Detecting Hidden Command & Control Traffic

Aprendizagem Aplicada a Segurança, 2022/2023
Universidade de Aveiro

Authors: **Camila Fonseca - 97880**
Rodrigo Lima - 98475

1. Introduction

In recent years, the threat of remote access trojans (RATs) has become increasingly prevalent in the cybersecurity landscape.

A Remote Access Trojan (RAT) is a type of malware that allows an attacker to gain unauthorized access to and control over a target system. RATs typically establish a covert communication channel with a Command and Control (C2) server, which is used to issue commands to the infected system and receive information from it. RATs are often used to steal sensitive information, install additional malware, or conduct other malicious activities on the target system.

RATs are stealthy and can be difficult to detect, as they often use encrypted communications and can be designed to blend in with normal network traffic. This makes it challenging for organizations to identify when they have been infected with a RAT and to protect against data exfiltration, the unauthorized transfer of sensitive data from the target system to the attacker.

The use of RATs and C2s has significant implications for the security of organizations and individuals. The potential for data exfiltration and theft of sensitive information can have a severe impact on an organization's reputation and financial stability. Furthermore, the difficulty in detecting RATs and C2s can make it challenging for organizations to respond effectively to security incidents, exacerbating the impact of the attack.

The detection of RATs and C2s is further complicated by the use of sophisticated techniques by attackers, such as the use of encrypted communications, dynamic IP addresses, and the ability to hide behind legitimate network traffic. This requires organizations to continuously monitor their networks and update their security measures to stay ahead of evolving threats. Failure to do so can result in data breaches, theft of sensitive information, and other serious consequences. In short, RATs and C2s represent a significant threat to the security of organizations and individuals, and it is crucial for organizations to be proactive in their efforts to detect and prevent these threats.

2. The Problem

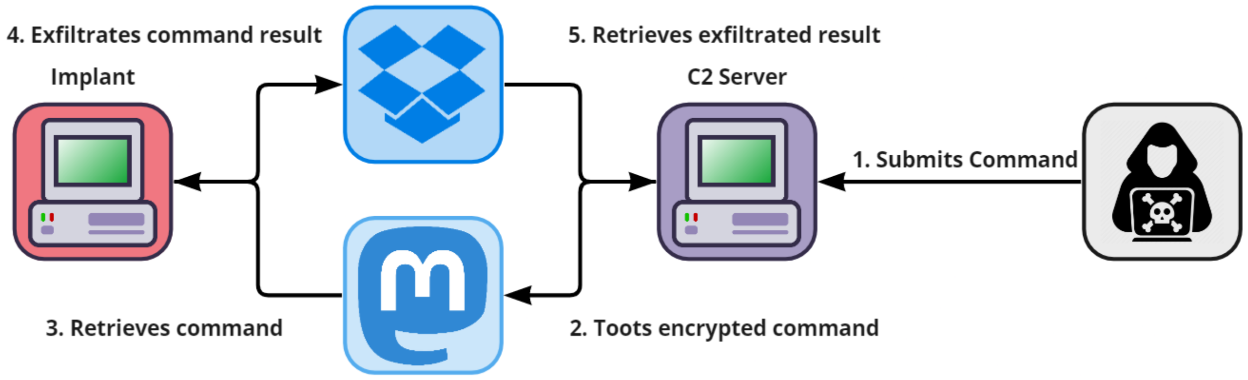
The tool BlueBreaker was developed to assist in the research of detecting Remote Access Trojans (RATs) in network systems.

BlueBreaker is a cyber attack tool that enables an attacker to execute commands on a target machine without alerting security measures. This tool was developed with the aim of mimicking user interaction by utilizing two main channels of communication. The two channels used are Mastodon, a free and open-source social media platform, and Dropbox, a cloud-based file storage and sharing service.

The BlueBreaker tool consists of two components: the implant and the command and control server. Both components are written in Python and work together to achieve the attacker's goal. The command and control server is responsible for publishing encoded commands to be executed by the implant. The implant retrieves these commands and executes them, uploading the output to Dropbox.

The use of Mastodon and Dropbox helps the attacker blend in with legitimate traffic, reducing the fingerprint left on the network. Mastodon is a social media platform that is similar to Twitter and is used by the target user, making the traffic generated by the implant blend in more easily. The implant interacts with Mastodon only through the backend API, reducing the fingerprint left on the network. The Dropbox service is widely used in corporations and enables the attacker to blend in with legitimate traffic.

The command and control server retrieves the output from Dropbox after a set period of time has passed and presents it to the attacker. This tool operates by publishing commands executed output and by querying the API for new commands at random intervals, mimicking a user's usage pattern.


4. Exfiltrates command result
Implant
5. Retrieves exfiltrated result
C2 Server
1. Submits Command
3. Retrieves command
2. Toots encrypted command

In conclusion, BlueBreaker is a powerful tool that enables attackers to remotely execute commands on a target machine, similar to a Remote Access Trojan (RAT). The tool's design, utilizing Mastodon and Dropbox as communication channels, makes it possible to blend in with the normal traffic of the target user, reducing the footprint left on the network. The only traces left on the network are encrypted HTTPS traffic or DNS requests, which are indistinguishable from normal usage. This makes it difficult for security measures to detect the presence of BlueBreaker and enables the attacker to remain undetected

One possible solution to detect BlueBreaker in a network is through the use of machine learning techniques applied to the network's packet capture (PCAP) data. PCAPs contain a wealth of information about network traffic, including the source and destination IP addresses, port numbers, and payload data. Machine learning algorithms can analyze this data to identify unusual or suspicious behavior, such as the use of encrypted traffic or the presence of specific patterns in the payload data that are indicative of a RAT like BlueBreaker.

By training machine learning models on PCAP data generated by the target, organizations can detect the presence of BlueBreaker or other RATs in their network. These models can then be used to continuously monitor the network for unusual activity, enabling organizations to quickly detect and respond to potential security threats. In this way, machine learning can provide a valuable tool for organizations looking to improve their cybersecurity posture and protect against advanced attacks like BlueBreaker.

3. Data Collection

In order to collect data for later analysis, wireshark was used as our packet capture tool. This was done in a single host, employing a virtual machine with a lightweight Linux installation. This setup was selected to minimize potential background traffic and limit the introduction of unwanted noise in the captured packets.

For this project, three different types of data were generated, with differing goals and environments:

- **Training Data**

  Clean data, where the BlueBreaker Implant is not running. The data collected spanned a large time window, but more data would have been desirable, since even "normal" browsing can have vastly different patterns, making anomaly detection harder.

- **Validation Data**

  Labeled data, where the BlueBreaker implant is running in the background and its activity is clearly labeled as anomalous. This was accomplished by running BlueBreaker inside a Docker Container and capturing its traffic in isolation, and later merging these data with a set of clean data, **different** from the Training data.

- **Testing Data**

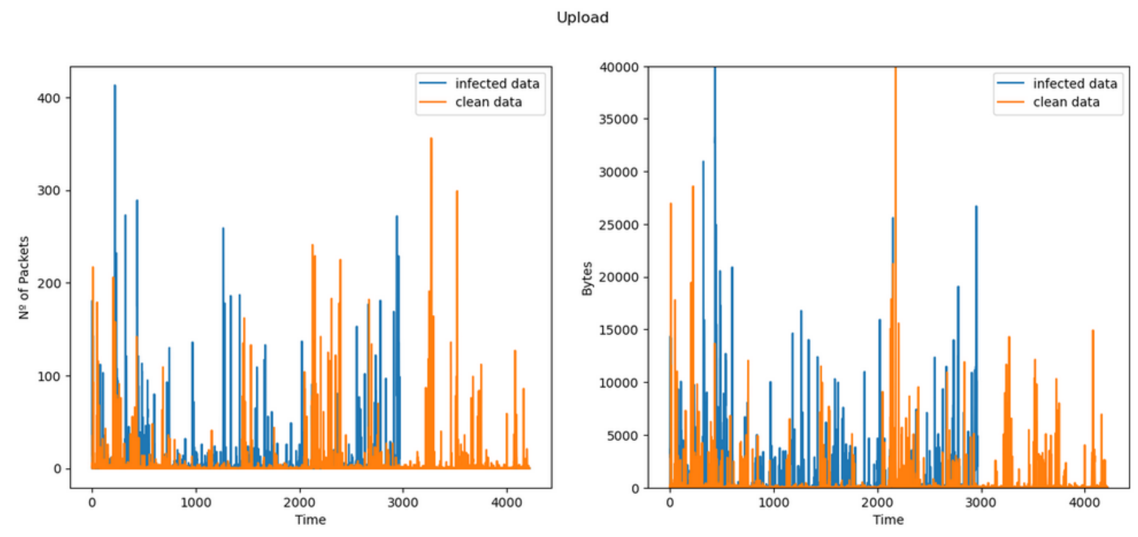
  Data collected in the same manner as Training Data, albeit with the BlueBreaker implant active and running in the background, listening, executing commands, and uploading their output.

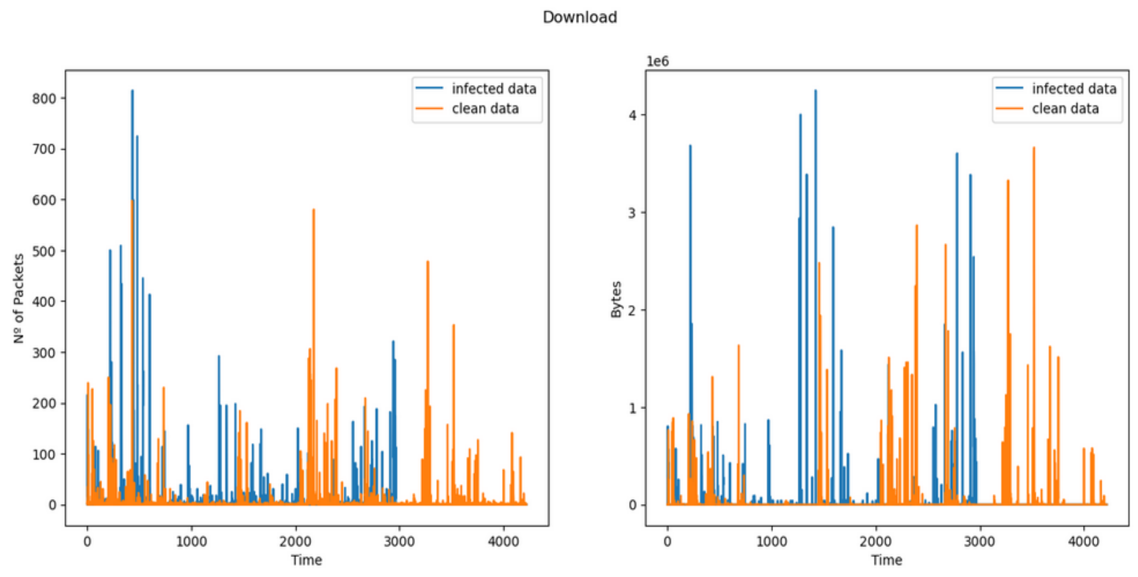
4. Data Processing

In order to obtain usable data from the packet captures, several scripts were utilized. These scripts were made available by the course professors.

With the goal of obtaining the needed features, four metrics were extracted from the packet capture files - Packet count and size (bytes) for upload and download respectively.

These metrics already allow us to visualize the data we’re working with:





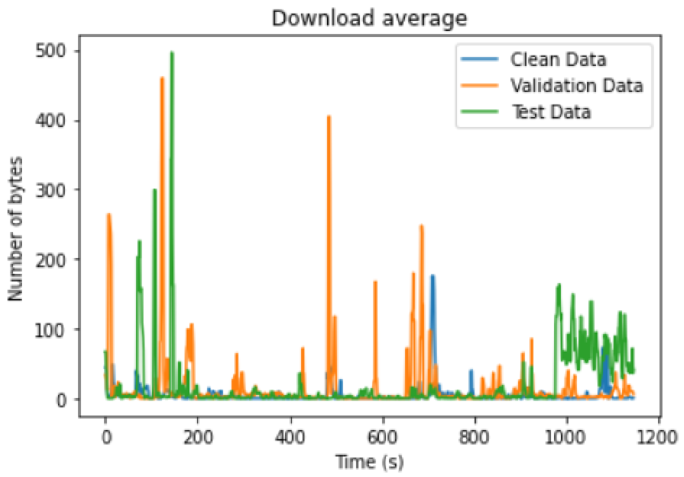
Then, these metrics are separated into chunks via the sliding observation window method. From these, the features are then extracted.

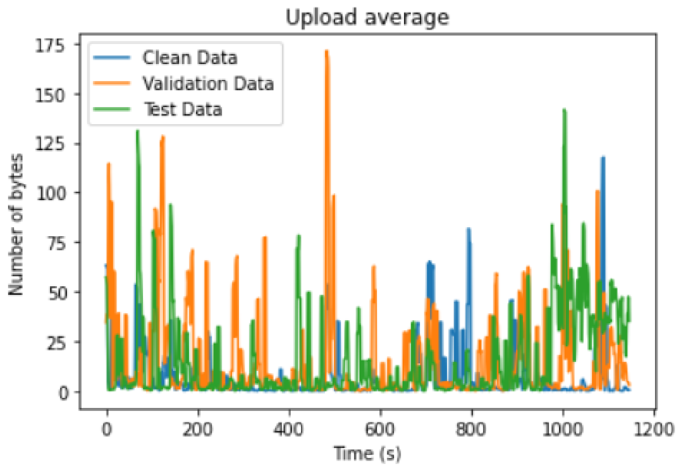
For each metric, the following features are obtained:

- Average
- Median
- Standard Deviation
- Skewness
- 75th, 90th, 95th and 98th percentiles

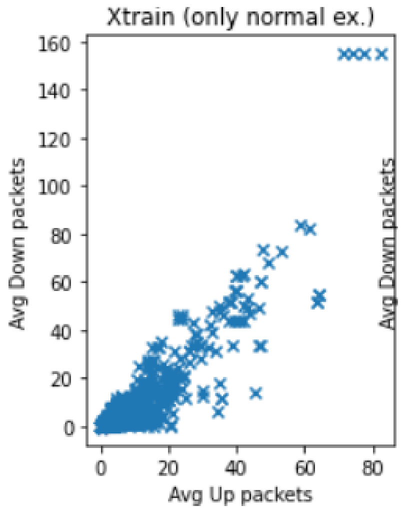
Some are as presented below:

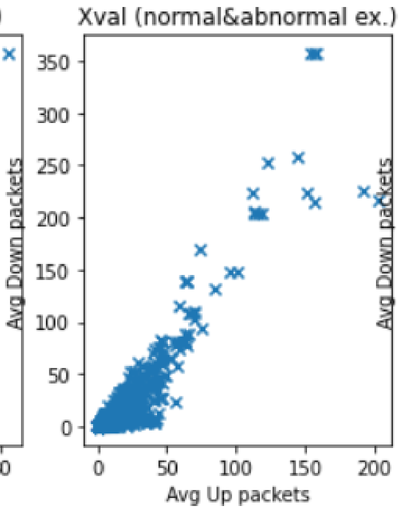
Feature: Upload/Download Avg.

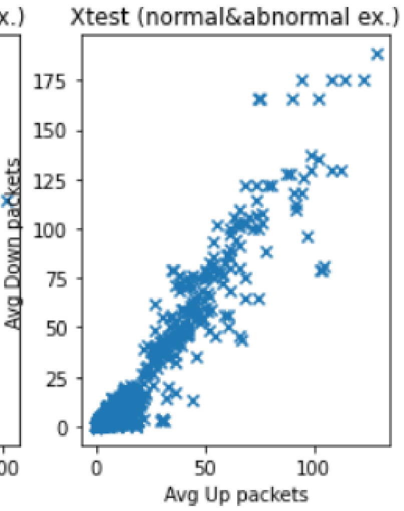




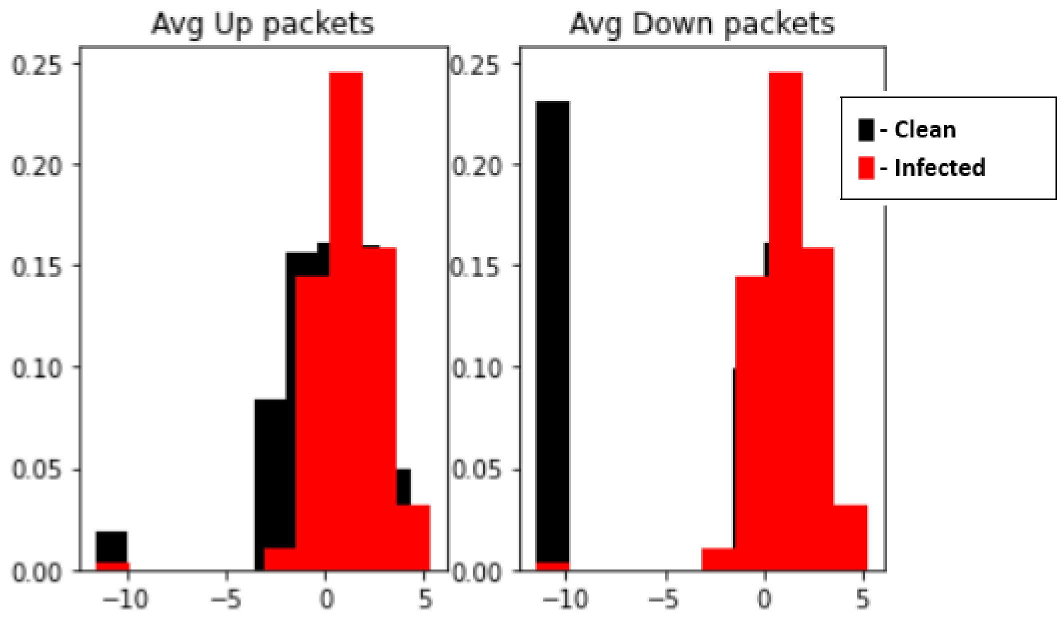
2D Feature Visualization







Feature Engineering - Comparison



5. Machine Learning & Results

The datasets being worked with consist of many non-anomalous examples with only a very small amount of anomalous samples. Of these anomalies, there's no way to tell if the future anomalies would be like the ones already seen before. As such, anomaly detection was deemed the most optimal approach to solve our problem, using a Multivariate Gaussian Distribution model.

The parameters (mean and variance) for the model are calculated for the training data (Without anomalies).

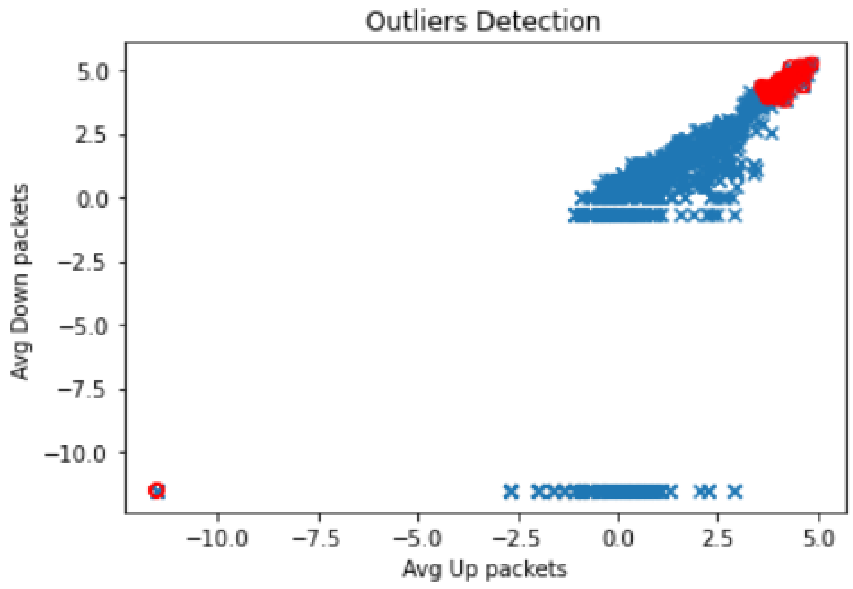
```
def estimateGaussian(X):
    """
         This function estimates the parameters of a Gaussian
distribution using the data in X
    """
    # number of examples in X
    m = X.shape[0]

    # compute mean
    mu = np.mean(X, axis=0)

    # compute variance =sigma^2
    sigma2 = 1/m * np.sum((X - mu)**2,axis=0)

    return mu,sigma2
```

mu: [-0.02332439 -3.69696273]
sigma2: [7.07376256 38.82417484]

Then, a multivariate Gaussian distribution is applied to the Validation data (With labeled anomalies), in order to compute the probability for each example of being anomalous. The best threshold value that separates the anomalous from the normal examples is then calculated for the validation data, based on the highest F1 score (Measuring the accuracy of the binary classification).

Best epsilon found using cross-validation: 0.0016846418016633349
Best F1 on Cross Validation Set: 0.15126050420168066

To finalize, and using the previously chosen parameters, the gaussian distribution is applied to Test data (Unlabeled anomalies), and using the threshold selected, the anomalies are identified.



Accuracy: 0.72 %