|  |  |
| --- | --- |
| Insert title | |
|  | |
| Student | Main supervisor |
| - Luca Lucchina | - Angelo Consoli |
| Co-supervisors |
| - |
| Company (if any) |
| - |
| Degree course | Project identifier (if any) |
| Data Science and Artificial Intelligence | - |
| Year |  |
| * 2025 |  |

Table of Contents

[Abstract 3](#_Toc206164093)

[Chapter 1: Omniport Data Exploration 4](#_Toc206164094)

[1.1 Geographical features 4](#_Toc206164095)

[1.1.1 Location based patterns 5](#_Toc206164096)

[1.2 Time based features 6](#_Toc206164097)

[1.3 Payload Features 8](#_Toc206164098)

[1.4 Text embedding 9](#_Toc206164099)

[1.5 Categorical encoding 10](#_Toc206164100)

[Chapter 2: AI unsupervised approach 10](#_Toc206164101)

[2.1 Full dataset Clustering 10](#_Toc206164102)

[2.2 Grouped dataset 11](#_Toc206164103)

[2.2.1 Feature selection and test 12](#_Toc206164104)

[Chapter 3: AI semi supervised approach 13](#_Toc206164105)

[3.1 Model trained on the cic and predicting on omniport 13](#_Toc206164106)

[3.2 Pseudo-Labelling configuration on CIC IDS 2017 13](#_Toc206164107)

[3.2.1 Initial Approach 13](#_Toc206164108)

[3.2.2 Consistency Regularization 14](#_Toc206164109)

[3.2.3 Curriculum Learning 14](#_Toc206164110)

[3.2.4 High-Confidence Sample Selection 14](#_Toc206164111)

[3.2.5 Fixing Class Imbalance in Pseudo-Labels 16](#_Toc206164112)

[3.2.6 Reducing Number of Classes 18](#_Toc206164113)

[3.2.7 KL Divergence for Pseudo-Labelled Samples 19](#_Toc206164114)

[3.3 Omniport Pseudo-Labelling Implementation 22](#_Toc206164115)

[3.3.1 Evaluation method 25](#_Toc206164116)

[Chapter 4: Payload analysis 25](#_Toc206164117)

[Chapter 5: Results 26](#_Toc206164118)

[5.1 Structure of the Results Section 26](#_Toc206164119)

[5.2 Results Table 27](#_Toc206164120)

[5.3 Step-by-Step Evaluation 27](#_Toc206164121)

[5.4 Narrative Summary 28](#_Toc206164122)

# Abstract

Cyberattacks have become highly sophisticated mechanisms used daily to halt the normal functioning of electronic systems in all sectors of the economy. The primary objective of this project was to analyse and classify a large set of unlabelled cyberattacks, leveraging payload characteristics to determine their type and key attributes. One of the main challenges was that unsupervised learning approaches proved ineffective due to the absence of well-defined cluster boundaries in labelled reference datasets. This limitation motivated the adoption of a semi-supervised learning strategy, which yielded improved classification performance, but was still constrained by the lack of ground-truth labels and the necessity of relying on an external, Kaggle-sourced labelled dataset, introducing additional uncertainty.

To enhance attack understanding and dataset analysis, the study examined multiple features beyond payload content, including attack timings, geographic origins, ASNs, and port identifiers. By integrating these features, we developed a robust pseudo-labelling methodology that, despite inherent uncertainty, provides sufficiently reliable labels to identify and isolate attacks. This could allow automatic alerts to security administrators and the activation of appropriate defensive protocols upon the detection of new attacks.

ITALIANO

Cyberattacks sono diventati dei meccanismi molto sofisticati utilizzati quotidianamente per mettere in crisi sistemi informatici di tutti i settori dell’economia. L’obiettivo del nostro progetto è di analizzare e classificare automaticamente una serie di cyberattacks non labelizzati utilizzando le caratteristiche dei payloads per identificare il tipo di attacco e i suoi dettagli. Un aspetto critico risulta essere la natura stessa del cyberattach, in quando un approcio IA unsupervised si è dimostrato inefficace in quando i datasets labelled non permettono di avere dei cluster definiti con chiarezza. Per questo motivo, ci siamo focalizzati su un metodo semi-supervised che si è rivelato migliore: cio’ nonostante la mancanza di label e il fare ricorso a datasets externi con label, hanno introdotto un livello di incertezza; abbiamo utilizzato un labelled dataset di Kaggle che ha provocato gli errori e le incertezze. Al fine di ottimizzare la comprensione degli attacchi e una migliore analise dei dataset, abbiamo studiato altri parametri oltre al payload: timing degli attacchi, provenienza, sequenza, asynch number, port ID.

Con questa estensione al di là dell’analisi del paylod unicamente, siamo riusciti ad ottenere un’analisi estesa che ci ha permesso di lavorare con un data-set pseudo-labelled che, pur non essendo certo al 100%, è comunque molto affidabile per isolare ed identificare gli attacchi e informare l’amministratore dei sicurezza in modo automatico su nuovi attacchi.

# Chapter 1: Omniport Data Exploration

The Omniport dataset, provided by my supervisor Angelo Consoli, consists of nearly one million recorded interactions between cyber attackers and honeypot servers deployed in cities such as Singapore, Paris, Mexico City, and New York and various other locations worldwide. A honeypot is a deliberately exposed, isolated system or service designed to attract cyber attackers, allowing researchers to observe malicious activity without putting real systems at risk.

## Geographical features

The data included the IP of the source and the destination, the country and city for both as well as the coordinates. Looking at the map shown in figure 1 we can see that there is little observable trend in the distribution of attack locations.

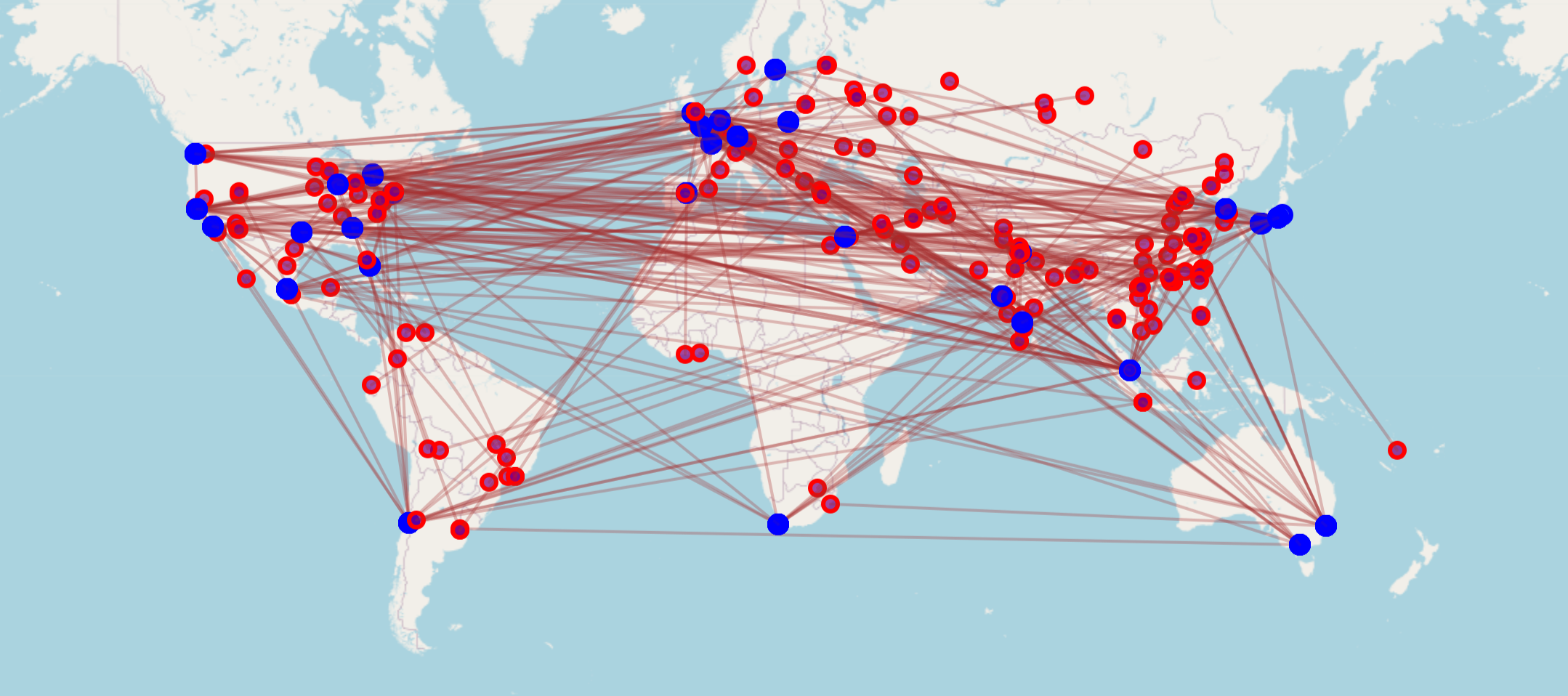


Fig 1: red dots being attackers and blue dots the honeypot servers

For the future models benefit, I kept the IP addresses but, split them into quartets so that each quartet is treated as its own feature, allowing each component to contribute independently. For the latitude and longitude, I applied cosine and sine transformations to remove the linearity of degree values ranging from −180 to 180, which effectively captures the circular nature of geographic coordinates and allows the model to better interpret proximity across the 360° boundary.

### 1.1.1 Location based patterns

China and the USA are the most represented countries being attacked, with India and the Netherlands not far behind. This is due to the high number of servers in them leading to, naturally, more attacks, as shown in Figures 2 and 3.

A graph of a number of people

AI-generated content may be incorrect.

Fig 2: Graph of outgoing attacks, showing high number of outgoing attacks for USA and China mainly

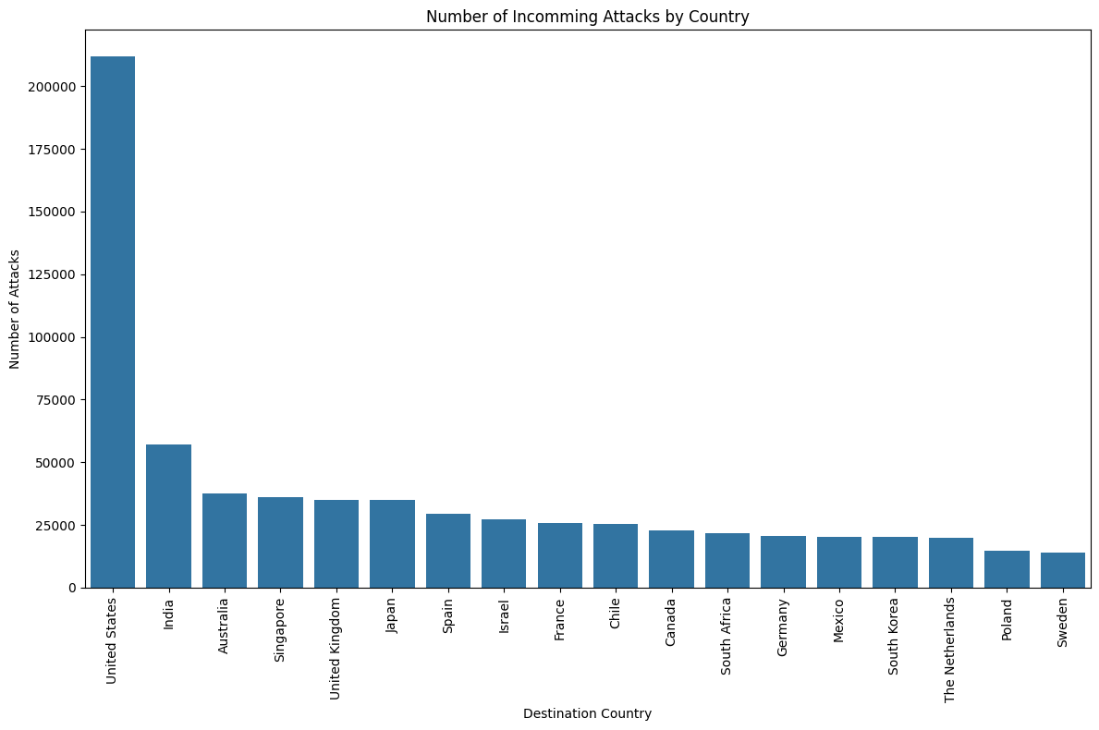


Fig 3: graph of incoming attacks, showcasing high number of incoming attacks for USA and India

## Time based features

Time could be a very important element for prediction. Humans inherently have patterns and preferences whilst bot network don’t. The issue that I came to find is that there aren’t any major patterns when looking at UTC time. I thought there would be patterns with local time of attack: at night or non-workable days but looking at figure 4 we can see that such patterns are only very slightly present. The timing for cyber attacks is evenly spread across the week.

A close-up of a graph

AI-generated content may be incorrect.

Fig 4: distribution of attacks by hour of day and day of the week

Bot networks are programmed to set intervals between cyber-attacks. Compared to human attack timing which can vary drastically. Therefore, I looked at the time between individual attacks separating it the following classes: sub 1 second, 1-60 second, 1-60 minutes, 1-24 hours, >24 hours and unknown (occurring when only one interaction with this IP was recorded in the dataset). The median time and class distribution for the top 10 countries can be seen in figure 5.

* Sub-second (<1 second)
  + Attacks occurring less than 1 second apart usually indicates automated/bot attacks or DDoS attempts: very high frequency, likely machine-generated traffic
* 1-60 seconds
  + Attacks occurring between 1-60 seconds apart often indicates automated scanning or systematic probing probably too fast for human operation
* 1-60 minutes
  + Attacks occurring between 1-60 minutes apart could be both automated (with delays) or human-driven may indicate more sophisticated scanning with rate limiting
* 1-24 hours
  + Attacks occurring between 1-24 hours apart more likely to be human-driven or scheduled automated tasks could indicate daily routines or scheduled scans
* >24 hours
  + Attacks occurring more than 24 hours apart usually indicates sporadic attempts or long-term reconnaissance more likely to be human-driven or opportunistic attacks

A graph of different colored bars

AI-generated content may be incorrect.

Fig 5: attack timings for the top 10 countries in the dataset.

## Payload Features

The payload was handled in two ways: feature extraction from the encoded payload, and analysis of command occurrences. For encoded features, I extracted payload length (pl\_length), unique bytes (pl\_unique\_bytes), entropy (pl\_entropy), average byte value (pl\_mean\_byte), and byte standard deviation (pl\_std\_byte).

A white background with text on it

AI-generated content may be incorrect.

Fig 6: Box plot distribution of payload features

This illustrating the distribution of payload features, reveals several key insights. The pl\_length feature exhibits a highly skewed distribution with numerous outliers, indicating a subset of payloads with significantly larger sizes compared to the norm. In contrast, pl\_entropy, pl\_mean\_byte, and pl\_std\_byte show more compressed distributions, with fewer extreme values. The compact nature of pl\_entropy may reflect relative uniformity in the composition of the bytes across most of the payloads.

A screenshot of a graph

AI-generated content may be incorrect.

Fig 7: correlation matrix of payload features

This presents the correlation matrix of the extracted payload features. Key observations include a strong positive correlation between pl\_length and pl\_unique\_bytes (0.88), suggesting that as the payload length increases, the number of unique bytes within the payload tends to increase proportionally. Similarly, there's a notable positive correlation between pl\_mean\_byte and pl\_std\_byte (0.85). This indicates that payloads with higher average byte values tend to exhibit greater variability in their byte composition. Additionally, pl\_entropy shows moderate positive correlations with pl\_unique\_bytes (0.69), pl\_mean\_byte (0.86), and pl\_std\_byte (0.76), suggesting that payloads with higher randomness in their byte distribution also tend to have more unique bytes, higher average byte values, and greater byte value variability.

## Text embedding

The dataset contained two primary sources of text amenable to embedding: the decoded payloads and the location-based attributes (city, country, and region names). Embedding the location features was considered promising, as a language model (LLM) could potentially discern non-spatial patterns and enrich the data with additional contextual details. The payload embedding, however, presented more significant challenges. These challenges were due to a combination of factors, including corrupted and empty payloads, as well as the need for an LLM embedding method tailored to code. CodeBERT emerged as the most suitable option for the payload embedding, being a pre-trained model on Python, Java, JavaScript, PHP, Ruby, and Go. While CodeBERT wasn't trained on bash script, the language typically used in cyber-attacks, its knowledge of coding synthetics provided a substantial advantage over general-purpose language models. However, this method was only sparingly used in the payload analysis due to the prohibitively long inference time required for embedding, which ultimately outweighed its potential benefits for this project. Companies such as CISCO have implemented and are using LLMs fine-tuned on cyber-attacks as primary protection.

## 1.5 Categorical encoding

Given the importance of network source information, the Autonomous System Number (ASN), represented by 'src\_as', was encoded using a frequency-based approach. The number of occurrences of each ASN was mapped to a new feature, 'src\_as\_encoded', to capture the prevalence of specific ASNs in the dataset.

# Chapter 2: AI unsupervised approach

## 2.1 Full dataset Clustering

Subsequently, all features were scaled, and the dataset was clustered using both K-Means and HDBSCAN algorithms. Cluster quality was assessed using the Silhouette score and visual inspection of dimensionality-reduced data using PCA and UMAP. The clustering results were deemed unsatisfactory, primarily due to increased computational demands associated with the high dimensionality of the feature space. Furthermore, K-Means clustering was only effective in identifying two to four distinct clusters, as illustrated in Figure 8.A group of colored dots

AI-generated content may be incorrect.

Fig 8: K-Means cluster visualization

HDBSCAN is a hierarchical clustering algorithm, it uses density-based connectivity to identify clusters of varying densities, providing a more flexible approach than traditional clustering methods. The ideal number of clusters for the data is determined by the algorithm, unlike k-means which is manually selected.

This method is a great help to make very efficient and realistic clusters based on the data. In my case the algorithm selected 61 clusters signifying that the data was too complex and it was not clustering based on the type of attack but other patterns. This makes sense as I had around 300 features, a vast majority being text embeddings. The silhouette score for these clusters was very low if not negative on some occasions which makes sense looking at the cluster visualization in figure 9 below.

A cluster of colored paint

AI-generated content may be incorrect.

Fig 9: HDBScan pca reduction visualization

## 2.2 Grouped dataset

As seen above, the dataset was of very high dimensionality and included a lot of samples. To reduce the size of the data and thus the computational time I decided to group the attacks based on the source and destination IP, where each package sent needs to have at maximum a 2-minute interval with the previous package. This reduced the number of samples, grouping them by the same attack timing. This reduced the time for clusters generation as well as removing some of the noisy samples between attacks. This also allowed the addition of features group-based features as: time between attack, byte/s, duration of the attack, average packet length, number of packets during the attack as the main ones.

### 2.2.1 Feature selection and test

After testing I realized that the embeddings and onehot encoded protocols were simply increasing the complexity, computational time and blurred the clusters together, so I removed them. This modification led to high silhouette up to 0.9 averaging 0.75 based on the initialization (figure 10).

A graph of a plot

AI-generated content may be incorrect.

Fig 10: silhouette score of hdbscans

The dimensionality reduction to graph the samples led to more defined groups of data showcasing better cluster potential, figure 11 below.

A graph with colored lines

AI-generated content may be incorrect.

Fig 11: PCA reduced HDBScan cluster

# Chapter 3: AI semi supervised approach

## 3.1 Model trained on the cic and predicting on omniport

The initial plan was to train a model directly on the CIC-IDS 2017 dataset and apply it to the OmniPort dataset. To enable this, I adapted the OmniPort feature set to match CIC-IDS, as described in Section 2.2, by grouping forward packets and extracting equivalent features.

However, this direct transfer approach performed poorly, predictions were heavily skewed toward the majority classes, indicating that the domain shift between CIC and OmniPort was too large for simple cross-dataset inference. This limitation motivated the exploration of semi-supervised methods, starting with pseudo-labelling on CIC-IDS 2017, described in Section 3.2.

## 3.2 Pseudo-Labelling configuration on CIC IDS 2017

Pseudo-labelling is a method of assigning labels to samples without ground-truth guarantees. The main challenge lies in generating labels with the highest possible confidence.  
To address this, I reviewed several research papers on pseudo-labelling and cyber attack detection. Two papers formed the primary basis for the approach: [CRUPL (arXiv:2503.00358)](https://arxiv.org/pdf/2503.00358) and [Cyber Attack Detection (arXiv:2001.06309v1)](https://arxiv.org/pdf/2001.06309v1).

The pipeline I built requires a small subset of labelled data. For testing and parameter tuning, I used a subset of the CIC-IDS 2017 dataset. Out of roughly 600,000 available samples, I selected between 10,000 and 20,000 as the labelled subset, treating the remainder as unlabelled. This section details the development and refinement of the pipeline.

### 3.2.1 Initial Approach

The first attempt involved training a model on the labelled subset, predicting labels for the unlabelled pool, and selecting high-confidence predictions based on the maximum Softmax scores. These pseudo-labelled samples were then added to the training set, and the model was retrained refining the new model with the newly added predicted samples from the first set.

However, results were poor, likely due to noisy pseudo-labels and the absence of structured selection. Over time, the model became biased towards majority classes and ignored minority ones completely.

After reviewing CRUPL, I adopted several of its key components, along with additional modifications of my own, starting with consistency regularization.

### 3.2.2 Consistency Regularization

Consistency regularization is a semi-supervised learning technique that constrains a model to produce similar predictions for different perturbations of the same input. The key idea is that the predicted label of a sample should not change significantly if the input is slightly altered in ways that do not change its true class (e.g., adding noise, applying augmentations, or modifying feature order).

In my implementation, Gaussian noise was added to the unlabelled samples, and the model was trained to minimize the difference between predictions on the original (clean) inputs and their perturbed (noisy) versions. This encourages stability in the decision boundary and improves robustness to small variations in the input space, making it a valuable component in the pseudo-labelling process.

### 3.2.3 Curriculum Learning

Instead of pseudo-labelling all unlabelled data at once, I implemented curriculum learning: running multiple training cycles with a gradually decreasing confidence threshold. This allows the model to start with the easiest (most confident) pseudo-labels and progressively include harder ones.

To reduce the influence of potentially incorrect pseudo-labels, I applied down-weighting:

labeled\_mask = np.concatenate([  
 np.ones(len(x\_labeled)),  
 np.full(len(x\_train\_curr) - len(x\_labeled), 0.3)  
])

This ensured that pseudo-labelled samples contributed less to the total loss, particularly in the early stages.

### 3.2.4 High-Confidence Sample Selection

Initially, all predictions above a fixed confidence threshold were selected. However, this often included overconfident but incorrect predictions, especially in imbalanced class settings. After a few steps, the model became heavily biased toward majority classes.

A graph with green and blue lines

AI-generated content may be incorrect.

Fig 12: curriculum learning progress showing spike down after 4 steps

To address this, I added entropy and confidencemargin filters, selecting only samples with low entropy and a large margin between the top Softmax scores. Additionally, I capped the number of samples added per step to avoid overwhelming the model.

Even with these measures, pseudo-labelled data still became dominated by majority classes after several curriculum steps, making minority class learning difficult.

A graph with numbers and a chart with blue squares

AI-generated content may be incorrect.

Fig 13 confusion matrix of pseudo labels vs true labels

The accuracy plateaued around **0.4–0.6**, with **F1 scores around 0.3**.

### 3.2.5 Fixing Class Imbalance in Pseudo-Labels

I introduced per-step class balancing at each step, the selected pseudo-labels had to include samples from all classes. This curbed the feedback loop where majority classes were reinforced each round.

Fig 14, 15, 16 show the results after applying this fix.

A black and white rectangular object with white text

AI-generated content may be incorrect.

Fig 14: label distribution

A graph with green and blue lines

AI-generated content may be incorrect.

Fig 15: curriculum learning process tracked

A graph with numbers and labels

AI-generated content may be incorrect.

Fig 16: confusion matrix at the end of the process

Accuracy improved to **0.80** with **F1 score at 0.60**, though some very small classes were still underrepresented.

### 3.2.6 Reducing Number of Classes

To simplify the prediction task, I reduced the number of target classes to the six most represented, removing the least frequent ones. While this reduced granularity between attack types, it significantly improved model performance and confidence in pseudo-labels.

**Results:**

* Accuracy: **0.82**
* F1 Score: **0.77**

A graph showing the growth of a curriculum

AI-generated content may be incorrect.

Fig 17 curriculum learning process with only 6 classes

However, even in this simpler setup, the model showed a tendency to favour the majority class (e.g., class 5).

A graph with numbers and labels

AI-generated content may be incorrect.

Fig 18: confusion matrix labelling on 6 classes

### KL Divergence for Pseudo-Labelled Samples

Later, I refined the loss function to treat labelled and pseudo-labelled data differently:

* Labelled samples: Sparse categorical cross-entropy
* Pseudo-labelled samples: KL divergence against soft pseudo-label distributions

KL divergence allowed the model to better capture uncertainty and avoid overfitting to potentially incorrect pseudo-labels. This soft alignment improved stability in later curriculum stages.

While this helped, overfitting still occurred after the first few steps. To counter this, I reduced the model’s dimensionality, which stabilized early-stage learning (Fig 19, Fig 20). Performance only declined when the number of selected samples fell below 6,000, meaning most selected samples were high-confidence and high-accuracy at that point.

This highlighted that too many pseudo-labelled samples being added early could clutter the learning process and introduce compounding errors.

A graph showing the growth of a curriculum

AI-generated content may be incorrect.

Fig 19:

A screenshot of a computer screen

AI-generated content may be incorrect.

Fig 20:

This led me to think more in the curriculum learning formatting where in this case too many samples would be pseudo-labelled early on which cluttered the learning process early leading to some inaccuracies and as soon as some inaccuracies are introduced the errors build up. For that I limited the number of samples that can be selected greatly:

To address this, I modified the selection strategy:

Initially:

* Limit selection to 1/8 of the unlabelled dataset per step
* Enforce per-class distribution constraint

Revised:

* Over the first 6 steps, gradually increase the limit until reaching 1/8 of the dataset (Fig 21)

A graph showing the growth of a curvy line

AI-generated content may be incorrect.

Fig 21:

This adjustment kept early-step accuracy high, with drops occurring only when the number of selected samples fell below 10,000, corresponding to the hardest samples to learn. This produced similar overall accuracy but with greater robustness.

Testing with different initial labelled sets showed slightly higher average F1 scores. Notably, the first 9 steps no longer exhibited a sharp drop after step 5–6, with accuracy remaining in the 90s instead of falling into the 80s.

A screenshot of a graph

AI-generated content may be incorrect.

Fig 22:

To further increase robustness, a stopping criterion could be added to end curriculum learning when the number of selected samples drops below a certain threshold, avoiding premature stopping in the early capped steps. This would ensure that only high-confidence samples are retained, with harder-to-predict samples ignored entirely.

## 3.3 Omniport Pseudo-Labelling Implementation

With the CIC-IDS 2017 pipeline working, I moved on to the main problem: labelling and identifying the type of attacks in the OmniPort dataset. Initially, there was a possibility of obtaining a small set of labelled OmniPort data, but when that plan fell through, I had to rely entirely on external sources. This included the CIC-IDS 2017 dataset, a labelled dataset from Kaggle, and clustering-based intuition and payload analysis to evaluate potential labels.

Although the OmniPort dataset lacked direct labels, the curriculum learning approach developed in Section 3.2 remained applicable for generating high-quality pseudo-labels. For evaluation purposes (discussed later), I first split the CIC dataset into training and test sets, then trained a curriculum learning model incorporating all the improvements described above.

Using this trained model, I applied the pseudo-labelling process to the OmniPort data. I then analysed the resulting labels in the context of the entire OmniPort dataset and compared their distribution to that of the CIC dataset. From the CIC-trained model’s predictions, I selected only the high-confidence samples, as described in earlier sections. The results of this process are shown below.

A chart of a diagram

AI-generated content may be incorrect.

Fig 23

A graph of blue and orange dots

AI-generated content may be incorrect.

Fig 24

A screenshot of a graph

AI-generated content may be incorrect.

Fig 25

As discussed earlier, the CIC and OmniPort datasets differ in distribution. However, they also complement each other in certain regions of feature space. In Figure 24, the CIC samples in the bottom right are complemented by the OmniPort samples, which show slight overlap but also extend the distribution toward the left.

When observing the clusters formed from the pseudo-labelled predictions, I found both strong similarities and meaningful differences between the datasets. This outcome appears far more promising than any unsupervised labelling attempts using HDBSCAN or k-means.

To check the stability of these results and ensure they were not a one-off coincidence, I repeated the process multiple times with different hyperparameters. Across these runs, the resulting labels were highly consistent, which supports the conclusion that the pseudo-labels have relatively high confidence.

### 3.3.1 Evaluation method

The visual evaluation provided a good starting point, but I also explored additional methods to assess pseudo-label quality:

1. **Reverse evaluation:** A model trained on the labelled data was used to generate pseudo-labels for the unlabelled set. These pseudo-labels were then used to train a new model, which was evaluated on the labelled data. This helped measure how well the pseudo-labels preserved the true class structure.
2. **Self-consistency check:** A model trained solely on the pseudo-labelled data was applied to the unlabelled set again, to see how closely its predictions aligned with the original pseudo-labels and whether the outputs were coherent.

# Chapter 4: Payload analysis

The payload is a key component of a cyber attack and analysing it in detail can help determine the type of attack. One quick approach is to use open-source cybersecurity models that take payload and contextual information to classify attacks.

To investigate this, I explored several open-source models, including Cisco’s Foundation AI model ([Foundation-Sec-8B](https://huggingface.co/fdtn-ai/Foundation-Sec-8B)). Unfortunately, most of the available payloads in the dataset were corrupted or unreadable, which significantly reduced the number of usable samples. Out of approximately 470,000 valid payloads (non-empty and non-duplicate), only about 170,000 were suitable for analysis.

The majority of usable payloads consisted of very simple SSH or HTTP requests with minimal or empty content — for example:

“GET /hello HTTP/1.1”.

Despite this limitation, I tested Cisco’s Foundation AI model to see what kind of results it would produce. The model sometimes gave inconsistent outputs for the same query, which might have been due to differences in initialization.

One of the more complex and interpretable payloads was:

Payload: GET /libs/js/iframe.js HTTP/1.0   
User-Agent: xfa1   
Accept: \*/\*   
Host: 216.238.86.69

The model returned different responses depending on the run:

**Response 1:**

*Scan*  
The request for "iframe.js" in the /libs/js/ directory, combined with a non-standard User-Agent ("xfa1"), suggests a potential probe for known vulnerabilities or common file locations. This is characteristic of a scanning attempt.

**Response 2:**

*Attack Class: WebAttack*  
Explanation: The payload requests a JavaScript file (iframe.js) using HTTP/1.0 with a suspicious user-agent ("xfa1"). The filename and path (/libs/js/iframe.js) are commonly associated with malicious iframes used in drive-by download or redirection attacks, which are typical of web-based exploitation attempts.

This example also illustrates that the same payload can yield different model outputs, particularly when the payload is very simple. Single requests often lack sufficient context, so accurately determining the type of cyber attack would likely require combining information from multiple packets or flows. This highlights the importance of considering temporal or session-level aggregation of payloads, as well as incorporating additional metadata, to improve the reliability of automated attack classification.

# Chapter 5: Results

## 5.1 Structure of the Results Section

This chapter presents the outcomes of the methods described in the previous sections, focusing on both quantitative performance metrics and qualitative observations. It is divided into three main parts:

1. **Quantitative evaluation**: summarising the performance of unsupervised clustering, curriculum-based pseudo-labelling, and cross-dataset evaluation.
2. **Qualitative analysis**: including stability of pseudo-labels, visual inspection of clusters, and payload-based plausibility checks.
3. **Ablation and sensitivity**: briefly discussing how specific design choices (grouping, class reduction, loss function) affected performance.

The evaluation reflects the central challenge of this project: the absence of true labels in the Omniport dataset. As a result, some metrics are derived from labelled datasets (CIC IDS 2017) and others are indirect or qualitative.

## 5.2 Results Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Experiments | Dataset | Metrics | Value | Notes |
| Ungrouped cluster | Omniport | Silhouette | 0.3 | Poor separation; high compute |
| Grouped cluster | Omniport | Silhouette | 0.77 | Grouping improved scores but clusters ≠ attack types |
| Curriculum pseudo-labelling | CIC IDS (train/test) | Accuracy / F1 | 0.80 / 0.60 | With class balancing and KL loss |
| Curriculum pseudo-labelling (6 classes) | CIC IDS 2017 | Accuracy / F1 | 0.82 / 0.77 | Reduced classes → higher accuracy |
| Pseudo-label → train on Omniport → eval on CIC | Omniport → CICsubset | Accuracy / F1 | TBD | Measure cross-dataset alignment |

## 5.3 Step-by-Step Evaluation

**Step 1: Unsupervised Baselines**  
As the initial baseline, the semi-supervised pipeline was compared against purely unsupervised clustering using the same feature space. K-Means was tested with varying numbers of clusters, reporting the best silhouette score achieved. HDBSCAN was evaluated using several min\_cluster\_size parameters. While clustering occasionally produced visually separable groups, these did not guarantee attack type-based clusters and did not align well with known dataset, highlighting the limitations of unsupervised methods.

**Step 2: Baseline on CIC IDS 2017**  
Next, baseline performance was established on a known, labelled dataset. The curriculum learning model was trained on the labelled portion of CIC IDS 2017 and evaluated on a held-out test split. Performance was measured using Accuracy and Macro F1 to capture both overall correctness and balance across classes. This baseline serves as a reference for later evaluations.

**Step 3: Pseudo-labelling Omniport**  
The trained model was applied to the unlabelled Omniport dataset, generating pseudo-labels using the confidence- and entropy-based selection rules outlined in Chapter 3. Early curriculum steps admitted only high-confidence predictions, while later steps gradually included more difficult samples. Pseudo-label distributions were examined for class skew, and the procedure was repeated under multiple seeds and hyperparameters to measure label stability (e.g., % agreement and Adjusted Rand Index).

**Step 4: Cross-dataset Evaluation**  
Finally, a model trained solely on the pseudo-labelled Omniport dataset was tested on a labelled subset of CIC IDS 2017. This indirect evaluation assessed whether the pseudo-labels captured meaningful attack categories. High-quality pseudo-labels would allow the model to perform competitively on the CIC subset, despite domain differences.

## 5.4 Narrative Summary

Direct clustering of the Omniport dataset, even when grouped by attack session, struggled to recover meaningful attack categories. Although grouped HDBSCAN achieved silhouette scores averaging around 0.8. When comparing the clusters with labelled data and looking at the loadings and features importance they were often defined by superficial features rather than underlying attack type.

Curriculum pseudo-labelling on CIC IDS 2017 provided a substantial improvement in label quality over naive pseudo-labelling. With all classes present, the approach achieved an Accuracy of 0.80 and a Macro F1 of 0.60, aided by class balancing and KL divergence loss for pseudo-labelled samples. Reducing the number of classes to six improved these metrics further (Accuracy 0.82, Macro F1 0.77) but at the cost of fine-grained attack differentiation.

Transferring the CIC-trained model to Omniport yielded pseudo-labels that were consistent across runs, with stability scores (TBD) indicating high reproducibility. Cross-dataset evaluation showed that a model trained on these pseudo-labels retained a significant portion of its predictive power when tested on CIC IDS 2017, suggesting that the pseudo-labels encode meaningful information.

Payload analysis highlighted a further limitation: despite 470,000 non-empty payloads, only ~170,000 were usable after filtering, with most containing minimal content (e.g., simple HTTP GET requests). This constrained the potential of payload-only classification and explains why metadata-driven approaches were more effective in this setting.

In conclusion, the curriculum-based pseudo-labelling pipeline demonstrated clear advantages over unsupervised clustering in the context of unlabelled cyberattack datasets. While domain shift and incomplete payload data remain challenges, the results support the viability of this approach for generating high-confidence labels in the absence of ground truth.