PHASE 1: PROBLEM IDENTIFICATION

EXFILTRATION USING DISCORD

Authors:

Ana Vidal 118408

Simão Andrade 118345

Problem

Discord is a popular **web-based** communication platform used in non-corporate and corporate networks (e.g. Software Development companies)

Being a **legit application** that run in very **usual network ports**, it can be used to **exfiltrate** data

Real Life Cases

Examples of data exfiltration using Discord:

- The Hacker News - NS Stealer Uses Discord Bots to Exfiltrate Data (2024)
- Intel471 - How Discord is Abused for Cybercrime (2024)

Why is it difficult to solve?

Discord uses HTTP/HTTPS to send data, therefore the data uses encryption during the communication

- This difficults the analysis for systems that perform DPI (Deep Packet Inspection) or based only on rules/policies

Although Discord made some updates regarding security (link), malicious users still take advantage of tools that allow development of plugins

Discovery

- It has a built-in function that enables automated messages sent to a text channel in the server (Webhooks)
- Allows the upload of a variety file types (e.g. PNG, PDF, MP4)
- The maximum file upload is 10MB

Data Filtering

- IP Network: 162.159.0.0/16 (nslookup.io)
- Uses WebSockets over **TCP** for real-time communication
  - Destination Port(s): TCP/80 and TCP/443
  - Source Port(s): UDP/50000-65535
- Voice/Video communication and background syncronization is done using **QUIC**

Data Agreggation

To perform the analysis, the following data will be extracted:

- **Group and Private Conversations** – the conversation type is obtained at the packet level (uploads/downloads)
- **Daily and Weekly message flow with various formats of files** – analyzing the timestamps of interactions (uploads/downloads)

Data Collection In Testing Context

Tools of network analysis:

- Wireshark:
- TCPDump

Proxy tools for traffic capturing:

- Burp Suite

Data Collection In Real Context

Tools to obtain data from devices, server, aplications, etc.:

- Syslog
- Agents

Qualitative Data - Packet Level

- IP Source
- IP Destination
- Used Protocol
- Packet Length
- Timestamp (in seconds)

Qualitative Data – Flow Level

- IP Source
- IP Destination
- Size of Exchanged Data
- Data Flow Start/End Timestamp (in seconds)
- IP Protocol Number

Data Sampling – Sampling Interval

- In order to convert our qualitative data into quantitive data, we chosed to use observation windows of **0.1 seconds** and **1 second**
- This allows a balance between the level of detail needed to capture relevant events and the volume of data generated

Data Sampling – Packet Level (1/2)

- **Number of UDP packets (Download/Upload)**
- **Number of TCP packets (Download/Upload)**
- **Number of UDP bytes sent (Download/Upload)**
- **Number of TCP bytes sent (Download/Upload)**

Data Sampling – Flow Level (2/2)

- **Mean and Standard Deviation of idle times:** Unusual gaps or consistency between flows.
- **Number of flows:** Indicating irregular usage patterns.
- **Size of exchanged data (Mean/Variance):** Changes in data size can point to unexpected or secretive data transfers.
  - Up/Down
- **Durations of Flows**

Data Production

It will be done using **tree types** of bots:

- **Easy to Detect:**
  - **Size:** 10MB
  - **Frequency:** Periodically (40s)
- **Intermediate to Detect:**
  - **Size:** 1-10MB
  - **Frequency:** Same variance as a normal behavior
- **Hard (almost impossible) to Detect:** Through embedded images, using Discord CDN

Malicious Behavior

It will be done by performing **normal usage** of the application, made by:

- **Humans:** sending messages and files as usual
- **Bots:** made by plugins added to the server

Normal Behavior

QUESTIONS?

PHASE 2: PROJECT IMPLEMENTATION

EXFILTRATION USING DISCORD

Authors:

Ana Vidal 118408

Simão Andrade 118345

Data Processing

For the processing of the samples, we used:

- **Multi-Observation Window**
- **Window Width Size of 300 samples (5 minutes for 1 sec. samples)**
- **Window Slide of 30 samples (30 seconds for 1 sec. samples)**

Feature Extraction

We have in total, **22 features**:

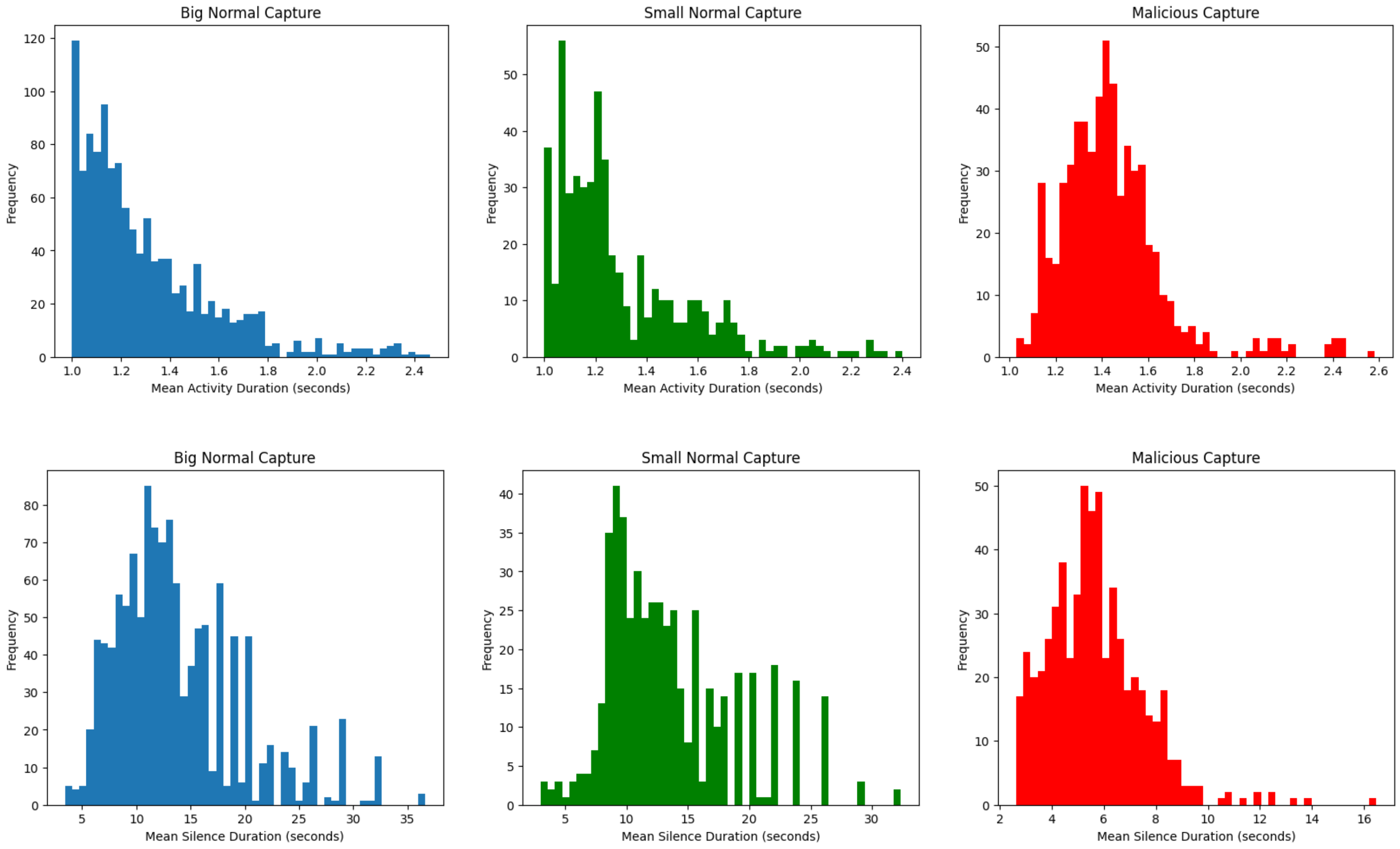
- **Mean and Variance of silence times**
- **Mean, Variance and 95th and 98th percentile of activity times**
- **Mean, standard deviation, 60th and 90th percentile of upload and download bytes for TCP and UDP (separately)**
- **Mean and standard deviation of total bytes**
- **Mean and standard deviation of number of packets**

Data Analysis (1/7)

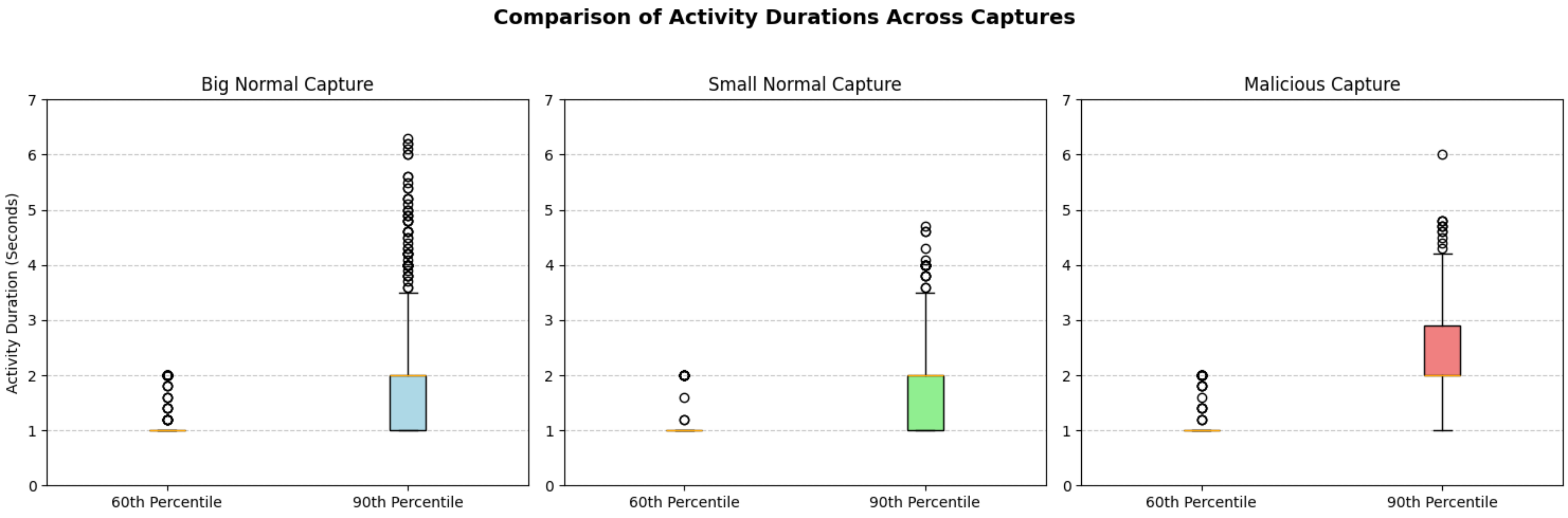
Prior to the behavior model, we analysed the following datasets:

- **Normal use of the application during a day**
- **Normal use during a shorter period**
- **50/50 of Normal and Malicious use during a day**

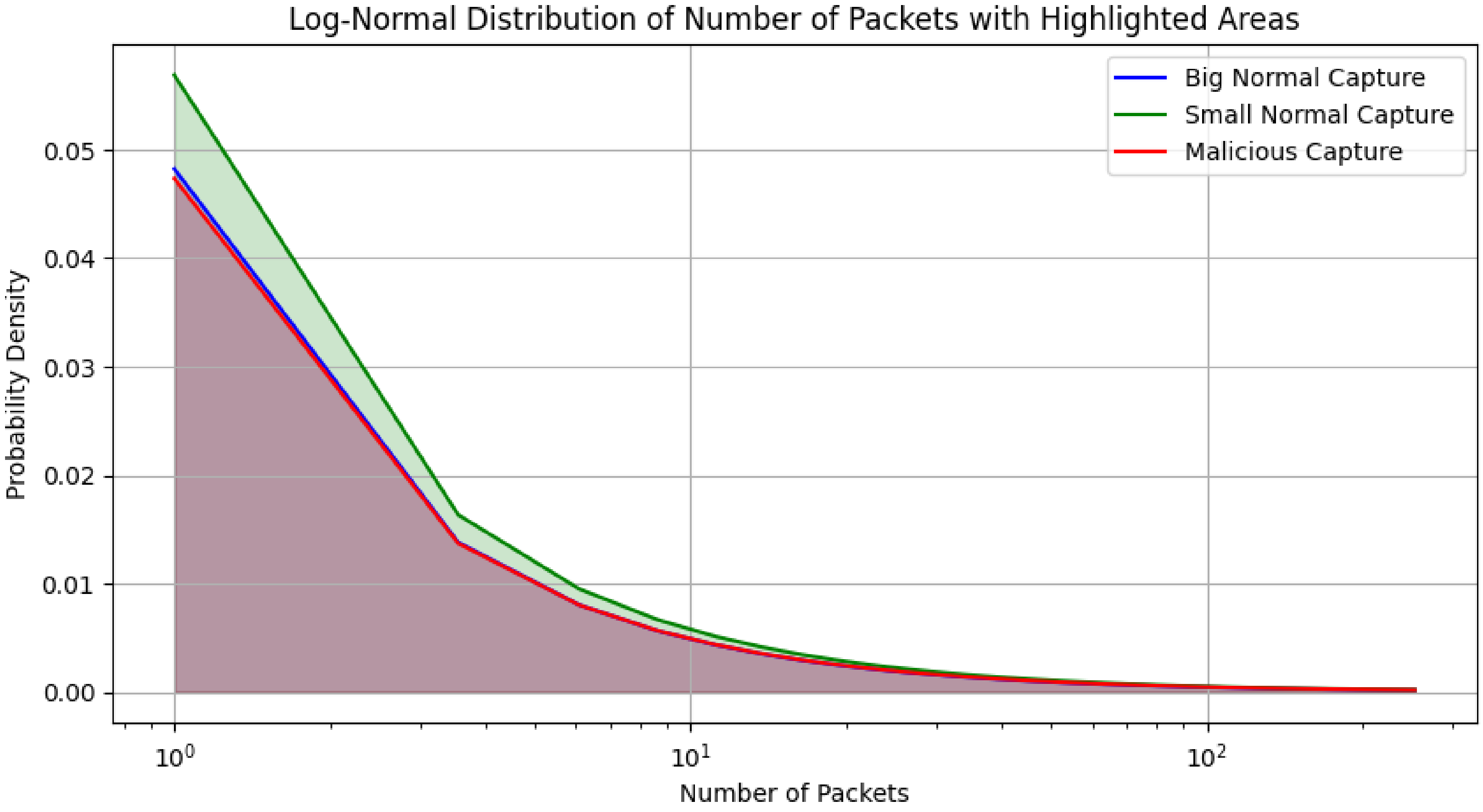
Data Analysis (3/7)



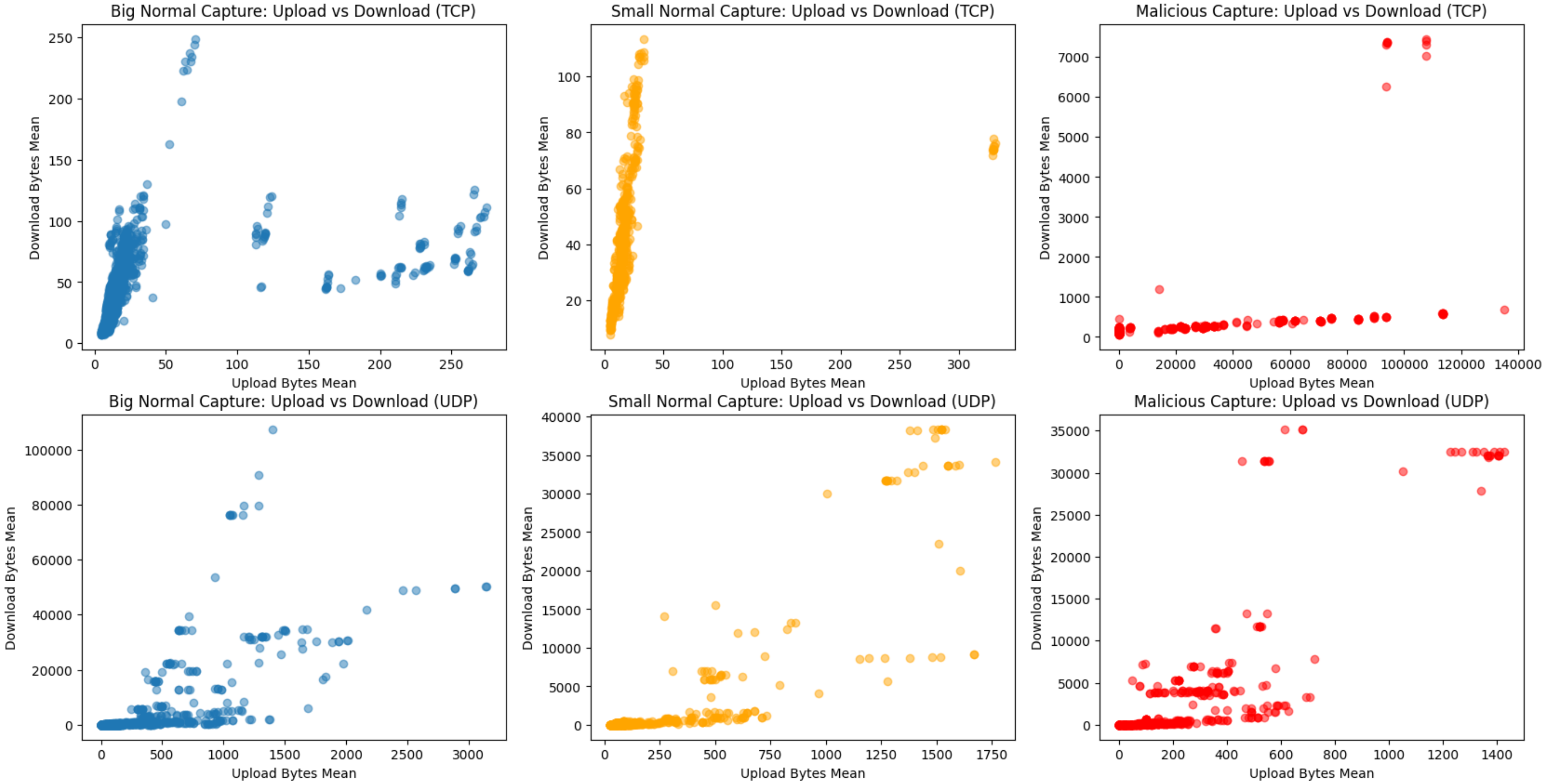
Data Analysis (4/7)



Data Analysis (5/7)



Data Analysis (6/7)



Big Normal Capture: Upload vs Download (TCP)

Small Normal Capture: Upload vs Download (TCP)

Malicious Capture: Upload vs Download (TCP)

Big Normal Capture: Upload vs Download (UDP)

Small Normal Capture: Upload vs Download (UDP)

Malicious Capture: Upload vs Download (UDP)

Data Analysis (7/7)

The use of ML is **crucial**, because:

- **The Discord App is too complex to understand.**
- **The solution can’t be performed by defining a threeshold**
- **Both the benign and malicious activity look similar**

Behavior Models

The models used were:

- **Autoencoder**
- **Isolation Forest**
- **OneClass SVM**

The normalization performed was:

- **MinMax**

Behavior Algorithms

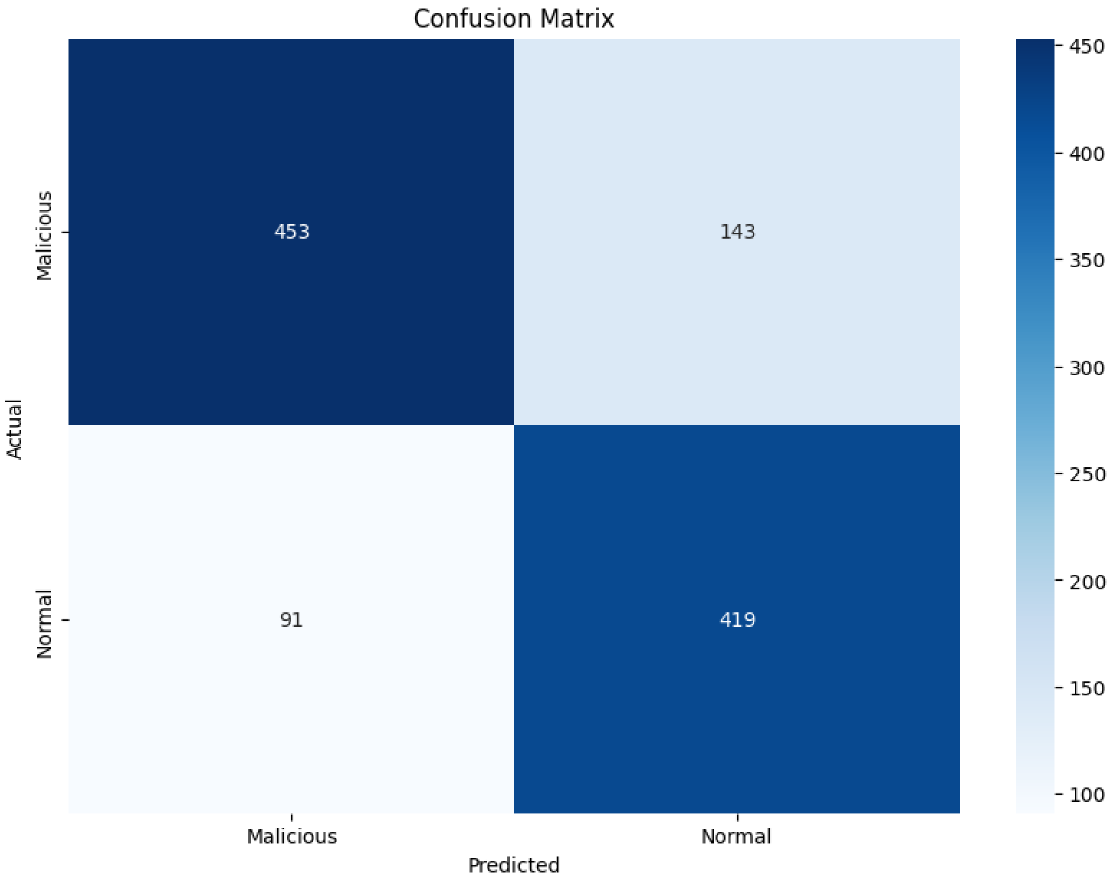
The Bot performed exfiltration by:

- **Using a prior capture of the user**
- **Made a histogram of the times they are most active**
- **Given the probabilities of the intervals, the data was sent**

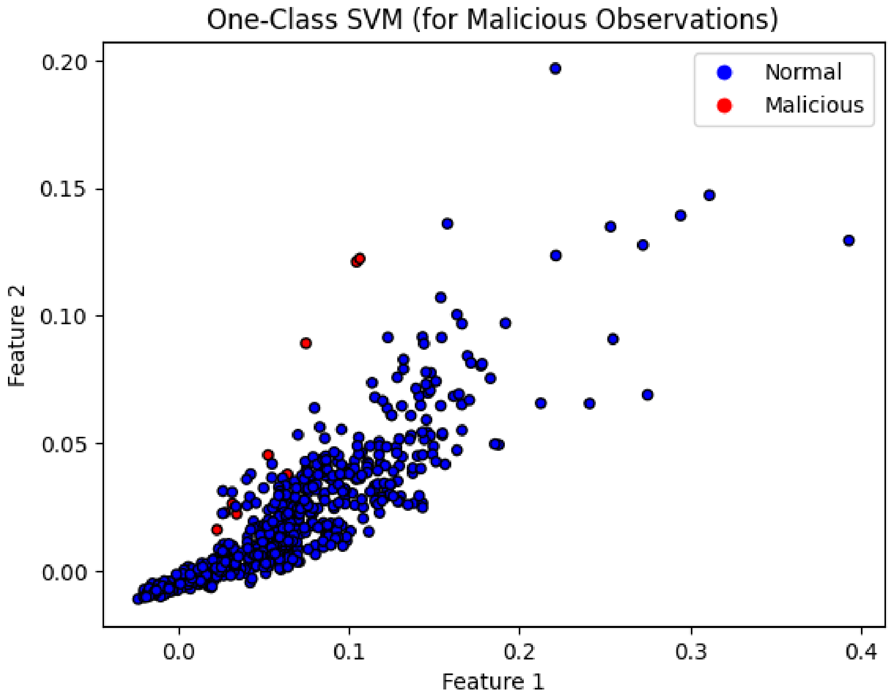
Obs.: The simpler version of the bot (periodically) was not analysed here.

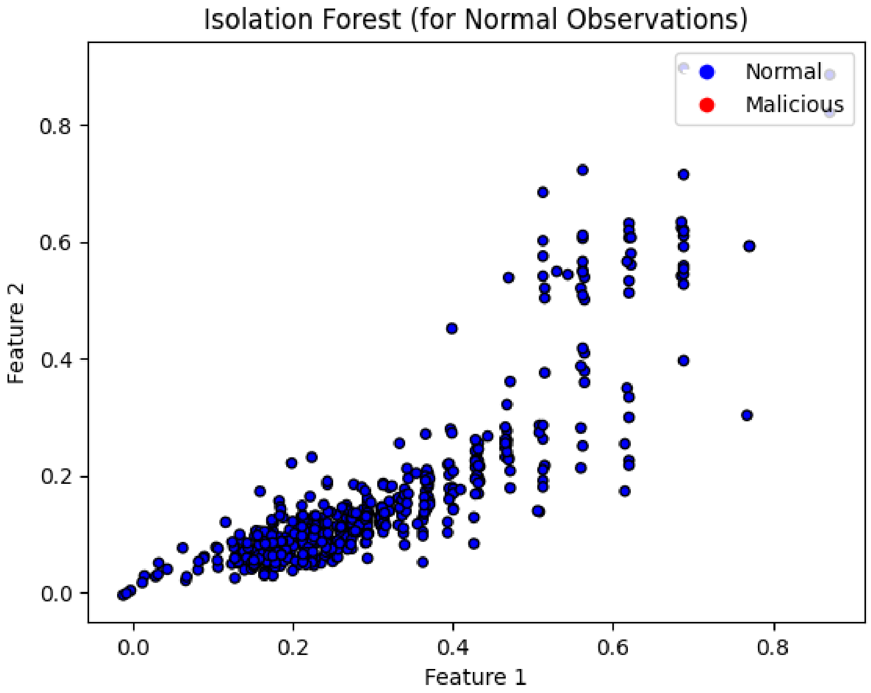
Results – One-Class SVM

- Precision: 0.83
- Recall: 0.76
- F1 Score: 0.79
- Accuracy: 0.79



Results – Isolation Forest (w/o PCA)

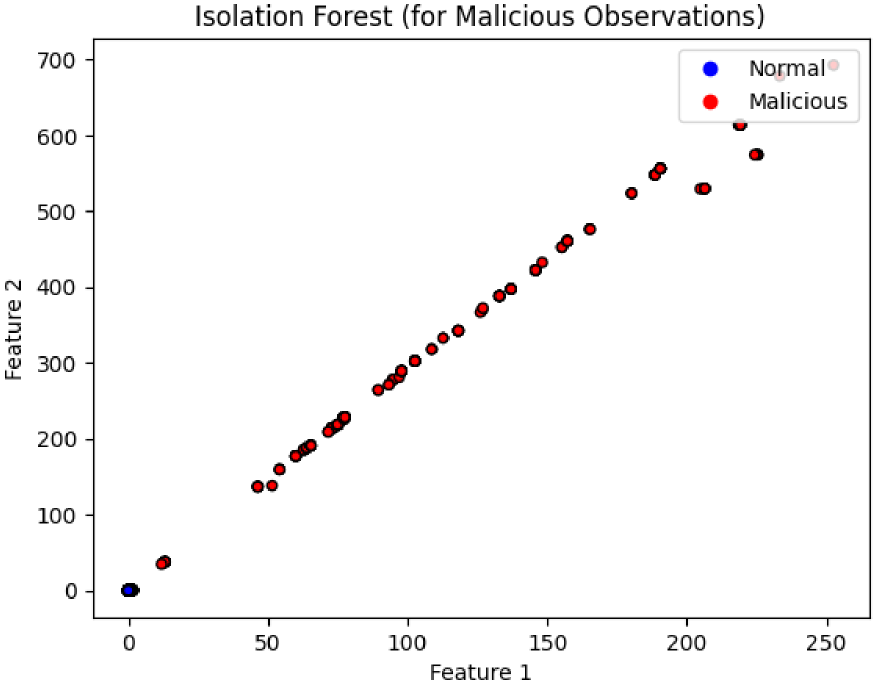


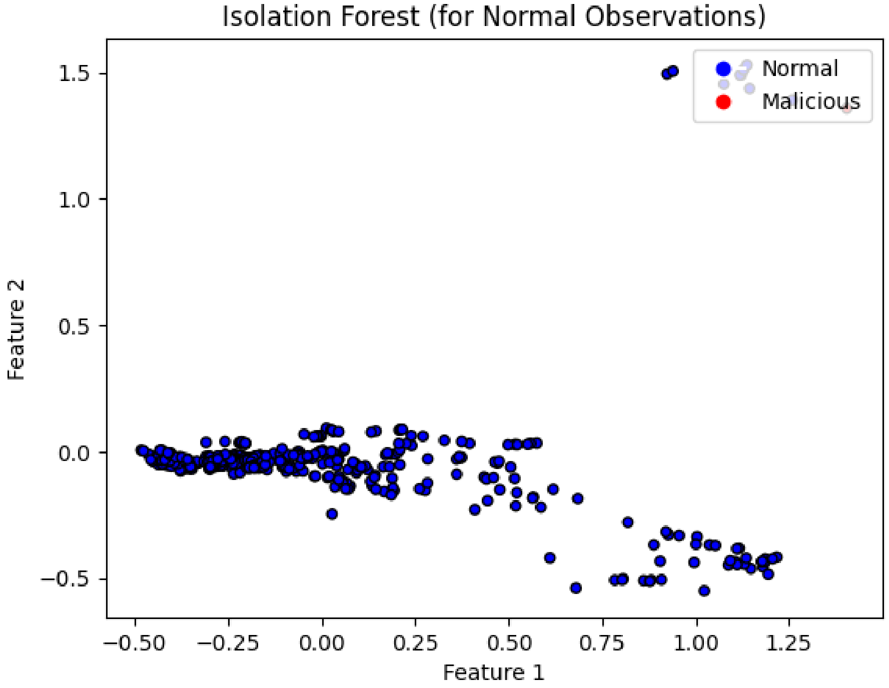


Normal observations: 587
Malicious observations: 9
Percentage of malicious observations: 1.51%

Normal observations: 510
Malicious observations: 0
Percentage of malicious observations: 0.00%

Results – Isolation Forest (w/ PCA)



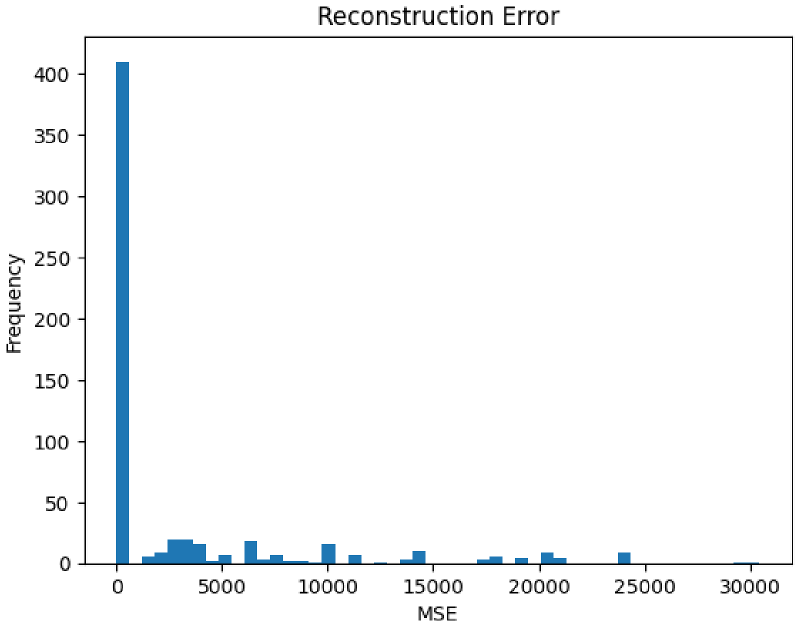


Normal observations: 400
Malicious observations: 196
Percentage of malicious observations: 32.89%

Normal observations: 509
Malicious observations: 1
Percentage of malicious observations: 0.20%

Results – Autoencoder

- Normal observations: 566
- Malicious observations: 30
- Percentage of malicious observations: 5.03%



Problem complexity

The biggest problem:

- Classification of behavior

Proposed Solution

- Proposing methodologies to improve the:
  - **Attack** – Make changes to the bot in order for him to differentiate sended messages from files,
    - This could be done by defining a threeshold of the number of packs sended, representing an attachment.
  - **Defense** – Make datasets more robust, in order
    - Dimension Reduction (prioritize the better features)
    - More observation Windows
    - Define labels of benign and malicious observation windowsum dataset mais rubosto, maior, com mais informação de modo a treinar melhor o modelo

QUESTIONS?