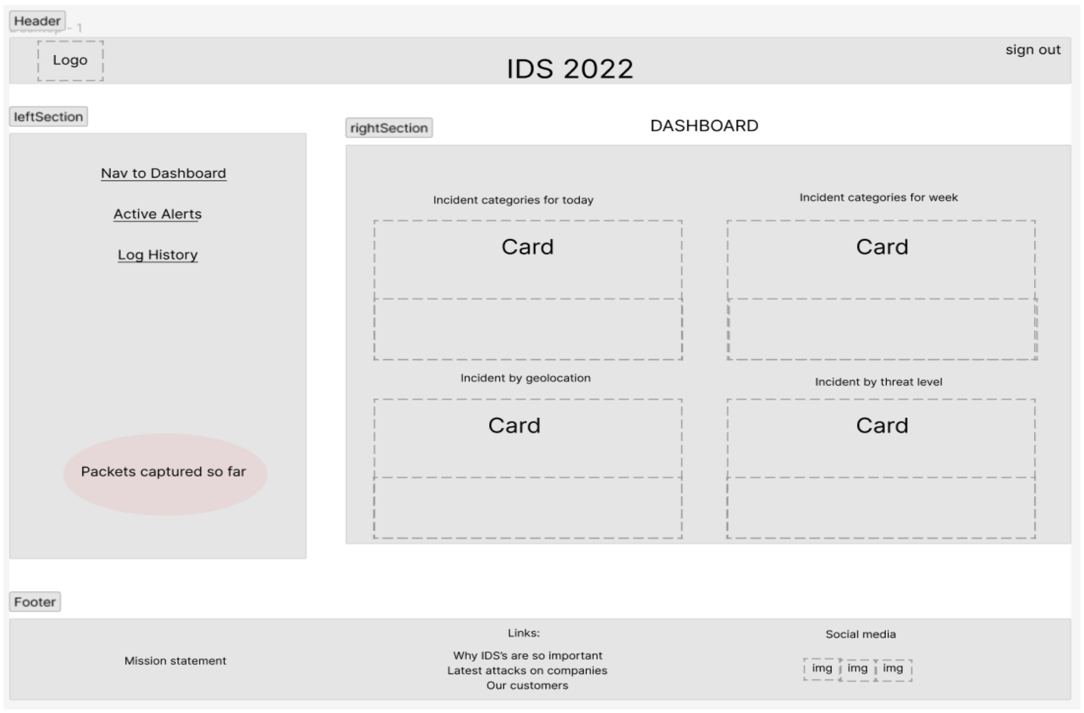

Figure 9: Wireframe for the dashboard. This will display any active alerts and statistics.

Technical Review

Traffic Monitoring Program:

- Python - PyShark, Scapy, Pandas, NumPy

Machine Learning Prediction Program:

- Python - TensorFlow, Scikit-learn, Pandas, NumPy

Front-End:

- PHP
- HTML, CSS
- MySQL
- AJAX
- Google Charts - Can interface with PHP to display visualizations
- FusionCharts - Interfacing with PHP to display visualizations

Database:

- MySQL

The dataset we are using to train our Machine Learning model has been generated using an open-source tool called CICFlowMeter, this tool is developed by the Canadian Institute of Cybersecurity. We have reached out to the creators of this tool and have been provided with adequate documentation and research papers. This extra documentation along with the already adequate information available on GitHub and the CIC website will be used by us to construct network flow details to a high degree of accuracy matching the data used to train the Machine Learning model. This is done to improve prediction/detection accuracy.

If we are unable to construct network flow in the exact format of the training dataset we have two alternatives:

1. Strip down the training dataset to include only columns which we can generate using our monitoring program.
2. Use the CICFlowMeter to generate the network flows.

We originally planned to use JavaScript/TypeScript with the Angular framework. Upon further consideration we decided to use PHP.

We aim to properly sanitize MySQL inputs and are aware of the inherent security flaws in PHP. However, we have also spent a lot of time in year 2 of this course researching PHP vulnerabilities and gaining experience removing them from our code.

The reason we are considering PHP instead of Angular is because:

1. PHP would be more suited to the scale of our front-end. The front-end will not provide major functionality apart from displaying alerts and statistics.
2. We have more experience using PHP and are more comfortable using it.

3. PHP websites are easier to host and there is a variety of free hosting sites out there that we have had experience using before.
4. PHP is executed on the server side rather than client-side.

Another alternative to the front end we have discussed is using Power Bi dashboards. We are keeping an open mind to our choices and are working with small scale prototypes of each to make the most suitable choice.

So far we have:

1. A Machine Learning prediction model which reads data live from the CICFlowMeter and predicts the label of each network flow. We have captured 'benign' traffic from our home networks and mixed it in with the dataset. This improved the accuracy as we have basically provided more data used to distinguish between benign traffic and malicious traffic.
2. A Python script using PyShark which captures packets live and runs predictions on TCP flags. This uses a different ML model trained only on a limited set of columns which include TCP flags and source/destination port numbers. This is obviously not ideal as a lot of information which can be used to identify attacks is missing. However, this program has shown to be 90% accurate in detecting TCP scans.

The Python script is being improved by now attempting to generate network flow information to match more columns present in the training dataset.

Data Storage – Model ERD

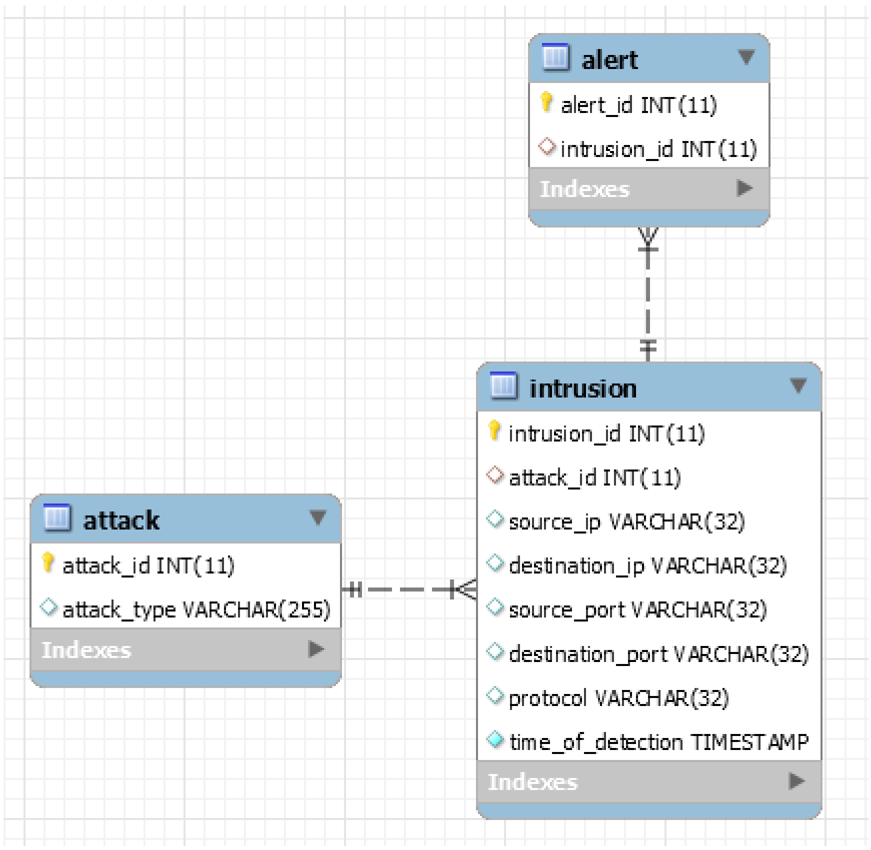

Figure 10: The ERD for our attack database. This database will feed information into the front-end. Only network flows that are interpreted as an attack will be stored in this database.