Universidade de Aveiro

Segurança em Redes de Comunicações

Report of Project 2

André Clérigo (98485), Pedro Rocha (98256)

Departamento de Eletrónica, Telecomunicações e Informática

June 11th, 2023

Contents

List of Figures

Chapter 1

Introduction

As digital technology becomes a fundamental part in today's business, we are also seeing an increase in potential cybersecurity risks. As businesses continue to grow their digital footprint, it is of the most importance to have strong and reliable systems in place. With this in mind, the project objective is to identify unusual behaviors and potentially compromised devices using Security Information and Event Management (SIEM) systems.

This project primary objective is to define and implement SIEM rules that can effectively identify unusual network activities that may indicate potential threats. The project utilizes a "corporate" network's historical data of traffic flows, knowing that one of the datasets provides a typical network behaviour and the other dataset contains anomalous network behaviors.

Based on the analysis of these datasets the outcome of the project will be a set of SIEM rules, conceived from our analysis, that can identify potential threats either by flagging them as an alarm or outright blocking actions based on the severity of the threat.

In conclusion, the project is an exercise in network security, using data analysis to understand network behaviors, define alert rules, and test their effectiveness. This endeavor is not only an exercise in cybersecurity but a crucial step in safeguarding the corporate network's integrity.

Chapter 2

Methodology

The project operates in a methodical manner, commencing with the analysis of the dataset "dataX.parquet", which contains a full day's data of typical network behavior. This data, already certified free of illicit behavior, forms the basis for our understanding of normal network operations.

Subsequently, we delve into the "testX.parquet" dataset. This dataset, mirroring the structure of the earlier dataset but potentially containing anomalous behaviors, enables us to contrast normal and abnormal network activities and identify unique patterns that signal potential threats.

The data analysis will use Python, leveraging the pandas library for data analysis. Alongside, we will use databases for geo-localization based on IPv4 addresses, we also used Jupyter Notebooks for a straightforward analysis of the data which greatly simplified the creation of the SIEM rules.

2.1 Data Analysis

In the data analysis part of the project, we started by collecting simple information from the datasets using pandas from both the normal and test datasets like the used ports and protocols, assessing that the ones used were: UDP over port 53, UDP over port 443 and TCP over port 443 that represent DNS, QUIC and HTTPS, respectively. It is possible to check which is the average of use of the protocols from the Figure 2.1.

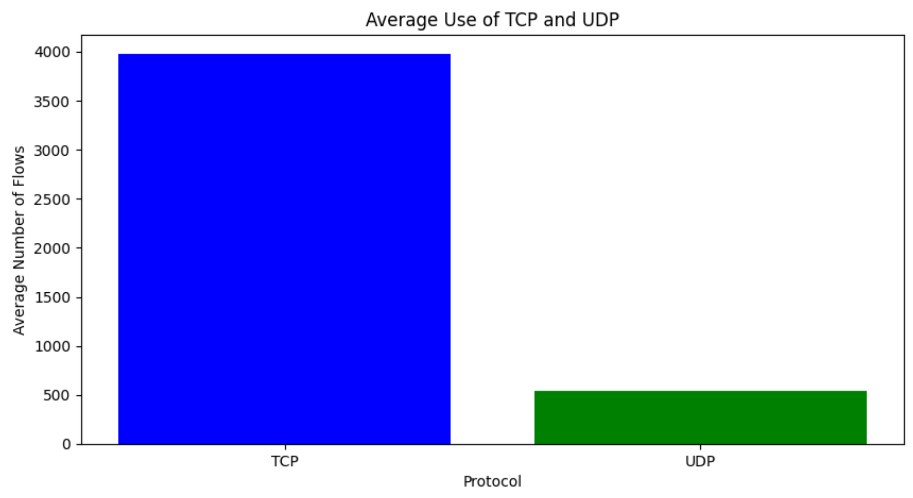


Figure 2.1: Protocols average use.

We also analyzed the amount of flows per source IP, uploaded bytes and downloaded bytes per flow and total, both for internal communications (private IP to private IP) and for external communications (from private IP to public IP). Also collecting metrics about the mean value, standard deviation and variance. Additionally, there were also some country statistics collected, having in mind all of the metrics said before.

Chapter 3

Non-Anomalous Behavior Analysis

Discuss the typical behavior of the network devices based on the analyzed data. Having this normal dataset given as a starting point for what's the typical behaviour of the network when there isn't any malicious activity, we collected some information from it in order to later compare it to the test dataset that has anomalous communication patterns.

3.1 Internal communications

When exploring the internal communications (between private IP addresses), we found several normal behaviours like the list of IP addresses that usually utilize the network and their communications inside of it. About the protocols and ports used, it was observed that there are no TCP connections on port 53. This port is generally used for DNS traffic. This finding suggests that DNS traffic on this network is using UDP instead of TCP, which is typical for DNS queries and responses. Besides that, a significant number of UDP connections were observed to be using port 443, which is typically associated with HTTPS traffic using TCP. The use of UDP on this port suggests that QUIC protocol might be in use. QUIC is often used for improving web page load speed and reducing latency in web traffic. Still on this part, we analyzed the number of UDP and TCP flows for each source IP, were we discovered which IPs were the ones belonging to the Servers of the network. Has it's possible to check on Figure 3.1, the IPs with more flows are the potential server addresses, so that leaves us with the server IPs: 192.168.101.234, 192.168.101.239, 192.168.101.224 and 192.168.101.228. The other IP (192.168.101.154) has a more regular amount of traffic.

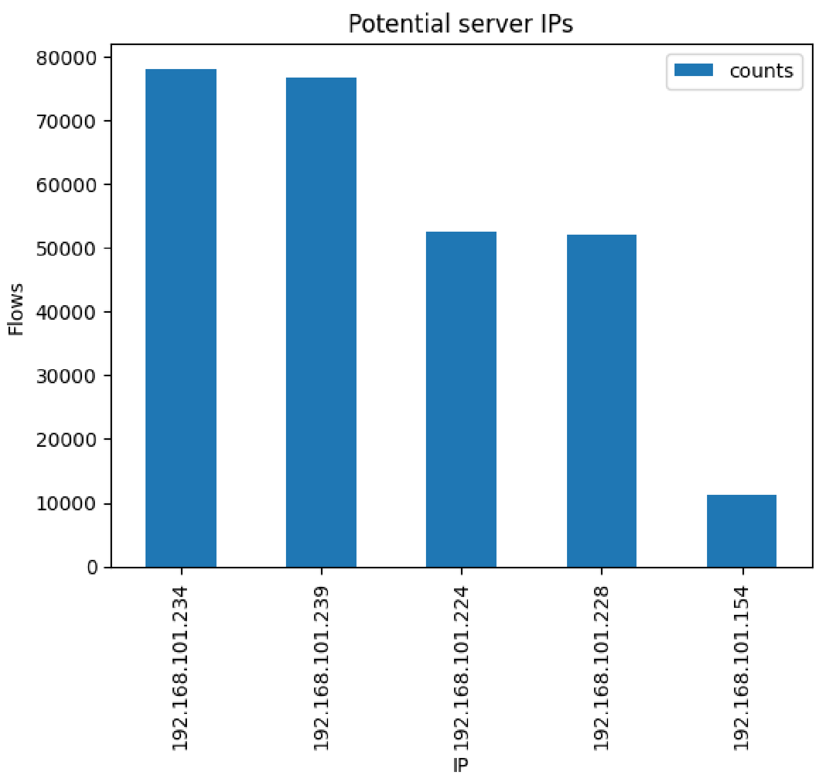


Figure 3.1: Potential Server IPs.

In the Figure above we can see the top 5 private IPs with the most flows. It is clear that the 5th IP doesn't have as much flows as the other ones, but this alone does not mean that this IP is not a server. We can assess that this IP is not a server because its flow count (11162) is in line with the mean values for flow count (4518).

3.2 External Communications

On this section, we adopted the same approach as before, like analyzing which protocols, ports, IP addresses and flows density there was. However, we also were able to collect which countries and organizations were being contacted from the internal network and how much it was happening, so that we would have a baseline of what were the internal communications to the outside network. Getting relevant information like the one displayed on Figure 3.2. We also obtained values like flows per country, flows per organization, etc.

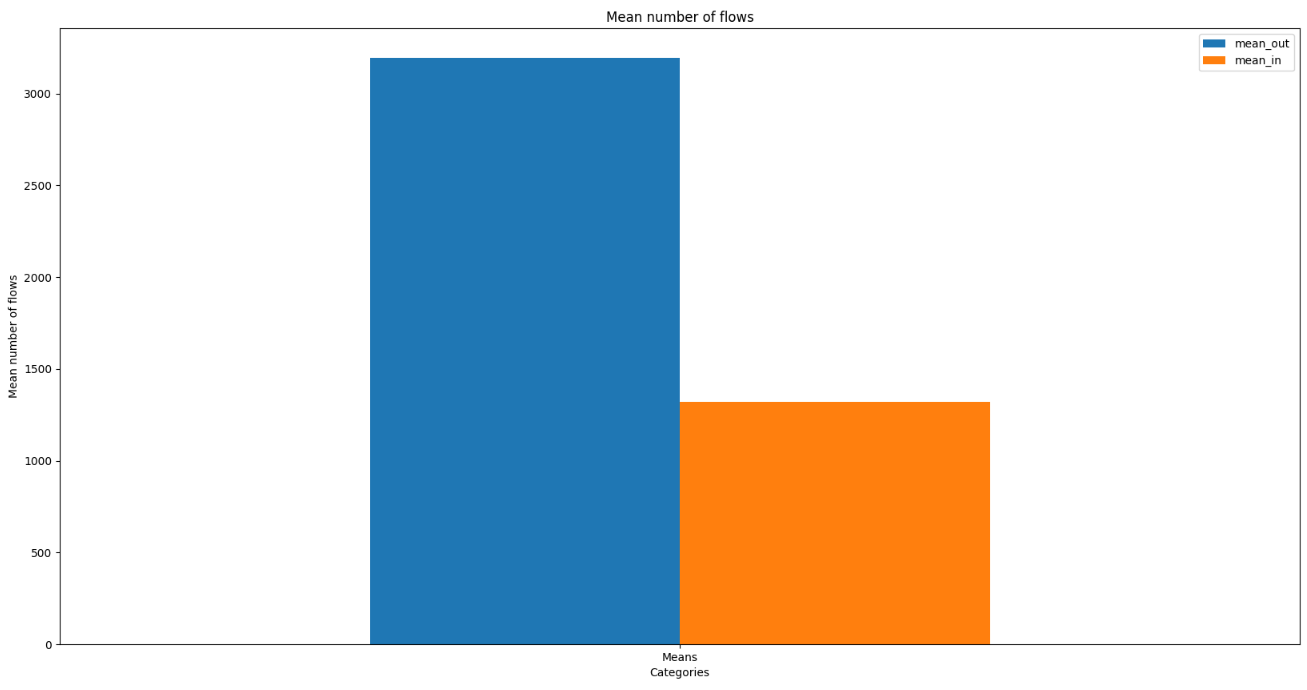


Figure 3.2: Average of flows to the outside and inside networks.

For reference we analysed the top 5 countries with the most flows, uploaded bytes and downloaded bytes. The countries with the most flows, uploaded bytes and downloaded bytes obtained were United States of America, Portugal, Netherlands, Namibia and Great Britain respectively.

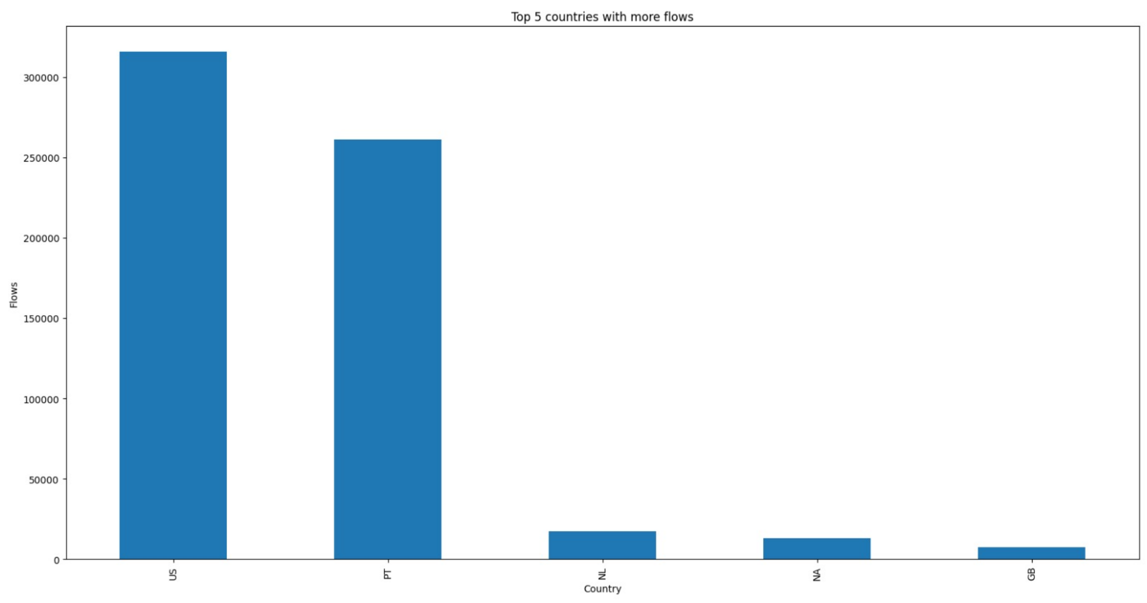


Figure 3.3: Top 5 countries with the most flows.

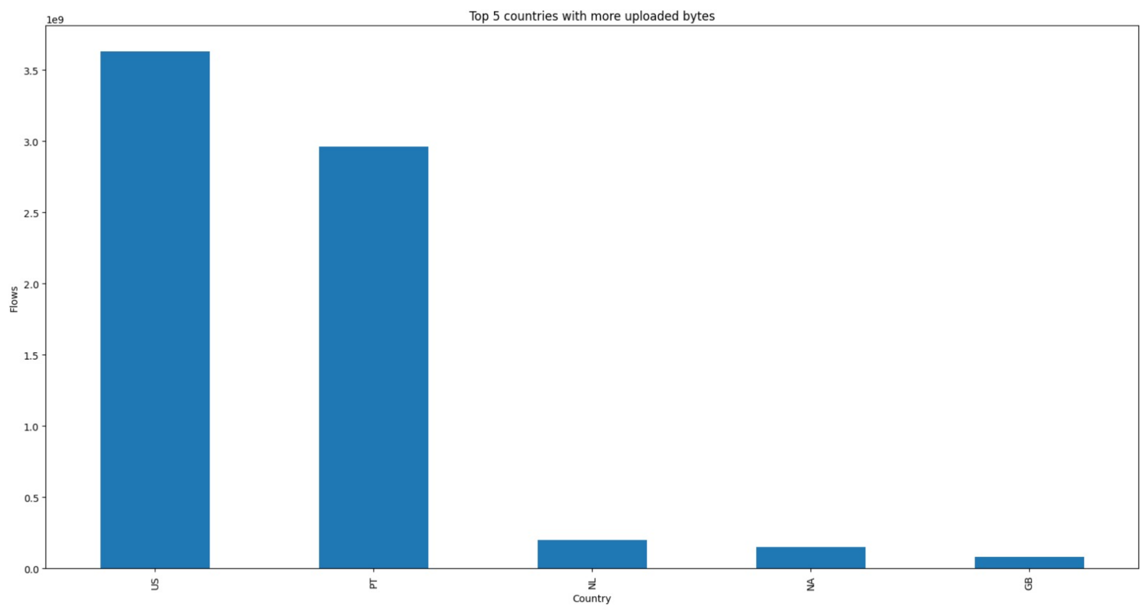


Figure 3.4: Top 5 countries with the most uploaded bytes.

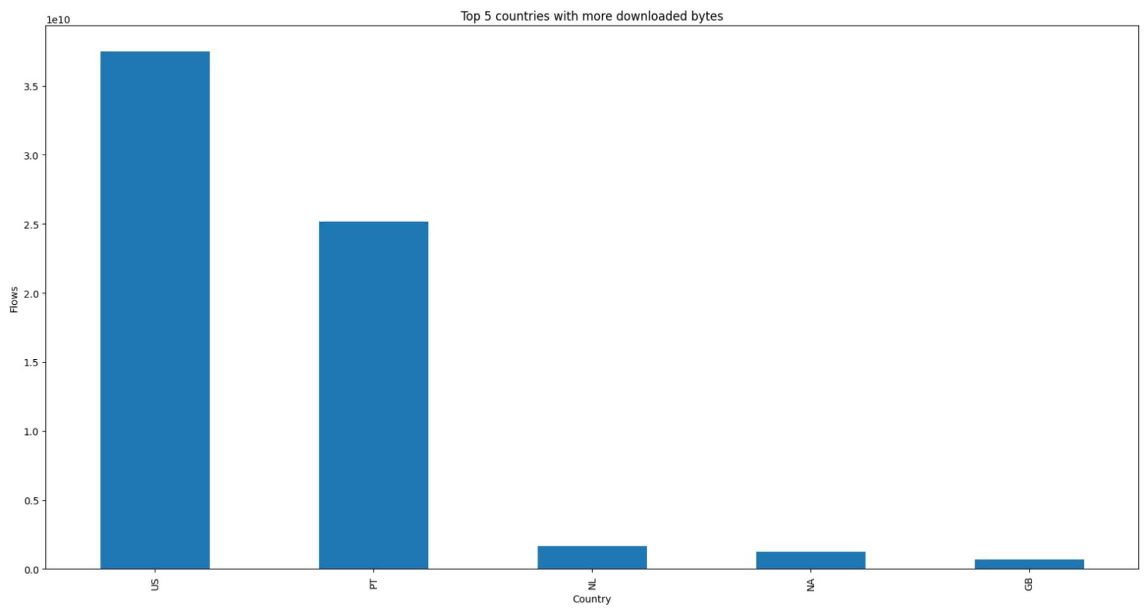


Figure 3.5: Top 5 countries with the most downloaded bytes.

Chapter 4

Anomalous Behavior Detection

4.1 Botnet activity

A botnet is a set of compromised devices that are under the control of a central command. These compromised devices, often referred to as "bots", are typically infected with malicious software without the knowledge their owners.

Based on that, we take the IP addresses that are engaging in new internal communications and consider them as suspects. Next, we analyze whether their communications are normal or malicious by observing factors such as abnormally high volume of data, the presence of numerous flows, etc.

The IPs that we detected as suspect were 192.168.101.11, 192.168.101.110, 192.168.101.14 and 192.168.101.34.

To determine whether these IP addresses are part of a botnet, further analysis should be conducted.

As an example we can refer to Figure 4.1 which shows a new IP making connections to some of the servers, however, these connections have a small time window for a human to do these many requests. This type of analysis is something that we couldn't perform with automation and had to analyse "by hand". The other types of abnormalities are easy to catch with a rule.

In our investigation, we identified several IP addresses engaging in suspicious internal communications. While some abnormalities can be easily detected with automated rules, identifying unusual connection patterns may require manual analysis. Isolation provides time for that analysis, allowing security teams to gather more information.

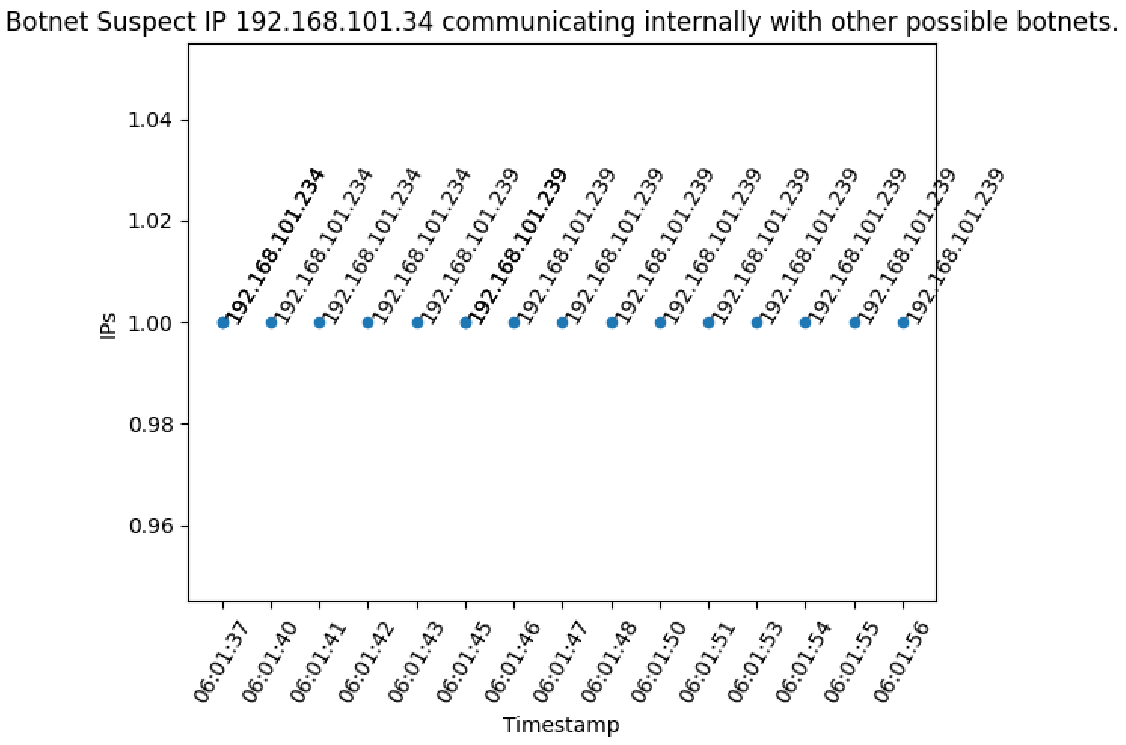


Figure 4.1: Botnet Behaviour.

4.2 C&C attacks

In the section dedicated to the Command and Control (C&C) assessment of our project, it is crucial to highlight the salient events that took place during the monitoring phase. Through meticulous data analysis, it was discerned that the servers were under a notable attack. This was inferred from an unusual spike in the number of flows targeting the server IPs (224 and 228) as it possible to see in Figure 4.2. Typically, the network traffic, or 'flows', remain within a certain expected range. However, during the evaluation period, a considerable surge in flows directed towards the server IPs was observed, which was indicative of a possible C&C activity.

Furthermore, our in-depth investigation revealed that there was an abrupt increase in communication between certain IP addresses and the servers. This manifested as a substantial rise in flows from these IPs (41, 42, 60) to the servers as it is possible to see in Figure 4.3. While some fluctuations in traffic are expected during normal operations, the sheer volume and abruptness of this increase in the number of flows raised concerns. It is important to mention that certain IP addresses show a significant increase in the number of flows, exceeding 1000%. However, these elevated flow counts remain within the range of the average number of flows observed during regular daily traffic.

To quantify the observed behavior, the average server access was computed and compared with the counts of flows from individual IP addresses. IP addresses that had a number of flows significantly higher than the average were earmarked for further scrutiny. Specifically, IPs whose flows were greater than five times the average server access were flagged. This allowed us to narrow down on the potential sources that could be part of the C&C network.

In a C&C attack, the compromised systems often communicate with a command server to receive instructions or send data. The unusual increase in flows from specific IPs to the servers could imply that these systems are compromised and are part of a botnet being used for the C&C attack. This kind of behavior is symptomatic of systems that have been commandeered to perform tasks under the control of a remote attacker.

In conclusion, the suspect IPs should be isolated and thoroughly analyzed for any malicious content or behavior. Additionally, network traffic should be continuously monitored for any unusual patterns, and preventive measures should be in place to mitigate the risk of future C&C attacks. This experience underscores the necessity for dynamic and robust security mechanisms to protect the integrity and confidentiality of data and systems.

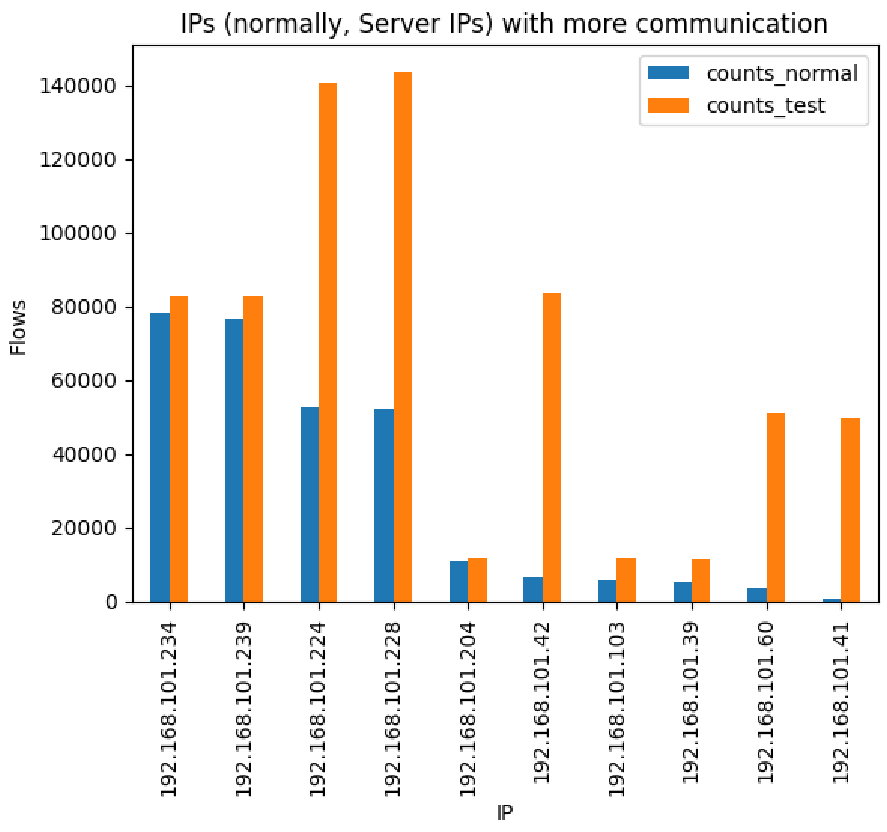


Figure 4.2: Increase of flows of the IPs with the most flows on the test dataset.

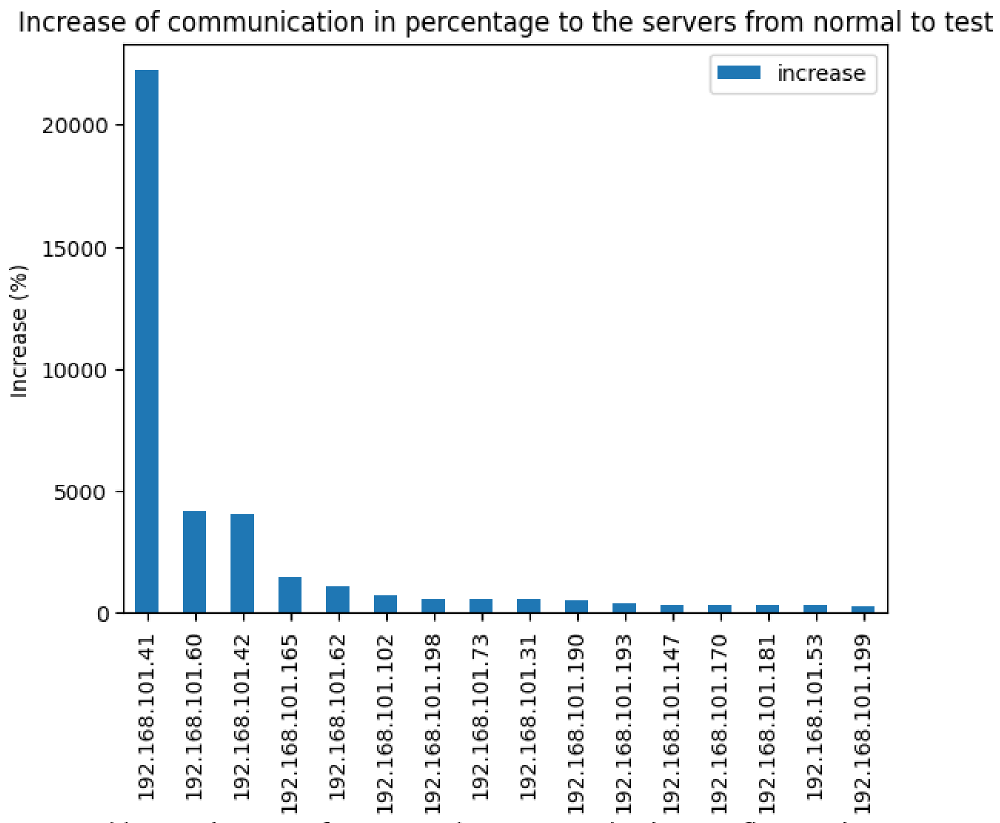


Figure 4.3: Increase of flows from the normal dataset to the test dataset.

4.3 Data Exfiltration

In this section of our analysis, we focus on identifying potential data exfiltration and anomalous external communications. Data exfiltration can be a serious threat as it involves the unauthorized transfer of data from within the organization to an external location, potentially compromising sensitive information.

Initially, the analysis involves identifying new IP addresses that have started communicating externally. The code segregates these IP addresses by comparing the list of source IPs in the current data (external_flows_test) with the source IPs from the baseline data (external_flows). This process is replicated for destination IPs as well. This helps in identifying any new or unusual communication patterns with external IP addresses that were not present before, and could be indicative of a security breach or an unauthorized connection.

Once the new IP addresses communicating externally are identified, the analysis delves deeper by investigating the amount of data these IPs are uploading and downloading. This is particularly important for identifying data exfiltration activities. Specifically, the analysis calculates the total volume of data in megabytes being uploaded and downloaded by these suspect IPs. A sudden increase in uploaded data could be indicative of data being exfiltrated from the network, like it happens in Figure 4.4.

Furthermore, the analysis also investigates the public IP addresses being accessed by these suspect IPs, as well as the countries and organizations associated with these public IPs.

In addition, the analysis involves checking the percentage increase in communication to the exterior by merging the baseline and test datasets and calculating the increase in the number of flows. IPs with an increase greater than 150% and having more than 50 flows are earmarked for closer examination.

Under the Data Exfiltration subsection, the emphasis is on the volume of data being uploaded to external sources. It is important to recognize that data exfiltration usually involves a higher volume of uploaded data compared to downloaded data. The analysis evaluates the average bytes uploaded per flow and looks for substantial increases.

Furthermore, the code looks for periodic communication patterns sending a consistent amount of data, which can be a red flag for automated exfiltration processes.

Towards the end, the analysis involves visual representation through plotting the average uploaded bytes per flow for the IPs involved in data exfiltration. This aids in visual analysis, making it easier to pinpoint anomalies.

In conclusion, this section of the analysis is critical in identifying and understanding any potential data exfiltration or unauthorized external communication activities. Through detailed examination of new IPs, volumes of data being transferred, and the nature of external communication, security analysts can make informed decisions about possible security threats and implement measures to safeguard the network.

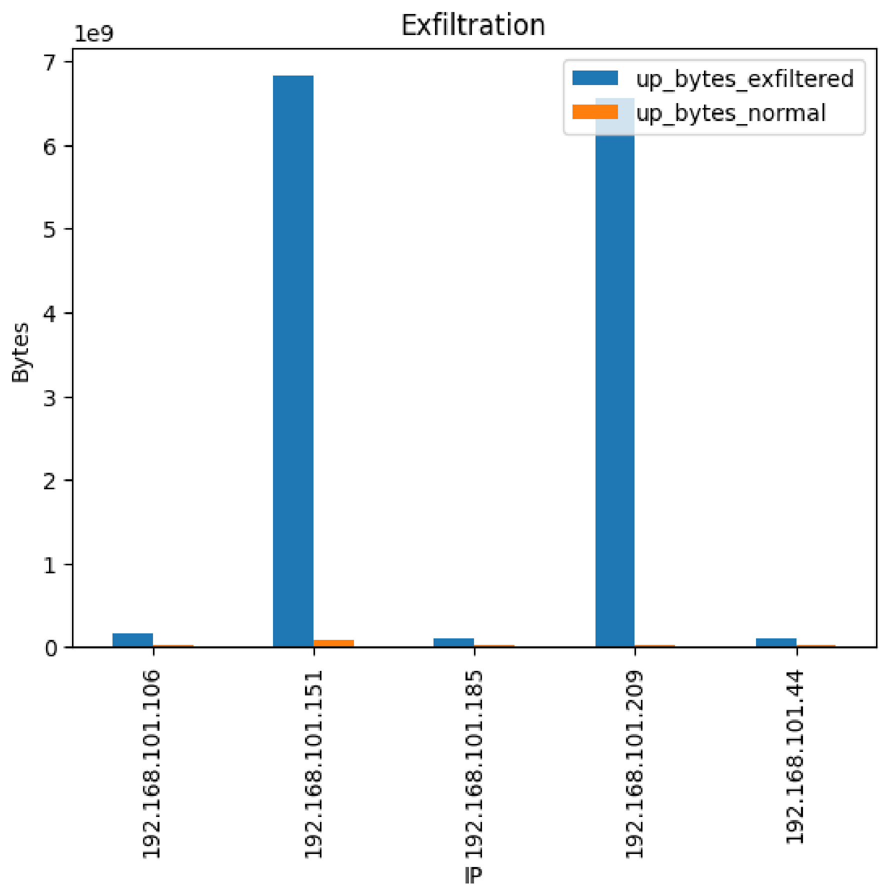


Figure 4.4: Average uploaded bytes.

4.4 Suspicious Country Communications

In this section of the report, we examine the network communication data to analyze the interactions with external countries. This involves understanding the distribution of network flows, uploaded bytes, and downloaded bytes to different countries and identifying any unusual or anomalous communications.

- Data Aggregation and Analysis - First, we aggregate the data by destination country codes (dst_cc) and calculate the total counts of network flows, as well as the sum of uploaded bytes (up_bytes) and downloaded bytes (down_bytes) for each country. For a more detailed analysis, we also calculate the mean of uploaded and downloaded bytes for each country. Local communications, where the destination IP address begins with '192.168.101.', are excluded from this analysis.

- Visualization of Top Communicating Countries - We visualize the data to gain insights into the communication patterns with external countries:

  Number of Flows: We create a bar chart displaying the top 5 countries with the highest number of communication flows. This chart reveals which countries have the most frequent communication.

  Uploaded Bytes: Another bar chart is created to show the top 5 countries where the most data is sent.

  Downloaded Bytes: Lastly, a bar chart displays the top 5 countries from which the most data is received.

  These visualizations help us to understand the usual communication patterns and establish a baseline for normal behavior.

- Comparative Analysis with Test Data - To detect any significant changes or anomalies, we compare the test dataset with the normal dataset.

We then analyze the changes in the number of flows and bytes, and flag countries with unusual increases in communication. A country is considered to have unusual communication if it meets the following criteria:

The number of flows is more than 200, and There is at least a 50% increase in the number of flows, or There is at least a 50% increase in uploaded/downloaded bytes.

Based on the observations from Figure 4.5, we have detected a slight abnormal behavior in communications with China. Additionally, considering the significant amount of traffic to Russia in the anomalous dataset compared to almost no communication with that country in the normal dataset, we recommend blocking communication with Russia.

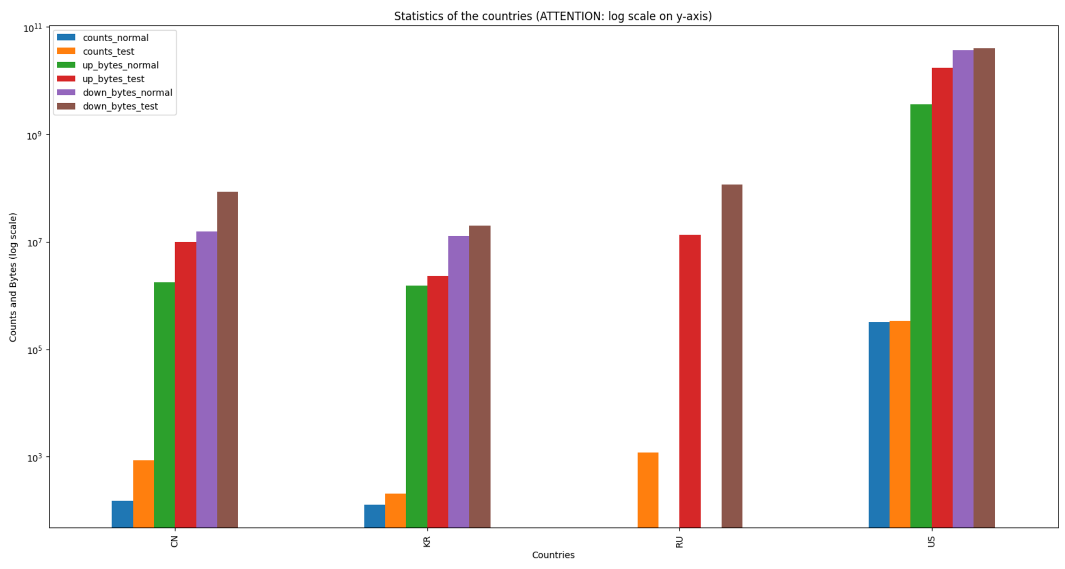


Figure 4.5: Countries Statistics.

Chapter 5

SIEM Rules Definition

A SIEM rule is a predefined set of criteria within SIEM system. It is designed to detect and respond to security incidents by analyzing event data from various sources. SIEM rules monitor specific conditions and patterns to trigger alerts or automated responses (like a communication block). By using SIEM rules, organizations can proactively identify and mitigate security threats.

5.1 Server Access Increase

The following rule aims to address the issue of server access, which will also catch Distributed Denial of Service (DDoS) attacks.

The first step of the rule involves calculating the average number of accesses to specific server IPs, namely 234, 239, 224, and 228, based on the provided dataset. This average serves as a baseline or reference point for evaluating subsequent access patterns.

To identify potential alarm-triggering events, the rule considers any access count that exceeds three times the calculated average. This threshold indicates a significant increase in server access, which may warrant closer attention.

Furthermore, the rule recommends a more stringent response when an IP address is found to have an access count five times higher than the average. This level of activity is considered a more severe threat, indicating a potentially malicious intent. In such cases, the rule advises blocking the IP address to prevent further unauthorized access and mitigate the risk of a potential DDoS attack.

It is important to note that the provided code snippet solely performs calculations and identifies test server access that exceeds the defined thresholds. The code does not include the actual implementation of alarms or IP blocking. Integrating these actions into a SIEM system or incorporating them into subsequent code would be necessary to fully operationalize the rule and respond effectively to potential DDoS threats.

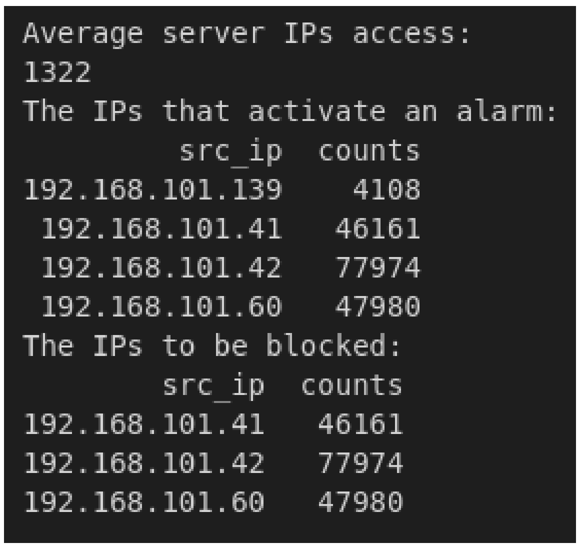
Concluding, this rule checks the increase in server accesses and connections and it triggers an alarm if there are IPs that reach over 3 times the average of volume flows and it blocks them when it is 5 times above it.

```
# Average server IPs access (server IPs: 234, 239, 224, 228)
avg_server_access = data.loc[(data['dst_ip'].isin(server_ips['index']))]
.groupby(['src_ip']).size().reset_index(name='counts')

avg_server_access = avg_server_access['counts'].mean()
print("Average server IPs access: \n" + str(int(avg_server_access)))

# check the test flows that access the server IPs (server IPs: 234, ...
    239, 224, 228) with more than 50 flows and 3 times the average
test_server_access = test.loc[(test['dst_ip'].isin(server_ips['index']))]
.groupby(['src_ip']).size().reset_index(name='counts')

test_server_access = test_server_access[test_server_access['counts'] ...
    > (avg_server_access*3)]
print("Test server IPs access: \n" + str(test_server_access))
```

5.1.1 Rule Testing and Device Identification

When testing the rule we got the following results:

The effectiveness of the rule is evident as it successfully detected and alarmed certain IPs, including those associated with malicious activities. Additionally, it likely flagged IPs that exhibited non-anomalous behavior but warranted further investigation. The rule proved valuable in blocking those IPs that clearly abused server access, ensuring



```
Average server IPs access:
1322
The IPs that activate an alarm:
         src_ip  counts
192.168.101.139    4108
 192.168.101.41   46161
 192.168.101.42   77974
 192.168.101.60   47980
The IPs to be blocked:
        src_ip  counts
192.168.101.41   46161
192.168.101.42   77974
192.168.101.60   47980
```

Figure 5.1: Server Access Increase Rule Result.

that only high-risk entities were blocked, while providing alerts for IPs that required additional scrutiny.

5.2 Internal Communications

The provided rule focuses on monitoring and responding to internal communications within a network, particularly targeting new internal communications that may require blocking. The rule follows a series of steps:

Firstly, it identifies internal communications between private IP addresses on the normal dataset and stores the information in the normal_internal dataframe.

Next, it calculates the average count of internal communications by taking the mean of the communication counts in the normal_internal dataframe. This average serves as a baseline for evaluating subsequent internal communications.

The rule then checks the test dataset to identify new internal communications between private IP addresses. By filtering the test data and grouping it based on the source and destination IP addresses, the rule calculates the size of each communication group, storing the information in the test_internal dataframe.

To determine new internal communications in the test dataset, the rule compares the internal communications from the normal dataset (normal_internal) with those in the test dataset (test_internal). It selects and stores the communications that exist only in the test dataset, indicating new internal communications, in the internal_diff dataframe.

Additionally, the rule checks if any old internal communications from the normal dataset have communication counts exceeding three times the average count (avg_internal). It compares the old internal communications with the new internal communications in the test dataset and selects the old communications that surpass the threshold. These communications are also stored in the internal_diff dataframe.

Finally, the rule outputs the internal communications that triggered an alarm. The information includes the source IP address, destination IP address, and communication count for each internal communication in the internal_diff dataframe.

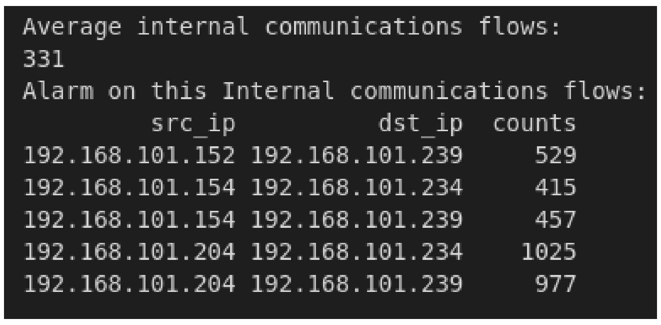
Concluding, new internal connections aren't allowed (it would be blocked like in UA's eduroam) and the ones who were allowed can't pass 3 times the average of server access flows, otherwise it triggers an alarm.

```
# Check for internal comms in normal (private IP to private IP)
normal_internal = data.loc[(data['src_ip'].apply(lambda x: ...
    ipaddress.ip_address(x).is_private)) & ...
    (data['dst_ip'].apply(lambda x: ipaddress.ip_address(x).is_private))]
normal_internal = normal_internal.groupby(['src_ip', ...
    'dst_ip']).size().reset_index(name='counts')
#print("Internal communications in normal: \n" + str(normal_internal))

# Average count of internal comms
avg_internal = normal_internal['counts'].mean()
print("Average internal communications flows: \n" + ...
    str(int(avg_internal)))

```

```
# Check for new internal comms in test (private IP to private IP)
test_internal = test.loc[(test['src_ip'].apply(lambda x: ...
    ipaddress.ip_address(x).is_private)) & ...
    (test['dst_ip'].apply(lambda x: ipaddress.ip_address(x).is_private))]
test_internal = test_internal.groupby(['src_ip', ...
    'dst_ip']).size().reset_index(name='counts')
#print("Internal communications in test: \n" + str(test_internal))

# Get the difference to check the new internal comms in test
internal_diff = pd.merge(normal_internal, test_internal, ...
    on=['src_ip', 'dst_ip'], how='right')
internal_diff = internal_diff.fillna(0)
internal_diff = internal_diff[(internal_diff['counts_x'] == 0) & ...
    (internal_diff['counts_y'] > 0)]
internal_diff = internal_diff[['src_ip', 'dst_ip', 'counts_y']]
internal_diff = internal_diff.rename(columns={'counts_y': 'counts'})
# print("New internal communications flows: \n" + str(internal_diff))

# Check in the old internal comms if there are any that have more ...
    than 3 times the average
internal_diff = pd.merge(normal_internal, test_internal, ...
    on=['src_ip', 'dst_ip'], how='right')
internal_diff = internal_diff.fillna(0)
internal_diff = internal_diff[(internal_diff['counts_x'] > ...
    (avg_internal*3)) & (internal_diff['counts_y'] > 0)]
internal_diff = internal_diff[['src_ip', 'dst_ip', 'counts_y']]
internal_diff = internal_diff.rename(columns={'counts_y': 'counts'})
print("Alarm on this Internal communications flows: \n" + ...
    internal_diff.to_string(index=False))
```

5.2.1 Rule Testing and Device Identification

When testing the rule we got the following results:



```
Average internal communications flows:
331
Alarm on this Internal communications flows:
         src_ip          dst_ip  counts
192.168.101.152 192.168.101.239     529
192.168.101.154 192.168.101.234     415
192.168.101.154 192.168.101.239     457
192.168.101.204 192.168.101.234    1025
192.168.101.204 192.168.101.239     977
```

Figure 5.2: Internal Communication Rule Result.

Using this rule, we were able to detect two distinct categories of countries: those experiencing a substantial increase in flow numbers compared to the average, and those that previously had no communication history but have now initiated communications with an unusually high number of flows (at least 500 flows).

5.3 Naughty Countries

This following rule analyzes the average number of flows per country by grouping the data based on the destination country code (dst_cc). It calculates the mean flow count across countries, providing an understanding of the normal flow level for each country.

Next, it checks the test dataset for countries that have flow counts exceeding 18 times the average or have an infinite number of flows. These countries are flagged as potential sources of concern or anomalies, and an alarm is triggered.

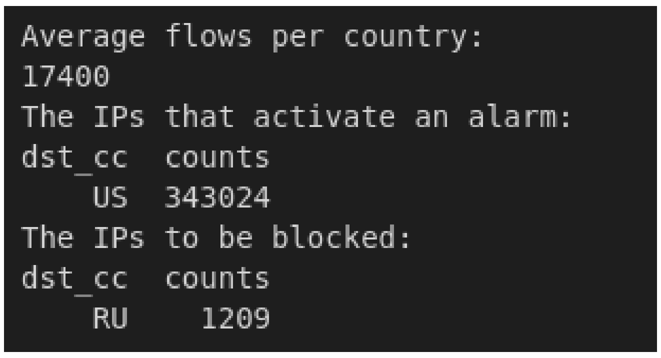
The rule further examines the test dataset to identify countries that exist in the test dataset but not in the normal dataset. These countries are considered new or previously unobserved in the network traffic.

Among the newly identified countries, the rule selects those with flow counts exceeding 500. These countries are deemed to have an unusually high volume of flows and may require blocking or further investigation.

```
# Check the average of flows per country
avg_flows_country = ...
    data.loc[¬(data['dst_ip']).str.startswith('192.168.101.')]
.groupby(['dst_cc']).size().reset_index(name='counts')
avg_flows_country = avg_flows_country['counts'].mean()
print("Average flows per country: \n" + str(int(avg_flows_country)))

# Check if the test has more than 18 times the average OR is INF of ...
    flows per country
test_flows_country = ...
    test.loc[¬(test['dst_ip']).str.startswith('192.168.101.')]
.groupby(['dst_cc']).size().reset_index(name='counts')
test_flows_country = test_flows_country[(test_flows_country['counts'] ...
    > (avg_flows_country*18)) | (test_flows_country['counts'] == ...
    float('inf'))]
print("The IPs that activate an alarm: \n" + ...
    test_flows_country.to_string(index=False))

# Check for countries that exist in test but not in normal
test_flows_country = ...
    test.loc[¬(test['dst_ip']).str.startswith('192.168.101.')].
groupby(['dst_cc']).size().reset_index(name='counts')
test_flows_country = test_flows_country[¬(test_flows_country['dst_cc']
.isin(data.loc[(data['dst_ip']).str.startswith('192.168.101.')]
.groupby(['dst_cc']).size().reset_index(name='counts')['dst_cc']))]

# Check which of those countries have more than 500 flows
test_flows_country = test_flows_country[test_flows_country['counts'] ...
    > 500]
print("The IPs to be blocked: \n" + ...
    test_flows_country.to_string(index=False))
```

5.3.1 Rule Testing and Device Identification

When testing the rule we got the following results:



```
Average flows per country:
17400
The IPs that activate an alarm:
dst_cc  counts
    US  343024
The IPs to be blocked:
dst_cc  counts
    RU    1209
```

Figure 5.3: Naughty Countries Rule Result.

The rule demonstrated its effectiveness by generating an alarm for the increase in flows observed in the USA. This increase is likely considered normal due to the country's role as a hub for numerous data centers and widely used platforms. However, the rule also identified and blocked flows originating from Russia, a country that had not previously been involved in any communications.

5.4 Data Exfiltration

This data exfiltration rule aims to analyze the average upload bytes in the normal dataset. It calculates the total upload bytes for each source IP address (src_ip), calculates the mean upload bytes across all IP addresses, and determines the average upload bytes.

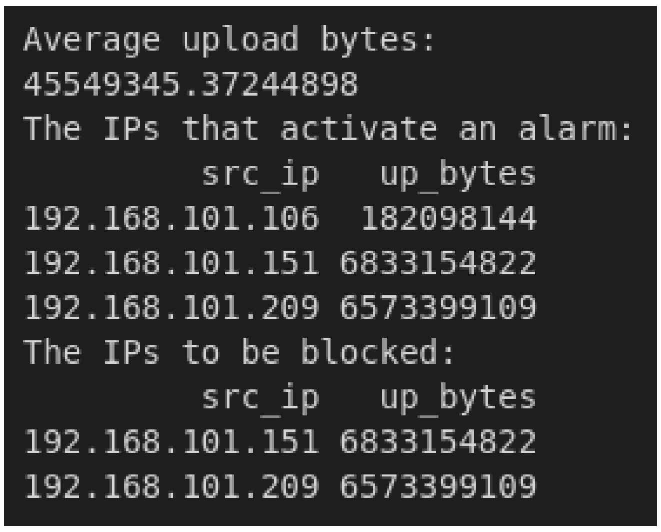
Next, the rule examines the test dataset to identify flows that exceed three times the average upload bytes. It groups the data by the source IP address, calculates the total upload bytes for each IP address, and selects the flows that surpass the threshold.

Additionally, the rule checks for flows in the test dataset that surpass five times the average upload bytes. It follows a similar process of grouping the data by the source IP address and selecting the flows that exceed the defined threshold.

```
# Average upload bytes in normal
avg_up_bytes = data.groupby(['src_ip'])['up_bytes'].sum().
reset_index(name='up_bytes')['up_bytes'].mean()
print("Average upload bytes: \n" + str((avg_up_bytes)))

# Check if there are flows in test that pass 3 times the average ...
    upload bytes
test_up_bytes = ...
    test.groupby(['src_ip'])['up_bytes'].sum().reset_index(name='up_bytes')
test_up_bytes = test_up_bytes[test_up_bytes['up_bytes'] > ...
    (avg_up_bytes*3)]
print("The IPs that activate an alarm: \n" + ...
    test_up_bytes.to_string(index=False))

# Check if there are flows in test that pass 5 times the average ...
    upload bytes
test_up_bytes = ...
    test.groupby(['src_ip'])['up_bytes'].sum().reset_index(name='up_bytes')
test_up_bytes = test_up_bytes[test_up_bytes['up_bytes'] > ...
    (avg_up_bytes*5)]
print("The IPs to be blocked: \n" + test_up_bytes.to_string(index=False))
```

5.4.1 Rule Testing and Device Identification

When testing the rule we got the following results:



```
Average upload bytes:
45549345.37244898
The IPs that activate an alarm:
        src_ip   up_bytes
192.168.101.106  182098144
192.168.101.151 6833154822
192.168.101.209 6573399109
The IPs to be blocked:
        src_ip   up_bytes
192.168.101.151 6833154822
192.168.101.209 6573399109
```

Figure 5.4: Data Exfiltration Rule Result.

The rule proved effective by triggering an alarm for three IPs. Among them, one IP was uploading nearly 2GB of data, which is notable considering the average upload bytes are around 500MB. However, it is important to note that there are instances where IPs upload 1GB or more, so triggering an alarm for this IP is within the normal range. On the other hand, the rule blocked IPs that were uploading over 6GB of data, which is a substantial amount and warrants immediate action.

5.5 New Protocols

To enhance the monitoring capabilities, we implemented a rule specifically designed to identify the use of new protocols that have not been previously observed. The presence of new protocols could indicate novel types of communication or modifications to the system.

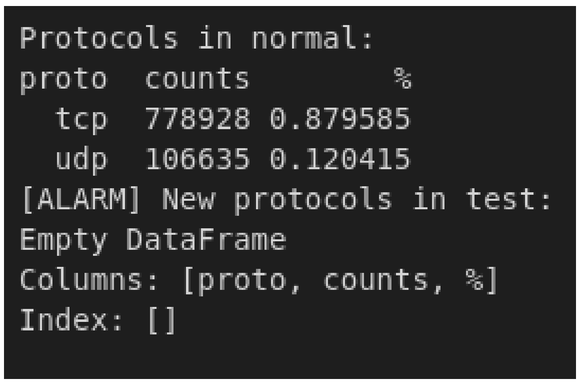
This rule starts by examining the protocols used in the normal dataset. It organizes the data based on protocol type (proto), calculates the count for each protocol, and determines the percentage representation of each protocol within the dataset.

Subsequently, the rule analyzes the test dataset to detect any new protocols. Following a similar procedure as before, it groups the data by protocol type, calculates the count and percentage for each protocol, and merges this information with the protocol data from the normal dataset. The rule retains only the records where the protocol count is zero in the normal dataset but has a positive count in the test dataset.

```
# Check the used protocols
normal_protocols = ...
    data.groupby(['proto']).size().reset_index(name='counts')
normal_protocols['%'] = ...
    normal_protocols['counts']/normal_protocols['counts'].sum()
print("Protocols in normal: \n" + ...
    normal_protocols.to_string(index=False))

# Check if there's a new protocol in test
test_protocols = ...
    test.groupby(['proto']).size().reset_index(name='counts')
test_protocols['%'] = ...
    test_protocols['counts']/test_protocols['counts'].sum()
test_protocols = pd.merge(normal_protocols, test_protocols, ...
    on='proto', how='right')
test_protocols = test_protocols.fillna(0)
test_protocols = test_protocols[(test_protocols['counts_x'] == 0) & ...
    (test_protocols['counts_y'] > 0)]
test_protocols = test_protocols[['proto', 'counts_y', '%_y']]
test_protocols = test_protocols.rename(columns={'counts_y': 'counts', ...
    '%_y': '%'})
print("[ALARM] New protocols in test: \n" + ...
    test_protocols.to_string(index=False))
```

5.5.1 Rule Testing and Device Identification

When testing the rule we got the following results:



```
Protocols in normal:
proto  counts        %
  tcp  778928 0.879585
  udp  106635 0.120415
[ALARM] New protocols in test:
Empty DataFrame
Columns: [proto, counts, %]
Index: []
```

Figure 5.5: New Protocols Rule Result.

As we can see, no results were found because on the datasets provided no new protocols were used. However, it would be important to implement this rule in order to make sure that the network keeps its normal flow.

5.6 New Ports

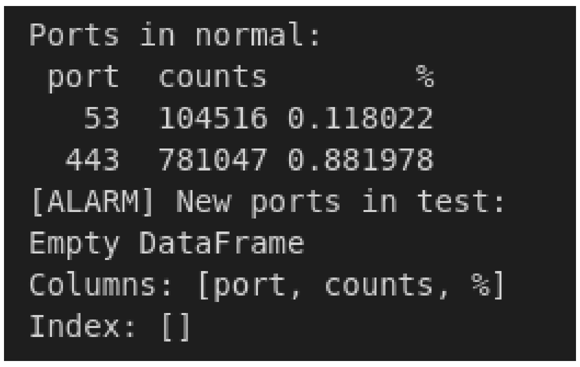
Just like the previous rule (New Protocols), we followed the same methodology for the used ports. The code snippet detects potential port scanning by analyzing the network traffic data and comparing the ports used in normal conditions to those in a test dataset.

In the first stage, the code categorizes the baseline traffic data by ports, tallying the occurrences and calculating the percentage share of each port. The test data undergoes the same process. The code then merges both datasets and filters out the ports that are present only in the test data. These new ports, likely not used under normal circumstances, are flagged with an ”ALARM” tag and displayed alongside their counts and percentage shares. The presence of new ports could indicate a port scanning attempt or other unusual network activities, warranting further investigation.

```
# Check the used ports
normal_ports = data.groupby(['port']).size().reset_index(name='counts')
normal_ports['%'] = normal_ports['counts']/normal_ports['counts'].sum()
print("Ports in normal: \n" + normal_ports.to_string(index=False))

# Check if there's a new port in test
test_ports = test.groupby(['port']).size().reset_index(name='counts')
test_ports['%'] = test_ports['counts']/test_ports['counts'].sum()
test_ports = pd.merge(normal_ports, test_ports, on='port', how='right')
test_ports = test_ports.fillna(0)
test_ports = test_ports[(test_ports['counts_x'] == 0) & ...
    (test_ports['counts_y'] > 0)]
test_ports = test_ports[['port', 'counts_y', '%_y']]
test_ports = test_ports.rename(columns={'counts_y': 'counts', '%_y': ...
    '%'})
print("[ALARM] New ports in test: \n" + ...
    test_ports.to_string(index=False))
```

5.6.1 Rule Testing and Device Identification

When testing the rule we got the following results:



```
Ports in normal:
 port  counts        %
   53  104516 0.118022
  443  781047 0.881978
[ALARM] New ports in test:
Empty DataFrame
Columns: [port, counts, %]
Index: []
```

Figure 5.6: New Ports Rule Result.

As we can see, there isn't any trace of port scanning.

Chapter 6

Conclusion

The project has demonstrated the critical importance of robust SIEM rules in monitoring a network and identifying potential threats. By analyzing a day's worth of typical network traffic, we gained valuable insights into the normal network usage and used it to identify deviations that could indicate potential threats.

In testing these SIEM rules against the "testX.parquet" dataset, we found that our rules were effective in highlighting anomalous network behaviors, confirming the value of a data-driven approach. Despite the complexity of the task, the use of pandas for data analysis, the use of databases for geo-localization based on IPv4 addresses and the used of Jupyter Notebooks were fundamental for a straightforward analysis of the data which greatly simplified the creation of the SIEM rules.

Future work should aim to refine the rules based on ongoing analysis of network. Furthermore, integrating machine learning algorithms could potentially enhance the efficiency and effectiveness of detecting anomalous behaviors.