# Anonymizing Network Traffic Datasets for Network Intrusion Detection

*Thomas Auger*

Master of Science
Computer Science
School of Informatics
University of Edinburgh
2021

# Abstract

Research on network security problems such as network intrusion detection require a dataset of realistic network traffic containing examples of abnormal or malicious activity. However, network traffic typically contains highly sensitive data such as Personally Identifiable Information (PII) and technical information about the network infrastructure which makes sharing such datasets publicly potentially illegal due to data privacy legislation, and a cyber security risk. These barriers have led to a shortage of high quality network datasets for academic research and makes it harder for cyber security companies to provide network security services as businesses are wary of sharing such data.

Using network intrusion detection as a case study this thesis investigates whether it is possible to protect sensitive information in a network traffic dataset using anonymization techniques while maintaining its utility. By devising a threat model and computing metrics, sensitive information is uncovered in a network traffic dataset and potential adversaries who may seek to obtain it are identified. A series of anonymization policies are designed using existing and novel anonymization techniques to protect the sensitive data which are then evaluated and analysed using a combination of privacy and utility measures.

Although the anonymization policies were limited in their ability to improve privacy and maintain utility, the weaknesses in the algorithms and metrics are identified. With confidence in the methodology devised and followed in this thesis for creating and evaluating anonymization policies, suggestions for improvement are made when applying it to future research in this area.

# Acknowledgments

Thank you to my supervisor David Aspinall and co-supervisor Sadiq Sani for their support and advice throughout the project. Thank you also to Robert Flood who generously gave me advice at the start of the project.

Finally, but most importantly, thank you to my family for their unwavering support, patience, and advice during this project and throughout the whole academic year. It is because of them that I have been able to follow my ambitions and complete this master's degree.

# Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Thomas Auger*)

# Table of Contents

# Chapter 1

# Introduction

The increasing number of critical services being digitised and connected to computer networks such as the internet has driven research interest in cyber security to develop methods of detecting and responding to threats. One of the most active research areas is network security and in particular the development of network intrusion detection systems (IDS) where the goal is to detect and respond to malicious network activity such as a computer virus running on a compromised machine, or a hacker attempting to break into or take down services.

Designing and developing an IDS requires access to large datasets containing examples of such malicious network activity. Systems that use machine learning or deep learning require very large datasets in order to train their models. However due to privacy concerns there virtually no high quality real world datasets in the public domain. In the commercial world many businesses wish to outsource their network monitoring and security to a third party security service which requires sending transcripts of their network traffic. However the risk of leaking personally or commercially sensitive information puts many of them off.

This situation motivates research into methods for sharing network traffic datasets to perform analytics, in particular network intrusion detection, without the risk of leaking sensitive information from the dataset. One of the most researched methods is *anonymization* where sensitive information is removed or obfuscated. However applying anonymization effectively has proved to be subtle and challenging. Network communications over the internet are transmitted over seven layers (the OSI model) of protocols, each containing information about the payload and its destination which may leak sensitive material.

Even when the sensitive information has been identified the challenge of anonymiza-

tion is to remove it without impacting the utility of the dataset. Simply deleting everything in the dataset will ensure that no sensitive information is leaked, but that won't make it useful for analytics.

The goal of this project is to explore and attempt to answer the following question:

*Is it possible to anonymize a network traffic dataset such that it can be shared with untrusted parties and retain its utility for network intrusion detection?*

This thesis contributes the following towards this goal:

1. An analysis of the sensitive information inside a network traffic dataset and a threat model with profiles of adversaries who would seek to obtain it.
2. Existing and novel anonymization techniques are devised to protect the sensitive information identified in the analysis.
3. The anonymization techniques are evaluated for privacy and utility on the CICIDS17 network intrusion detection dataset.
4. The difficulties and limitations of the anonymization techniques, experiments, and evaluation methods are analysed, with further research directions proposed.

Section 2 explores the previous work and provides the necessary background on anonymizing network traffic. Section 3 investigates what sensitive information is contained in a network flow dataset and PCAP dataset, defines a threat model of adversaries who would attempt to obtain it. Section 4 designs anonymization policies to protect the sensitive information and devises experiments to evaluate their ability to enhance privacy and preserve utility. Section 5 presents the results of the experiments and evaluates them in context of the threat model and objectives of the project. Finally Section 6 draws conclusions and proposes further research directions.

# Chapter 2

# Background

Research in anonymization gained considerable attention in the 2000s with many advances made during that decade. The research interest was driven in part by high profile cases of public datasets being deanonymized such as the Netflix Prize where Arvind Narayanan and Vitaly Shmatikov showed how to reidentify users and what movies they had watched, which in the USA is illegal to disclose, causing embarrassment and legal consequences to Netflix [15]. Cases like this demonstrated the difficulty in knowing whether a dataset has been suitably anonymized and whether it's still possible to recover sensitive information.

## 2.1 Anonymization Techniques for Network Data

While numerous techniques were developed to anonymize microdata (tabular) datasets such as the Netflix Movie database and medical records, Coull et. al. pointed out that it was not obvious how to apply such techniques to network traffic datasets due to their inherent complexity [5]. While anonymizing health records may seek simply to prevent identification of the patient, anonymizing network traffic may seek to protect not just the identities of users, but also user behaviour, which applications and services are running, and how the network is secured. Consequently many published real network datasets are heavily anonymized to remove as much identifiable information as possible. Unfortunately this tends to remove so much data as to make them unusable for research.

To set some parameters for what sensitive information should be protected Pang et. al. suggested three necessary criteria that an adversary should not be able to determine: 1) the identities of specific hosts such that they can be used to reveal user behaviour, 2)

the identities of internal hosts to be able to determine the internal network topology and which services are running, and 3) details of security practices that wouldn't otherwise be known [16]. To prevent host and user identification the most commonly applied technique is to anonymize the IP addresses and timestamps, however as analysed in detail by Slagell et. al. there are numerous pitfalls, particularly the ability to derive user profiles, that must be avoided to prevent deanonymization attacks [22].

The most sophisticated IP anonymization techniques are prefix-preserving algorithms that anonymize the subnet and host portions of the IP address separately and in a consistent manner such that machines in the same subnet have anonymized addresses in the same subnet. One of the most popular implementations Crypto-PAn uses a cryptographic hash to compute the anonymized addresses [10]. There are a variety of approaches to anonymize timestamps, some of the most popular being 'annihilation' of the least significant portions to reduce the precision of the timestamp to minutes or hours, and replacing timestamps with sequence counts [2]. Burkhart performed a thorough analysis on the effectiveness of IP address anonymization for network anomaly detection comparing prefix-preserving, partial prefix-preserving and deletion of IP addresses [1]. His results showed that while all anonymization methods showed a drop in utility, prefix-preserving anonymization performed best and deleting addresses performed the worst, however for partial-prefix preserving the performance depended on the types of anomaly being detected.

Although much attention has been placed on IP addresses there are numerous other attributes that may be sensitive. Coull et. al. demonstrated how to infer sensitive information such as the network topology from the dataset [7]. Van Dijhuizen et. al. performed a recent survey of anonymization techniques for network datasets and discuss the 'field sensitivity' issue for a plethora of protocols [8]. Evaluating the sensitivity of all the fields and how they are interrelated is an open research problem.

## 2.2 Measuring Privacy and Anonymity

One of the first measures of privacy, developed by Latanya Sweeney after deanonymizing a medical dataset by correlating with other public data, is $k$-anonymity [18]. Given a tabular dataset, where rows are records and columns are attributes, the attributes can be classified into one of three types: identifiers, quasi-identifiers, and sensitive attributes. Identifiers uniquely identify a record in the dataset, quasi-identifiers do not uniquely identify a record but if correlated with other datasets could lead to reiden-

tification, and sensitive attributes are data that is unique to the dataset that would be damaging if associated with the identifier. Classifying the attributes of a dataset into these categories is a manual process and there is no precise way of identifying whether a given attribute belongs to one of the categories.

Once the attributes are classified the records can be clustered into equivalence classes where the quasi-identifiers of each record are equal to each other. The $k$-anonymity condition states that each of these equivalence classes contains at least $k$ records. Thus if another dataset is found with records containing the quasi-identifiers, then there are at least $k$ records in the original dataset that could be associated with them. As an example consider a dataset with attributes source IP (identifier), timestamp (quasi-identifier), and website visited (sensitive attribute). $k$-anonymity says that for every set of records with the same timestamp there are at least $k$ different source IPs represented. For a high enough value of $k$ this makes it much harder to deidentify records through inference with other datasets. Determining an appropriate value of $k$ is an inexact science and depends on the dataset, but in general the larger the value of $k$ the stronger the anonymity condition as it becomes harder to distinguish individual records from each other in the equivalence class.

One of the limitations with $k$-anonymity is if the equivalence classes have small variation in the values of the sensitive attributes as this can reveal information about the individuals represented in that class. Using the same example as before, if all the records with timestamp at 10AM are from the same website this could reveal information about the users represented by the source IPs. The $l$-diversity condition strengthens $k$-anonymity by requiring that the sensitive attributes in each equivalence class take at least $l$ distinct values [13]. Determining a suitable value of $l$ is also an inexact science and depends on the dataset and anonymization procedure.

For completeness, there is a further condition $t$-closeness which requires that the distribution of sensitive values in each equivalence class be at most distance $t$ from the distribution of sensitive values across the entire dataset [12]. Thus, a smaller value of $t$ implies that breaking the data down into finer equivalence classes based on quasi-attributes does not reveal additional information about that class compared with the entire dataset. Although this strengthens anonymity even further, a poorly chosen value $t$ risks destroying useful patterns in the dataset. As with the other conditions there are no heuristics for choosing the value of $t$, and it ultimately depends on the dataset and its desired utility.

However, even when armed with these privacy measures, the challenge with apply-

ing them to network datasets is the complexity of sensitive information in them. As will be explored in Section 3.5 sensitive information is not limited to individual attributes and includes information derived from multiple records such as profiles of traffic between hosts. Coull et. al. attempted to measure anonymity of network datasets by computing information theoretic statistics such as entropy and mutual information between attributes [6]. However their methods are computationally intensive making them unsuitable for large datasets, and their analysis is limited to IP address anonymization.

## 2.3 Measuring Utility for Network Intrusion Detection

The challenge of anonymization is to delete or obfuscate sensitive information to preserve privacy while keeping enough data intact to retain its utility. This is known as the privacy-utility trade off, where stronger anonymization typically results in degraded utility. Thus when devising an anonymization policy it is vital to measure its utility to ensure the dataset is still useful. One approach is to develop a global utility measure such as by Woo et al. [26], however such methods can only be applied to microdata (tabular) datasets, and since they try to factor in multiple measures of utility they can be difficult to interpret.

For network intrusion detection the prevailing approach in the literature is to use an off the shelf intrusion detection system (IDS) and use the number of alerts detected as a utility measure. For example, Lakkaraju et. al. compare the number of alerts of Snort [23] on the anonymized and unanonymized data, where a decrease in the number of alerts is interpreted as a loss of utility [11]. However their analysis uses the DARPA 1999 dataset which doesn't contain the same patterns of network traffic seen in modern networks and they don't consider the effectiveness of the anonymization methods applied. Burkhart et. al. performed a similar analysis on private dataset from a Swiss telecoms company using the Kalman filter for anomaly detection with more convincing success [2].

While running the dataset through an IDS is likely the best way to measure utility for intrusion detection, the limitation is that the IDS will not necessarily use all the fields in the dataset. With systems that use neural networks understanding what parts of the dataset contributed to the decision to raise an alert or not is also a major challenge. Thus when using this approach it is important to be cautious about claiming whether applying an anonymization algorithm resulted in some or no change in utility if the part of the dataset that's anonymized isn't being used.

## 2.4 Network Intrusion Dataset

Obtaining a suitable dataset for this project proved challenging. Ideally the dataset would not be anonymized and contain a rich amount of sensitive information. However, by definition such datasets are not made publicly available. Unfortunately it was not possible to obtain a private dataset in the time available. Instead the only viable alternatives were synthetic datasets generated for the purpose of network intrusion detection, however as they are generated using software automation they don't typically contain much sensitive data.

The dataset chosen for this thesis is CICIDS17 the Intrusion Detection Evaluation dataset by the Canadian Institute for Cyber Security [20]. The dataset is synthetically generated using statistical profiles of users and attackers to simulate benign and malicious behaviour. The infrastructure approximates a small business network comprising of three servers, a firewall, two switches, a domain controller, and ten PCs, see Figure 2.1. The adversaries' network contains a router, one switch, and four PCs.

The dataset comes in two forms: network flow data: a tabular format where each row represents a summary of the traffic communicated over a single session, and PCAP data, a binary format containing every packet that was sent over the network. The traffic was captured over five days between 9AM and 5PM at one of the network switches, and stored in a single PCAP file for each day. The PCAP data was transformed into network flows using the CICFlowMeter tool [3] and published as a single CSV file per day. Each flow contains over 80 features summarizing the network packets sent, the complete list is in Appendix A.

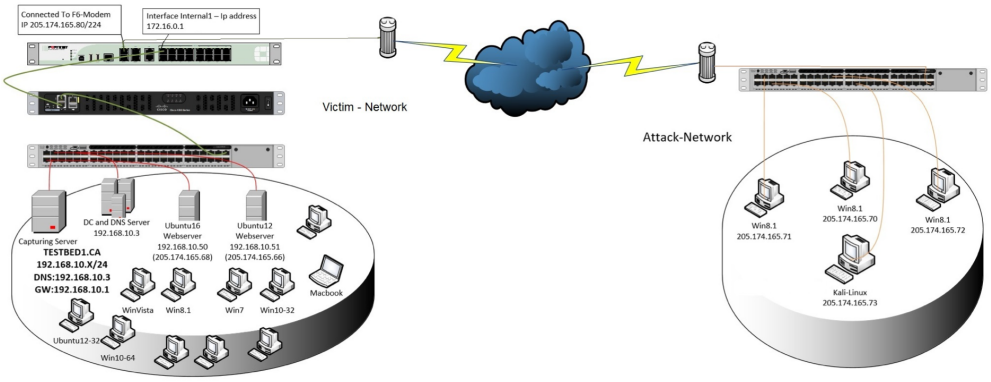


Figure 2.1: Network architecture used to generate the CICIDS17 dataset [20, p.g. 112]

# Chapter 3

# Analysis of the Dataset and Requirements

## 3.1 Requirements and Objectives

As stated in the introduction, the goal of this project is to determine whether it's possible to anonymize a network traffic dataset to protect sensitive information while maintaining its utility for network intrusion detection. This aim is broken down into the following objectives:

1. What sensitive information may exist in a network traffic dataset that requires protecting?
2. How can this sensitive information be anonymized while preserving the utility of the dataset?
3. Is it possible to measure whether the dataset has been successfully and sufficiently anonymized?
4. How can the utility of the dataset for network intrusion detection be measured?

These objectives will be explored using the CICIDS17 dataset (section 2.4), where both forms of the dataset, network flows and PCAPs, will be analysed.

To achieve the first objective a threat model is devised to set the context for how the dataset is intended to be used and profiles of potential adversaries and what sensitive information each would hope to obtain in section 3.3. Alongside this an analysis of the CICIDS17 dataset is performed to compute metrics that identify and measure sensitive information contained in it in section 3.5.

To achieve the second objective, section 4 several anonymization policies using existing and novel anonymization algorithms are defined along with methods of evaluating their ability to obfuscate the sensitive information and retain utility. Finally to achieve the third and fourth objectives section 5 presents the results of experiments along with an analysis and evaluation with respect to the threat model.

## 3.2 Threats, Risks, and Consequences

As observed by Coull et. al. before considering anonymization algorithms and policies it is vital to explore and state the security problem that is being tackled and come up with requirements in order to design a suitable solution [5]. Devising a threat model explores the scenarios in which the security problem under consideration may occur, considers the risks of such events, and evaluates their impact. Using the knowledge gained from this exercise it is hopefully easier to select the scenarios which are most probable and most desirable to prevent.

In this thesis the security problem is *privacy failure* or *data breach*, where sensitive information is leaked or discovered by persons to whom the data should not have been attainable. The type of data under consideration is network traffic captures taken from a business or enterprise network, which may either be PCAPs containing full traffic captures or network flows which do not contain the payloads and instead contain summaries of communications.

There are several types of sensitive data contained in such datasets that may be undesirable to disclose:

1. *Personally Identifiable Information* (PII) is information that uniquely identifies an individual, e.g. names, email addresses, phone numbers, IP Addresses, etc.

2. *Personally Sensitive Data* is information about individuals which should be given extra protection such as ethnicity, political views, web browsing history, etc.

3. *Corporate Sensitive Data* is information about a business that it does not want to disclose, e.g. financial performance, intellectual property, etc.

4. *Cyber Security Data* is information about the infrastructure and applications used to secure the network, e.g. firewalls, VPNs, network topology, etc.

5. *Customer Sensitive Data* is information about customers or users, e.g. customer PII, user credentials, usage statistics, behaviour profiles, etc.

The consequences of disclosing each type of data vary. Disclosing PII and personal sensitive data could have serious legal and financial repercussions if it breaches data protection laws. Leaking corporate sensitive data such as intellectual property could give competing companies an advantage. Revealing cyber security data could make it easier for criminals to launch cyber attacks and demand ransom. A data breach of customer sensitive data could be damaging to the reputation of the business if customers lose confidence.

The risk associated with revealing each type of sensitive information is also dependent on the scenario. If this work was being undertaken for a client the level of risk would be determined by them. In the absence of this a fictitious scenario is created to illustrate how to devise and reason about each kind of threat.

## 3.3 Threat Model

To provide background and context for the threat model the following fictitious scenario is considered:

*Software Enterprises is a small/medium sized company with a corporate network at their office where each employee has a dedicated machine used solely to do their daily work. In addition to employee machines the network contains several servers to provide services such as DNS, Active Directory, and a web server which hosts the companies product accessed and used online by their customer's daily.*

*Software Enterprises want to ensure their network is secure and has not been compromised, so they signed a contract with Network Forensics, a cyber security services company, who installed a dedicated server planted in Software Enterprise's network that captures all traffic, saves it to PCAP files, and sends it to Network Forensics for analysis. Any malicious findings are reported back to Software Enterprises who then decide what action to take.*

*To further research in network security Software Enterprises also share their dataset with academic researchers. Software Enterprises also has a rival, Micro Services, who develop similar technology.*

Using this scenario the profiles of various users of the dataset are created:

- *Alice* is an academic researcher in privacy and security who has been granted access to the dataset for her research. She is trying find weaknesses in the anonymization process applied to the dataset to reidentify users and find sen-

sitive data, and if successful publish a paper.

- *Bob* is software engineer at Software Enterprises who has access to both the dataset for debugging issues and his work machine with access to the company network, but does not have access to servers, log files, or other data. Bob wants to use the dataset to spy on some of his colleagues to find out what they've been looking at online.

- *Catherine* is a system administrator in charge of the cyber security infrastructure at Software Enterprises who has access to the dataset and admin access to all the servers, network infrastructure, and logs. Although it's not official company policy, she wants to use the logs to identify colleagues abusing the network and block inappropriate use of the network.

- *Daniel* is an engineer at Network Forensics with knowledge of the company network infrastructure and access to the dataset, but doesn't know the identities or habits of the people who work there. He has access to the network telemetry received from Software Enterprises and can use it for debugging issues with the threat detection software.

- *Eliza* is a business analyst at Micro Services. She wants to poach customers from Software Enterprises and improve Micro Services' products to be more competitive.

- *Frank* is a cyber criminal who has obtained a copy of the dataset. He wants to make money by either extracting Personally Identifiable Information and selling it or by launching cyber attacks on Software Enterprises and demanding ransom.

Each user presents potential threats and risks associated with sensitive information in the dataset. Not all of the users will have malicious intentions and neither will all of them be breaking the law, but there may still be consequences for Software Enterprises which must be considered.

*Alice* is keen to find weaknesses in the anonymization process and recover as much sensitive information as possible. If successful she wants to publish a paper revealing the techniques she developed and the information she was able to recover. This would give Alice good publicity as a researcher, but also be damaging and embarrassing for Software Enterprises.

*Bob* uses the dataset for testing his code. Given his ability to access the company network it would be easy for him to correlate IP addresses with users and find out more about what his colleagues are up to. The ability for any employee at the company to snoop on their coworkers online activities just by accessing the dataset is a breach of trust that should be prevented.

*Catherine* doesn't want people outside the company to know exactly how their systems are secured as this could be leveraged to launch a cyber attack against them. Such an attack could be devastating, resulting in the leak of customer data and disabling Software Enterprises products. Unlike Bob, Catherine already has access to logs and data sources that record network traffic and could be used to reveal employees habits on the network, so having an extra dataset doesn't change this capability.

*Daniel* uses the dataset from Software Enterprises for testing Network Forensic's products to make sure they continue to work when they add new features. Network Forensics signed a Non Disclosure Agreement with Software Enterprises to ensure that their dataset is never shared with anyone else and it's in Daniel's interest to uphold that agreement.

*Eliza* is looking to gain a commercial edge on Software Enterprises. Using the dataset she may be able to determine what customers Software Enterprises is negotiating with and offer a more competitive deal. She may also be able to determine details of Software Enterprises products that Micro Services could copy and improve to gain a commercial advantage.

*Frank* is looking to obtain sensitive information such as names, addresses, and payment details that he can use or sell to make money. Using the dataset Frank may be able to determine the entry points into Software Enterprise's network, allowing him to launch a cyber attack that he can use as leverage to demand a ransom payment.

The security problem and requirements can now be stated concretely as *mitigating and thwarting the risks of sensitive data disclosure posed by each user in the threat model*. While devising a threat model in this manner is more instructive than applying data protection strategies at random and hoping for the best, it suffers from the limitation that it's only as good as the imagination of those who created it. An important user profile may be missing and some threats might not have been thought about. The best way to mitigate this is to get the model reviewed by other stakeholders to ensure it is as thorough as possible.

## 3.4 Anomalies and Corrections to the Dataset

Early the project it was found that Engelen, et al. had published a paper discovering flaws in the way the CICIDS17 flow data was compiled from the PCAPs, providing fixes for CICFlowMeter and publishing a corrected version of the flow data [9]. This section is an aside from the main thread of the report to note observations made in the differences between the uncorrected and corrected versions of the dataset.

The main flaw identified by the researchers was the incorrect handling of FIN and RST packets by CICFlowMeter of terminating TCP connections resulting in a large number of superfluous flows going in the wrong direction. The difference the fixes make to the flow dataset are profound. Taking the Monday dataset (which contains only benign traffic) as an example the corrected version contains 30% less flows than original. While the flows from the uncorrected flows originated from 8239 source IPs, the corrected flows originate from just 30, almost all of which are private IPs from the internal network, see Table 3.1.

Focusing just on TCP flows, the mean number of FIN packets is almost 2 in the corrected dataset, as expected by the TCP protocol, compared with 0.04 in the original dataset. Most TCP flows in the original dataset simply didn't contain any FIN packets. This corroborates with the reported corrections, removing invalid TCP flows originating from the terminating host that just contain the FIN and ACK packets from the termination handshake.

| Dataset | No. Flows | No. Unique Source IPs | Mean FIN Flag Count of TCP Traffic |
|---|---|---|---|
| Original | 529,918 | 8,239 | 0.04 |
| Corrected | 371,749 | 30 | 1.9 |

Table 3.1: Comparing the original CICIDS17 dataset with the corrected version by Engelen et. al.

For the remainder of this thesis the corrected version of the CICIDS17 dataset is used and will continue to be referred to by that name.

## 3.5 Metrics on Network Flow Data

As Coul et. al. observe, in order to devise an anonymization policy it must first be determined what sensitive data should be protected [5]. To achieve this the CICIDS17

dataset is explored by computing numerous metrics to measure the information contained in it. Given the goal is to look for sensitive data the metrics are selected based on the amount of information they reveal about individual hosts or services. Later on the metrics will also serve as a rapid test to explore how different anonymization policies alter the data.

The network flow dataset consists of five CSV files, one for each working day, where each row describes a single flow itself described by 83 features including source and destination IPs, port numbers, protocol, and timestamps.

### 3.5.1 Basic Statistics

To get a feel for the size and complexity of the dataset some basic statistics were computed for the Monday dataset, which contains only benign flows and the Tuesday dataset which contains FTP and SSH brute force attacks. The results are displayed in table 3.2. Of the approximately 350,000 flows in the dataset almost all originate from internal IP addresses with only a few communications originating from public IPs. Most of the traffic is sent and received by a single IP address (192.168.10.3), the domain controller and DNS server. The most common source port is 123, the Network Time Protocol, while the most common destination port is 53, DNS.

| Statistic | Monday Dataset | Tuesday Dataset |
|---|---|---|
| Total Number of Flows | 371,749 | 322,003 |
| Number of Unique Source IPs | 30 | 22 |
| Number of Unique Destination IPs | 9698 | 8490 |
| Most Common Source IP | 192.168.10.3 | 192.168.10.3 |
| Most Common Destination IP | 192.168.10.3 | 192.168.10.3 |
| Most Common Source Port | 123 | 123 |
| Most Common Destination Port | 53 | 53 |
| Total Flow Bytes (over all flows) | 189GB | 189GB |
| Mean Flow Bytes | 533KB | 547KB |
| Ratio of TCP:UDP | 0.66 | 0.61 |

Table 3.2: Basic statistics for Monday and Tuesday in the network flow dataset

In the process of computing Total Flow Bytes and Mean Flow Bytes it was discovered that some of the flows had invalid *Flow Bytes* and *Flow Packets* equal to infinity

and *Flow IAT* equal to NaN. The Monday dataset had 60 such flows while the Tuesday dataset had 20 with a mixture of both TCP and UDP traffic. Further investigation revealed that these flows represented a single packet travelling in one direction and while the *Backward* and *Forward* features were correct, these aggregated flow statistics were invalid. This is potentially a bug in CICFlowMeter that is either creating invalid flows with just a single packet, or incorrectly computing these features in this scenario. Investigating the cause was out of scope and wasn't investigated further. Since there were only a small number of anomalous flows compared to the size of the dataset they were removed when calculating these statistics and will be excluded from the dataset throughout the rest of this thesis.

### 3.5.2 Determining Sensitive Attributes

A first step to finding sensitive information in a dataset is to determine which attributes are themselves sensitive. Coull et. al. describe a process for achieving this where attributes are classified as identifiers which uniquely identify a flow, or quasi-identifiers (or key fields) that could be correlated with other datasets to identify the flows [5].

In this dataset each flow can be uniquely identified by a tuple (Src IP, Dest IP, Src Port, Dest Port, Protocol, Timestamp). In particular the IP addresses uniquely identify a host and hence the user too. The unique source IPs are of particular interest as each day of the dataset only contains around 30 of them, the majority being internal addresses. Attributes such as the port numbers, timestamp, and flow bytes could be correlated with other log files to recover the identities of the hosts the flow was sent between, hence are candidate quasi-identifiers.

One question that naturally arises is whether it's possible to derive profiles of behaviour for each source IP using the other features which could be invariant under anonymization of the IP addresses. This motivates an exploration to define metrics computed per source IP and compare the differences between them.

### 3.5.3 Network Topology

The network topology describes the connections between hosts and the applications installed on them, for example figure 2.1 shows the topology for the CICIDS network. It is trivial to reproduce the topology of the internal network from the flow data by drawing a graph where vertices represent hosts and edges represent existence of a flow between them. The idea was to investigate whether it is possible to recover the topol-

ogy after anonymizing the dataset using inspiration from the methodology by Coull et. al. [7]. However after an investigation this became impractical as the topology of the CICIDS17 dataset is too small and either became trivial to redraw the graph if the relationship between IPs was preserved or impossible if the IPs are heavily redacted or removed, and was dropped in favour of looking at anonymization techniques more broadly.

### 3.5.4 Host Profiling

A single network flow can be associated with user using an identity field such as the source IP address, while the other non-identity fields such as port number and packet counts are not sufficient. However given a collection of flows and measuring patterns in these non-identity fields it may be possible to build up a profile of network behaviour for each host which could then be used to re-identify users after the identity fields have been anonymized.

As described by the CICIDS17 dataset authors, the benign and adversarial traffic was generated by creating collections of alpha (adversarial) and beta (benign) user profiles based on real captured behaviours and then applying statistical algorithms to generate synthetic patterns that are close to the real ones [20]. This enabled the generation of realistic network traffic without relying on real user data avoiding privacy concerns [21]. This motivates creating metrics across non-identity attributes per source IP address to try and recreate these host profiles.

Since almost all of the flows originate from the 192.168.10.x subnet, the flows with source IPs in this range were used to recreate the host profiles. While calculating the following metrics it was observed that the host 192.168.10.1 contained a single flow to and from port 0. This host is the capturing server that monitored and collected all the packet traces and isn't part of the simulated network used to generate the dataset [20] and hence was removed before calculating any of the statistics. To compare and validate observed patterns the statistics were generated using the Monday, Tuesday, and Wednesday datasets, of which the first contains only benign flows and the latter two contain both benign and attack flows.

### 3.5.5 Metric 1: Number of Flows per Internal Source IP

This metric computes the total number of flows per internal source IP, breaking them down by protocol as shown in Figures 3.1 and 3.2.

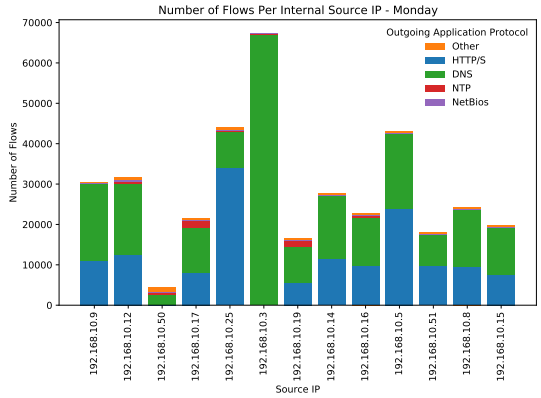


Figure 3.1: Monday flow breakdown

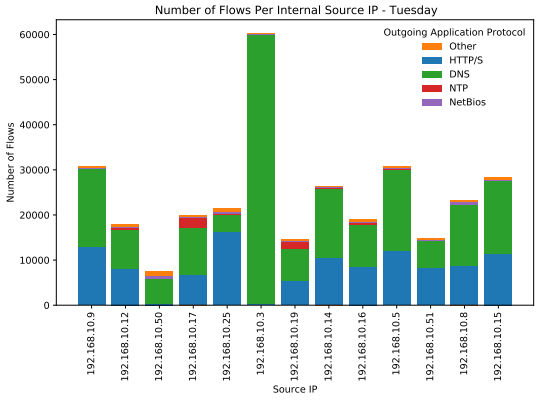


Figure 3.2: Tuesday flow breakdown

The most striking observation is that the host 192.168.10.3 participates in the most flows which are almost exclusively DNS traffic, giving a big clue to its purpose as the DNS server. All the other hosts have varying distributions of DNS and HTTP/S traffic with approximately the same total number of flows but none stand out in a manner that could clearly fingerprint a particular host or give a clue to its role in the network.

### 3.5.6 Metric 2: Flow Distribution Per Hour

This metric is a variant of Metric 1 and computes the proportion of flows for each machine during each hour of the day. Although the dataset authors claim to have recorded data between 9AM and 5PM [20] this metric shows some have a significant number of flows between 8-9AM too. Charts are displayed in figures 3.3 and 3.4.

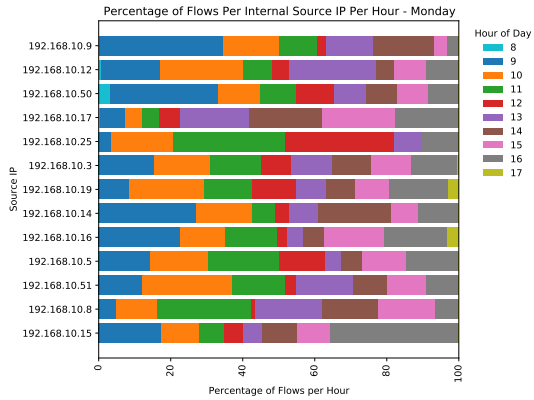


Figure 3.3: Monday proportion of flows per hour

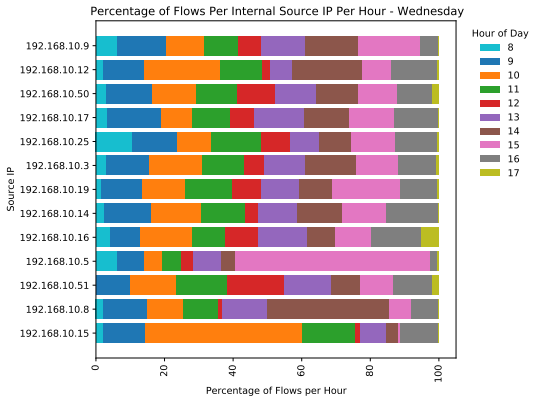


Figure 3.4: Wednesday proportion of flows per hour

Observe that most flows are between 9AM and 12PM, significantly dropping be-

tween 12PM-1PM, and then recovering in the afternoon. These likely indicate patterns of work behaviour where users begin activities at the start of the day, take a break for lunch, and then resume until they leave. Some hosts show unusual fluctuations in the proportion of flows in each hour, for example 192.168.10.25 has a high proportion of flows in the middle hours of the day. This machine is the only Mac computer in the network, but there's not enough data here to confirm whether there's a strong correlation between that fact and the proportion of flows.

### 3.5.7 Metric 3: Outgoing Flows by Protocol

This metric breaks down number of outgoing flows per protocol for each host and displays them as proportion of the entire days traffic, see figures 3.5 and 3.6. The protocols are identified using port numbers, however this isn't foolproof as it's possible to obfuscate communications over different ports, for example encrypted DNS is transmitted over port 443 rather than 53. In this analysis it is assumed all traffic for a particular port is of the same commonly defined type.

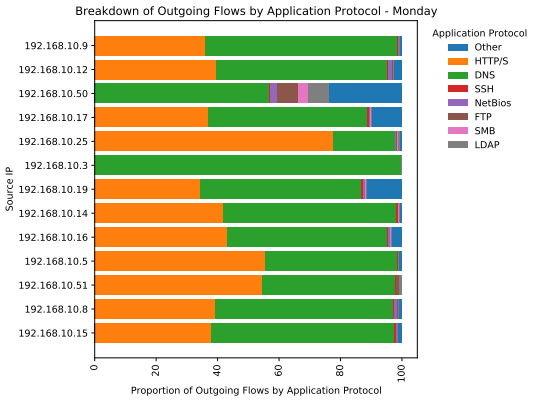


Figure 3.5: Proportion of outgoing flows by application protocol for Monday

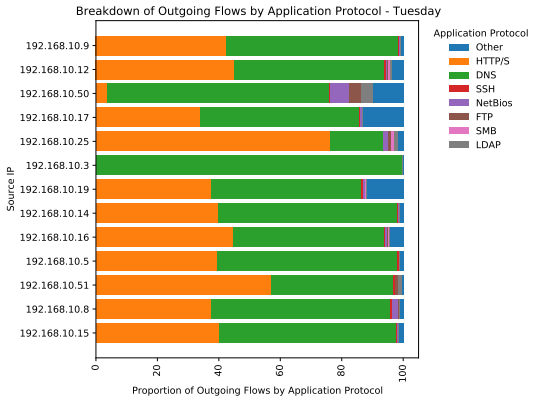


Figure 3.6: Proportion of outgoing flows by application protocol for Tuesday

As observed in section 3.5.5 the host 192.168.10.3 initiates almost exclusively DNS traffic. Another interesting host is 192.168.10.50 which initiates very little HTTP/S traffic and instead has a significant amount of FTP and NetBios traffic compared to the other hosts, clearly singling it out. This machine happens to be the sole web server in the network, the FTP connections likely being other users accessing it for development purposes. As seen previously the other hosts mostly initiate HTTP/S and DNS traffic, but nothing statistically significant to uniquely identify a particular machine.

### 3.5.8 Metric 4: Incoming Flows by Protocol

In the same vain as metric 3 a breakdown of flows per incoming protocol for each host are computed identifying protocols by port number, see figures 3.7 and 3.8.

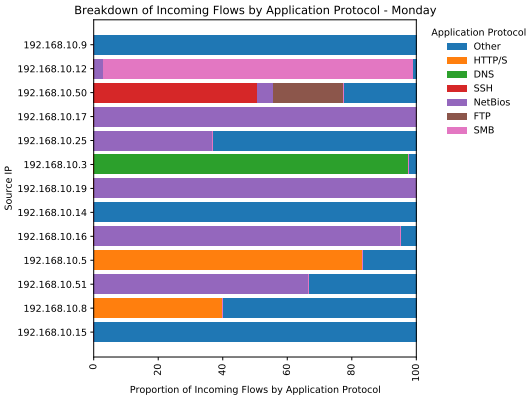


Figure 3.7: Proportion of incoming flows by application protocol for Monday

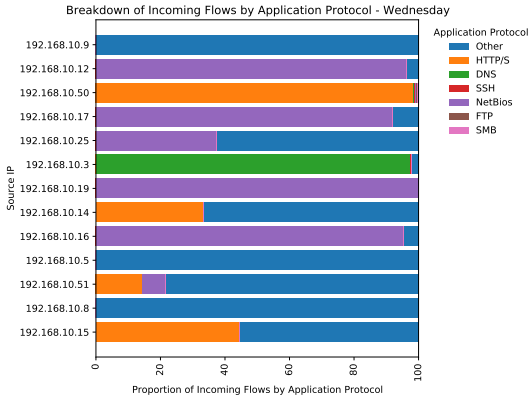


Figure 3.8: Proportion of incoming flows by application protocol for Wednesday

Again observe that 192.168.10.3 is the only host with predominantly incoming DNS traffic. An interesting set of hosts is 192.168.10.{12,17,19,16} which all exhibit high proportions of incoming NetBios or SMB traffic. These four hosts are the only Ubuntu machines in the network suggesting this pattern fingerprints this platform.

Another observation is the quantity of incoming HTTPS traffic to 192.168.10.{5,8,14,15}. On further analysis these flows originated from external IP addresses such as 72.217.12.147 and 185.49.84.72 which according to `who.is` records obtained July 2021 originate from Iran and China. Although these flows are marked benign, potentially this is incorrect as these hosts may have been probed by external attackers!

### 3.5.9 Metric 5: Number of Outgoing SSH Flows Per Host Per Hour

This metric measures the amount of outgoing SSH traffic from each internal source address per hour, see figures 3.9 and 3.10.

First observe there is a lot of SSH traffic with some hosts such as 192 168.10.9 having over 20 outgoing SSH flows in a single hour! In each case the SSH connections are being made to the web server 192.168.10.50. These flows look unrealistic as there are too many to have been initiated manually by a single user. Possible explanations include other applications regularly connecting to the server, or the beta profiles used to generate the data are faulty. While the hosts which have SSH traffic have similar

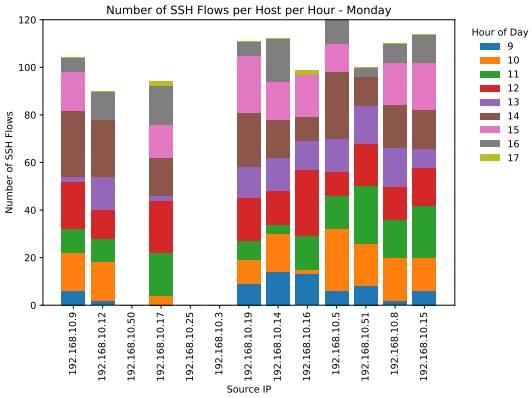


Figure 3.9: Number of SSH flows per host per hour for Monday

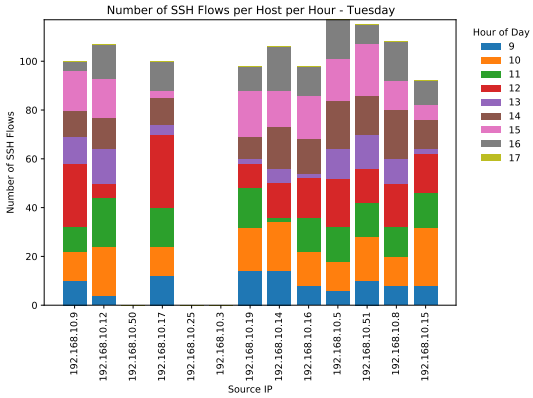


Figure 3.10: Number of SSH flows per host per hour for Tuesday

profiles, of greater interest are the hosts which have no outgoing SSH traffic at all namely 192.168.10.{50,25,3} which are the web server, Mac, and domain controller. Combined with the other metrics a process of elimination would be able to identify the Mac as it's the only non-server that has no outgoing SSH traffic.

### 3.5.10 Metric 6: Number of Outgoing NTP Flows per Host per Hour

This metric computes the number of outgoing Network Time Protocol flows from each internal source address per hour, see figures 3.11 and 3.11.

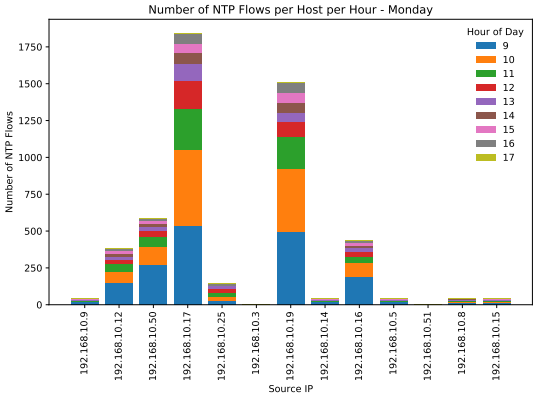


Figure 3.11: Number of NTP flows per host per hour for Monday

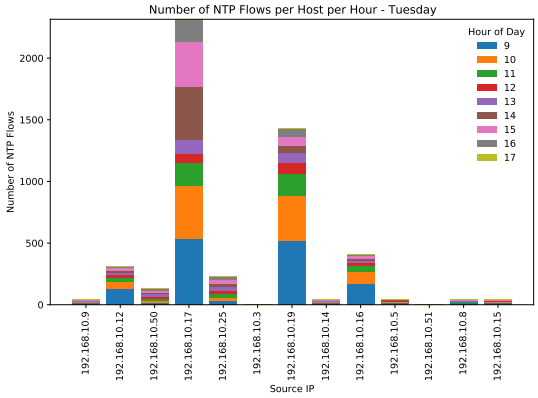


Figure 3.12: Number of NTP flows per host per hour for Tuesday

Observe that six hosts have significantly higher amounts of outgoing NTP traffic where the two largest peaks for hosts 192.168.10.{17,19} are both Ubuntu 14.4, the next two peaks at 192.168.10.{12,16} are Ubuntu 16.4, the peak 192.168.10.25 is

the Mac, and the final peak at 192.168.10.50 is the Ubuntu web server of unspecified version [20]. Thus all the hosts with significant NTP traffic are Unix based and furthermore the volume of NTP traffic discriminates between version of Ubuntu. It would be interesting to investigate further why these platforms have different NTP traffic profiles and whether the patterns hold for the most recent versions.

## 3.6 Metrics on PCAP Data

PCAP files contain an entire record of traffic captured within a network and consequently are a much richer source of data compared with network flows. The OSI model breaks internet communications down into seven layers, each with multiple protocols for different purposes. To be anonymized effectively each protocol should be analysed to identify where sensitive information might reside.

Using the PCAP data from Tuesday there are a total of 11,551,954 packets transmitted using 117 different protocols. Evaluating all of these protocols would be an enormous task that is out of scope of this project. Instead the objective of looking at PCAPs is to identify PII and other sensitive information that, due to the highly summarized form, is not present in the flow data. However since protocols such as HTTPS and FTPS are encrypted this prevents analysis of the headers and payloads and are typically ignored by intrusion detection systems. Thus to simplify the task only HTTP traffic is analysed.

### 3.6.1 Metric 7: URLs and Search Queries

An HTTP request performs an action such as retrieving a web page on a host with a specified Path. In a web browser this information is encoded in the Uniform Resource Locator (URL). However URLs can also contain additional data to be passed to the remote host which could be sensitive, for example when performing a Google search the query is encoded in the URL as a query parameter which is unencrypted can be easily extracted.

In the Tuesday PCAP data there are 8696 HTTP requests to 4925 distinct URLs and 776 distinct hosts. Table 3.3 displays the counts of the top 10 host names. The most frequent hostnames are advertising platforms (adnxs), a shopping center (interpark), and Google. The fact that `google.ca` appears high in the list suggests based on the top level domain that the user performing the search is based in Canada.

| Hostname | Frequency |
|---|---|
| nym1-ib.adnxs.com | 349 |
| www.interpark.com | 112 |
| www.google.com | 110 |
| goodhousekeeping-com.c.richmetrics.com | 86 |
| www.google.ca | 83 |

Table 3.3: Selected top hostnames from the Tuesday dataset

To analyse Google search queries the path is extracted from each HTTP request. Google search URLs have a query parameter `&q=` which contains the search query in plaintext. If the path matches this form the query is parsed and decoded to reinsert special characters such as spaces and quotes. A selection of the queries are listed in Table 3.4. Search queries reveal a lot about a user and in this scenario we can infer someone is attempting to install software on their Mac and is interested in events in Montreal, Canada.

| | |
|---|---|
| ”install failed” mac os x nmap | mac os x |
| copy files meterpreter | install xcode |
| install homebrew mac 10.5.8 7 | nmap source |
| install nmpa mac os | homebrew installer |
| montreal canada day billion dollar | port scan |

Table 3.4: Google search queries extracted

### 3.6.2 Metric 8: Email Addresses

To discover email addresses the payloads of each HTTP request are parsed using a regular expression. While many of the addresses from GET requests may simply be from public web pages, and hence not PII, addresses captured from POST requests, such as emails entered in a web form to register or login to a service, are likely to be more sensitive. In total 53 distinct email addresses were discovered, one of the more revealing ones being `cic.email01@gmail.com` presumably a Canadian Institute for Cyber Security address. In case the other addresses are sensitive they will not be reproduced here, lest to say there appeared to be several personal gmail addresses of the form `firstname.lastname@gmail.com`.

# Chapter 4

# Design and Implementation of Anonymization Policies

## 4.1 Anonymizing Network Flow Data

As explored in the network flow metrics (section 3.4) there are several privacy sensitive attributes that allow inference of sensitive information such as identifying the network behaviour of individual users, applications, and operating systems and versions running on each host.

Each flow has over 80 feature attributes describing the network activity and analysing all of them would be a massive task that is out of scope. Almost all of the sensitive information is founded on knowing the source IPs, destination IPs, and timestamps. To narrow the scope of the task these are the only fields considered for anonymization, however successfully anonymizing them will not guarantee the entire dataset is free of the sensitive information identified. This will be evaluated in section 5.

To anonymize these fields three anonymization polices have been designed. The following sections contain mini design specifications outlining the motivation, implementation, and method of evaluation for each policy.

### 4.1.1 Methodology of Baseline Privacy Measures

To analyze the effectiveness of the anonymization policies at preventing the leak of sensitive data, the amount of sensitive information in the dataset and the ability to make inferences will be measured using a combination of $k$-anonymity, $l$-diversity, and the metrics from section 3.5. To detect whether the policy has had any meaningful

effect the metrics will be recomputed on the anonymized dataset and compared with the baseline metrics from the unanonymized dataset. If the metrics change in a way that hides the sensitive statistics this does not prove that the anonymization technique is effective, as it may be possible to infer sensitive information via some other calculation. However they are useful to get a visual indication of how the dataset has been transformed by the anonymization policy, potentially leading to further improvements. Conversely if the metrics don't change in any meaningful way this demonstrates that the anonymization policy is not effective at removing sensitive information.

To measure the effectiveness of given technique at preventing inference, $k$-anonymity will be evaluated on a subset of the attributes. Recall that $k$-anonymization holds if for all equivalence classes of records with equal quasi-identifiers there are at least $k$ different identifiers represented. To make the analysis simpler the data is restricted to the following attributes: *Source IP, Destination IP, Destination Port, Timestamp_Hour*, where *Timestamp_Hour* is the hour component of the timestamp column. The source ports are not included as are chosen at random by the host for each flow so are not considered privacy sensitive. The source IP is classified as an identifier and the remaining attributes quasi-identifiers. To compute the $k$-anonymity value the data is grouped by the quasi-identifiers in equivalence classes and the number of unique source IPs in each equivalence class calculated. The mean, median, and mode of the number of source IPs in each class is then reported to estimate the value of $k$.

### 4.1.2 Methodology of Baseline Utility Measures

To measure the utility of an anonymization policy the task of anomaly detection is used. One potential method is to train a classifier to distinguish between all the different types of attack. However as shown in table 4.1 there is a big class imbalance in the number of flows of each type. Some attacks such as port scanning and DoS (denial of service) initiate lots of connections to hosts in order to detect active ports or overload the machine. Just because these 'noisy' attacks are highly represented in the flow data doesn't mean they are more likely, and training a model on the data as is would lead to a strong bias towards these types of attack.

Since the focus of the project is on anonymization rather than developing a good IDS, in order to get a simple utility metric a binary classifier is built to distinguish between benign and malicious traffic. All of the attack flows, including those classified as *Attempted*, are merged together into one *Malicious* class containing 442,466 flows.

Before training the dataset must be balanced to eliminate bias. Since there are a sufficiently large number of malicious flows the benign data is randomly undersampled to equalize the class sizes. The data was then split into a training and test set in a 9:1 ratio using stratified sampling to preserve the distribution of flow classes in each set, i.e. the training set and test set both have around 50% benign and malicious flows.

The selected model is a Random Forest classifier from the sci-kit learn Python library. This was chosen for two reasons. First it is the same model used by Sharafaldin et. al. in their paper on the CICIDS17 dataset [20], and by Engelen et al. in their paper on the corrected version [9], making it possible to draw comparisons with their experiments. Second it is easy to understand how the Random Forest makes its classifications by querying the importance weights for each feature in the dataset which will aid the analysis and evaluation of each anonymization policy.

| Label | No. of Flows | Label | No. of Flows |
|---|---|---|---|
| Benign | 1,654,875 | FTP-Patator | 3,974 |
| PortScan | 158,960 | SSH-Patator | 2,985 |
| DoS Hulk | 158,773 | Bot | 2204 |
| DDoS | 95,027 | Web Attack - Brute Force | 1,365 |
| DoS GoldenEye | 7,641 | Web Attack - XSS | 677 |
| DoS slowloris | 5,689 | Infiltration | 48 |
| DoS Slowhttptest | 5,100 | Web Attack - Sql Injection | 12 |
| Heartbleed | 11 | | |

Table 4.1: Number of flows of each type

### 4.1.3 Policy 1: Permuting IP Addresses

#### 4.1.3.1 Motivation

The source and destination IP addresses identify individual hosts and as shown in the metrics, section 3.5, can be grouped together to generate profiles of user behaviour. Additionally they can reveal personally sensitive information such web browsing history as the source IP identifies the user and the destination IP identifies the website. Thus a necessary condition to disassociate each flow from individual users is to anonymize the source IP addresses.

As described in the threat model, Network Forensics need to be able to report back

to Software Enterprises any malicious communications, so Software Enterprises needs to be able to identify which machines were involved. Simply deleting or randomly substituting IP addresses would not allow the data to be easily deanonymized for this purpose. Permuting IPs is an invertible operation that makes it easy for the party anonymizing the data to deidentify it while making it infeasible for an adversary to deanonymize by brute force as there are over 4 billion IP addresses, hence 4 billion factorial permutations.

#### 4.1.3.2 Implementation

There are numerous ways to anonymize IP addresses but the state of the art approach, as described in the background section 2 is prefix-preserving anonymization, where addresses that share a common prefix are anonymized to addresses that also share a common prefix of the same length, preserving the network structure. To implement this the Crypto-PAn [10] anonymization algorithm is selected due to its popularity and fast performance. The algorithm is provided with a key that is kept secret by the anonymizer and can be used to deanonymize the data. An off the shelf implementation of Crypto-PAn for Python pycryptopan is used to anonymize the source and destination IP addresses of each flow [17].

#### 4.1.3.3 Evaluation Methodology

To evaluate privacy first the metrics from section 3.5 will be recomputed and compared with the baseline to observe if they change. To evaluate inference, *k*-anonymity will be computed and compared with the baseline. To evaluate utility the binary classifier will be retrained on the anonymized dataset and the performance on the test set will be compared with the baseline performance on the unanonymized dataset.

### 4.1.4 Policy 2: Virtual NAT

#### 4.1.4.1 Motivation

While permuting IP addresses obfuscates the originals it fails to obfuscate patterns of traffic making it easy to reidentify hosts with known or distinctive traffic patterns. If an internet user wishes to hide their activities they don't simply select a random IP address for the same reason. One of the most effective ways to be anonymous is to hide in a crowd. On the internet this is implemented by routing a large number of

individuals' traffic through a single server so that an IP address represents multiple users, typically implemented using a VPN or a proxy. A similar idea, although not designed for the purposes of anonymity, is implemented at the transport layer of the internet where several machines in a local network reside behind a single public IP address. From the public facing network only the public IP address is known and the router sitting between the local and public network uses Network Address Translation (NAT) to route traffic across the interface. This motivates a way to anonymize groups of sensitive source and destination IP addresses by rewriting the flows such that they all appear to come from a much smaller number, potentially just a single, address. This idea is called Virtual NAT.

#### 4.1.4.2 Implementation

The flow dataset contains 14 internal IP addresses on the 192.168.0.0/16 subnet which constitute the vast majority of source IPs. Two thirds of the flows are between internal machines, the remaining third being between internal and public addresses. A naive implementation would be to replace all the internal source IPs with a single internal IP. However for internal to internal traffic this would result in loopback flows between the same machine which would never appear on the network, and so wouldn't appear in the dataset. Instead a pair of internal IP addresses are chosen and when anonymizing a flow the source IP is randomly selected from the pair and the destination IP, if also internal, is replaced with the other IP in the pair. Public IP addresses are not transformed to allow identification of connections to malicious external endpoints, for example using a blocklist of known bad public IPs. This results in a stream of flows between two internal machines and from internal to external ones, collapsing the topology of the internal network.

Since NAT uses port numbers to map traffic to hosts this anonymization process is invertible using a mapping table (essentially the NAT table) which records the source and destination IP, ports, and timestamps for each transformed flow. When transforming the flows it's important to ensure that the source and destination ports are not being used at the same time as a currently active flow, as would be enforced by a real NAT device. Since there are 14 internal machines and 65,535 port numbers, of which a subset are available for NAT, mapping the traffic between two machines is sufficient. However for a larger network with higher volumes of traffic there is a risk of running out of port numbers. This limitation was considered out of scope, but a potential solution would be to add more internal IPs.

#### 4.1.4.3 Evaluation Methodology

Privacy and utility will be evaluated in the same manner as Policy 1.

### 4.1.5 Policy 3: Flow Diffusion

#### 4.1.5.1 Motivation

Although an algorithm like Virtual NAT is able to hide individual users' traffic by clumping them all together there is still the possibility of identifying unique traffic patterns of individual users. Using the analogy of gaining privacy by hiding in a crowd, it's important to hide in a crowd of people who are similar to you. If you're wearing a red shirt, you want to hide in a crowd of people wearing red shirts or a mixture of colours, not a crowd of people wearing white shirts. If Bob is the only person to start work at 7AM, then his web browsing history at this time could be easily identified using timestamps. For example, observe in Figure 5.2 that there is still a distinctive drop in traffic during hour 12, indicating that network activity drops at lunch time. This motivates modifying the timestamps to destroy this type of inference.

#### 4.1.5.2 Implementation

There are various possibilities for modifying timestamps including time unit annihilation, where the least significant portions such as the seconds or minutes of timestamps are set to zero, and enumeration where timestamps are replaced with sequential integers incrementing from the earliest to the latest flow [1, p.g. 22]. Since each flow is an independent communication between two hosts another possibility, which in this thesis is named time diffusion, is to randomize the timestamps such that the flows are uniformly distributed. To diffuse the dataset the timestamp of each flow is set to a uniformly random time between 9AM and 5PM, the same time span of the original flows.

Although not implemented in this policy the flow diffusion could be made an invertible operation by keeping a map of the original and new timestamps of each flow that is kept secret by the anonymizer.

#### 4.1.5.3 Evaluation Methodology

To measure privacy first the metrics and $k$-anonymity will be measured and compared with the baseline. To measure the effectiveness of timestamp diffusion on privacy

the $l$-diversity measure will be used. Using the same methodology for computing $k$-anonymity the flows will be grouped into equivalence classes by destination port and IP and the number of unique timestamp hours in each group computed. The mean, median, and mode will be calculated to give an estimate of the value of $l$. The effectiveness will then be evaluated by comparing the $l$-diversity of the timestamps between the anonymized and unanonymized datasets.

Utility will be evaluated using the same methodology as in Policy 1.

## 4.2 Anonymizing PCAP Data

### 4.2.1 Methodology of Privacy Measures

In a similar fashion to the flow data, the metrics computed in section 3.5 can be compared between the anonymized and unanonymized versions as a first test to see if the dataset has changed. However unlike the flow data it's not immediately clear how to apply algorithms such as $k$-anonymity and $l$-diversity due to the many layers of communications protocols. One approach is to compute the flow data from the PCAPs and then compute the privacy statistics, however computing the flows summarizes and discards lots of the data. Just because the privacy measures hold on the derived flow data doesn't imply the sensitive data cannot be inferred from the PCAP.

Due to time limitations a suitable approach to measure privacy along the ideas of $k$-anonymity for PCAP data was not discovered or devised. In any case the task is complex enough to be a project in its own right. Thus the evaluation of these policies will primarily focus on utility for which there are more convincing measures.

### 4.2.2 Methodology of Utility Measures

To evaluate utility Suricata, an open source rule based IDS, is used [25]. Suricata has a list of built in rules to detect malicious network activity and outputs a report containing alerts with priority 1-3 where 1 is the highest priority.

For each day in the dataset there is a single PCAP containing the traffic sent and received between all machines in the network. However each file is around 10GB in size which is difficult to manipulate. To make the experiments easier a subset of the data between 14:00 and 15:00 in the Tuesday dataset, is split out and used for all the experiments. The PCAP was split using the *editcap* tool which comes with WireShark. This hour of data contains a mixture of benign traffic of a variety of protocols and also

malicious SSH traffic. However the focus isn't on getting Suricata to detect these attacks, rather on investigating whether and how the alerts change after an anonymization algorithm is applied.

### 4.2.3 Policy 4: Removing URLs and Email Addresses

#### 4.2.3.1 Motivation

In section 3.6.1 it was shown that the HTTP Path header can contain sensitive information in the query string, e.g. Google search queries, such that even if the source IP were removed it could still be traced back to an individual. Similarly in section 3.6.2 the payload may also contain sensitive information such as email addresses. Such data is not needed to detect malicious network activity motivating its removal from the dataset.

#### 4.2.3.2 Implementation

The Path HTTP header field typically looks like `/path/to/page?key=value` where the query string, is the portion appearing after the question mark. To remove the query strings the Path fields of every HTTP packet are modified using a regular expression to extract and keep the part before the query string, thus keeping the portion `/path/to/page`. The packets are modified using the Python library Scapy [19].

To anonymize the email addresses the payload of each HTML packet is edited using a regular expression to match and delete them.

#### 4.2.3.3 Evaluation Methodology

The same procedure used to compute the URL and email metrics is run on the PCAP files to verify they have all been removed. Utility of the dataset is measured by comparing the number of Suricata alerts of each priority level with the baseline.

### 4.2.4 Policy 5: Virtual NAT for PCAPs

#### 4.2.4.1 Motivation

The same issues with traffic identification for flow data identified in section 4.1.4 apply to PCAPs too.

#### 4.2.4.2 Implementation

The concept of Virtual NAT was explained in section 4.1.4 and the same algorithm is used to anonymize traffic from internal addresses. To ensure consistent IP addresses are used the PCAP was split into flows using SplitCap [24]. The packets in each session were then anonymized by changing the IP addresses and ports as per virtual NAT using the Scapy Python library. After anonymization the flow PCAPs were merged back into one file using the mergecap tool from Wireshark [14].

#### 4.2.4.3 Evaluation Methodology

As with Policy 4, utility will be evaluated using Suricata.

### 4.2.5 Policy 6: Flow Diffusion for PCAPs

#### 4.2.5.1 Motivation

The same issues with extracting traffic patterns based on timestamps with flow data identified in section 4.1.5 apply to PCAPs too.

#### 4.2.5.2 Implementation

As with Policy 5, the packets in the PCAP file are split into flows or sessions. For each session a uniformly random timestamp between 14:00 and 15:00, the original time span of the PCAP, is selected. Each packet within the session is assigned a randomly chosen timestamp with a 1 millisecond offset for each subsequent packet to simulate the delay between packets. The modified sessions are then merged back together.

#### 4.2.5.3 Evaluation Methodology

As with Policy 4, utility will be evaluated using Suricata.

# Chapter 5

# Analysis and Evaluation

## 5.1 Analysis of Anonymized Network Flow Data

Table 5.1 summarizes the results of the experiments on the flow data.

| Policy | $k$-anonymization | | | Time_Hour $l$-diversity | | | Classifier |
|---|---|---|---|---|---|---|---|
| | Mean | Median | Mode | Mean | Median | Mode | Accuracy |
| No Anonymization | 1.4 | 1 | 1 | 2.0 | 1 | 1 | 99% |
| IP Permutation | 1.4 | 1 | 1 | 2.0 | 1 | 1 | 99% |
| Virtual NAT | 1.6 | 2 | 2 | 2.0 | 1 | 1 | 99% |
| Flow Diffusion | 1.4 | 1 | 1 | 3.6 | 3 | 1 | 99% |

Table 5.1: Summary of results for policy evaluations on the flow data

### 5.1.1 Analysis of Baseline Privacy Measures

On the unanonymized data the $k$-anonymization measure for the number of unique source IPs had a mean, median, and mode of 1.4, 1, and 1 respectively. This shows that $k$-anonymity is only satisfied with $k = 1$, the worst possible case which means the quasi-identifiers (destination IP, port and timestamp hour) almost perfectly infer the identifier and hence these fields must be anonymized to prevent inference.

### 5.1.2 Analysis of Baseline Utility Measures

When the Random Forest classifier was evaluated on the test set it achieved 99% accuracy. Having such high accuracy was a cause for alarm, however it is in line with

experiments using the same classifier by Engelen et. al. [9]. Having a high accuracy is potentially due to CICIDS17 being a synthetic dataset where due to less variation in the benign traffic than would occur in the real world, the malicious traffic stands out more distinctly. Since an anonymization algorithm is expected to have a negative impact on utility, because the information in the dataset is degraded, starting from such high accuracy does not immediately invalidate the experiments.

However, as can be seen in table 5.1 throughout all of the experiments the accuracy remained static at 99% on the test set. This raises questions about the quality and usefulness of this test. How must the data be changed in order for the accuracy of the test to decrease? To help answer this the Random Forest classifier can calculate importance weights for each feature. The top five important features for the flow dataset are displayed in Table 5.2.

| Feature | Importance | Accuracy with Features of Equal or Greater Importance |
|---|---|---|
| RST Flag Count | 1 | 92% |
| ACK Flag Count | 2 | 95% |
| Bwd Segment Size Avg | 3 | 98% |
| FWD Init Win Bytes | 4 | 99% |
| Idle Mean | 5 | 99% |

Table 5.2: Top 5 important features and test set accuracy for the binary classifier

When just training on just the five most important features the accuracy on the test set was still 99%, with just the top 3 features 98%, top 2 features 95%, and with just the *RST Flag Count* the classifier has an accuracy of 92% on the test set. This indicates that the task of classifying the malicious traffic is too trivial for this dataset.

While there exist more sophisticated models for intrusion detection such as [4] which uses LSTM models, there wasn't enough time during the project to get them working and evaluate them. Instead more attention was spent on anonymizing the PCAP data for which there are a wider range of intrusion detection tools that can provide convincing measures of utility.

### 5.1.3 Analysis of Policy 1: Permuting IP Addresses

Crypto-PAn was applied to all the source and destination IP addresses and the metrics recomputed. In each case the results were identical graphs where the source IPs for

each machine were simply relabeled, see Figure 5.1 for an example.

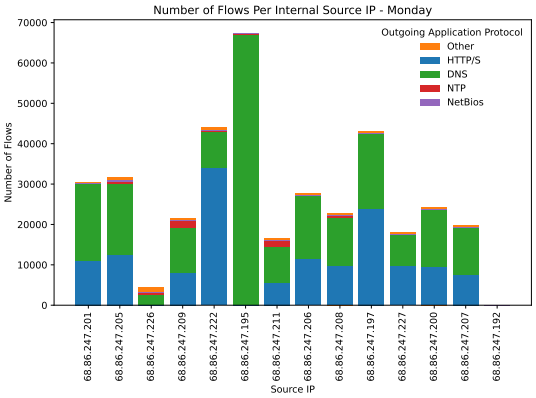


Figure 5.1: Monday flow breakdown after Crypto-PAn anonymization

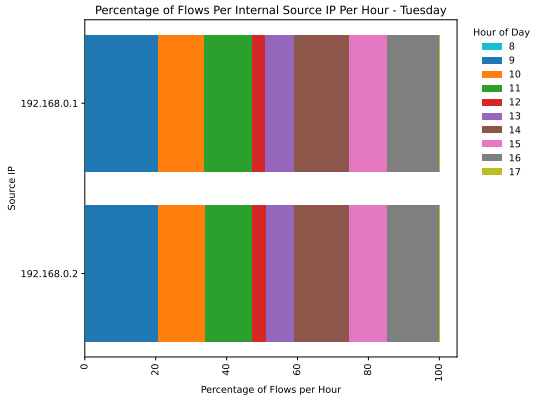


Figure 5.2: Flows per internal IP per hour for Tuesday after applying Virtual NAT

When measuring inference with $k$-anonymity the results were also identical to the baseline. This is unsurprising given the entropy of the data hasn't changed, and the same would be observed for any anonymization function that permutes IP addresses.

In terms of utility The performance of the binary classifier was also unchanged with 99% accuracy on the test set. This is also unsurprising as it shouldn't matter which two hosts a malicious flow occurs between in order to detect that it is malicious.

Overall Crypto-PAn provides a poor level of anonymization. While it's not easy to derive the original IP addresses just from the anonymized ones, it's still very easy to identify hosts by profiling the traffic. Indeed if the profiles of particular users are known, for example there's a certain employee at Software Forensics who starts work at the same time every day, then Bob would be able to identify them and once the anonymized IP is known then this deidentifies all other traffic from that user. Similarly Frank would be able to identify the purpose of different machines in the network, for instance he would be able to infer there was likely a DNS server accessible from all the machines in the network which he could try to compromise in a cyber attack.

### 5.1.4 Analysis of Policy 2: Virtual NAT

After applying virtual NAT the metrics changed such that all the flows are clumped together as highlighted in Figure 5.2. After recomputing $k$-anonymity the mean, median, and mode are 1.6, 2, and 2. This suggests that most of the data now satisfied 2-anonymity, where each set of flows that agree on destination IP, destination port, and

timestamp hour have at least one flow originating from both internal source IPs. Since there are only two source IPs this is the best value of $k$ that can be achieved.

In terms of utility the performance of the Random Forest classifier was unchanged with 99% accuracy on the test set. This is unsurprising as the classification of whether a flow is malicious or not does not depend on the machines on which it occurred.

Overall Virtual NAT improves privacy by clumping flows together behind a small number of IP addresses and reduces the ability to derive profiles of users or reveal the network topology. Looking back at the threat model this would hamper Bob and Catherine from identifying their colleagues' network behaviour as they would be able to easily infer which colleague each flow was associated with. However there is still an issue where traffic occurring at a certain time of day could still be associated with a user. For example if Bob knew that his colleague started work before everyone else, he may be able to guess that traffic at that time came from that user. Frank and Eliza would also be hampered as they wouldn't be able to easily infer the network topology as the internal network has been collapsed to two machines.

Another issue with virtual NAT is making the algorithm work when there are very high levels of traffic as the number of available ports may run out. Ensuring that the traffic looks realistic after the transformation is a complex task, and it's possible that inconsistencies, errors, or unusual remappings of traffic could be leveraged to undo the anonymization, however this would require further research to understand whether this is a credible threat.

### 5.1.5 Analysis of Policy 3: Flow Diffusion

When flow diffusion was applied to the original dataset the metrics changed to show a uniform distribution of flows across the day as illustrated in Figure 5.3. After recomputing the $k$-anonymity metrics the mean, median, and mode were unchanged from the baseline.

Of particular interest for this policy are the $l$-diversity values for the timestamp hour. Recall on the original dataset the mean, median, and mode were 2.0, 1, and 1 respectively. On the anonmyized dataset the values were 3.6, 3, and 1. This indicates that the value of $l$ on the anonymized dataset is around three which means in each equivalence class of flows with the same destination IP and port there are flows occurring in at least three different hours of the day. Since the dataset contained 8 hours of data this shows that the flows occur in just under half the available hours, much more

diverse than in the original dataset. The fact the mode hasn't changed suggests there are many one-off communications to a particular host and port which would remain unchanged by flow diffusion.

In terms of utility the performance of the classifier was unchanged remaining static at 99% accuracy on the test set. One explanation is that the time of day of a flow may not be a strong feature in identifying it as benign or malicious as cyber attacks can occur at any time of day.

Overall flow diffusion does appear help anonymize the dataset, making it harder to associate traffic with users based on the time of day, however the metrics computed here don't constitute a proof that such information could not be recovered by other means. A further interesting experiment is to combine Virtual NAT with flow diffusion to get the best of both policies. This smooths out the distribution of flows between the two machines, compare Figure 5.2 with Figure 5.4.

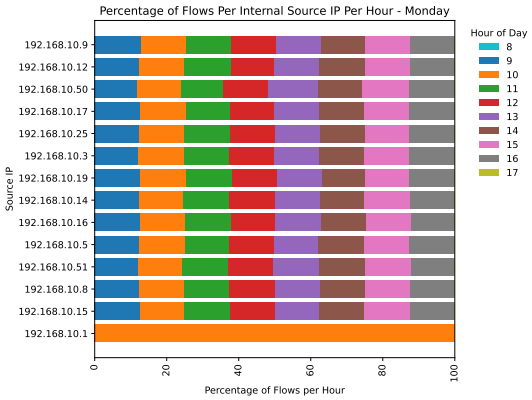


Figure 5.3: Flows per internal IP per hour for Monday after applying diffusion

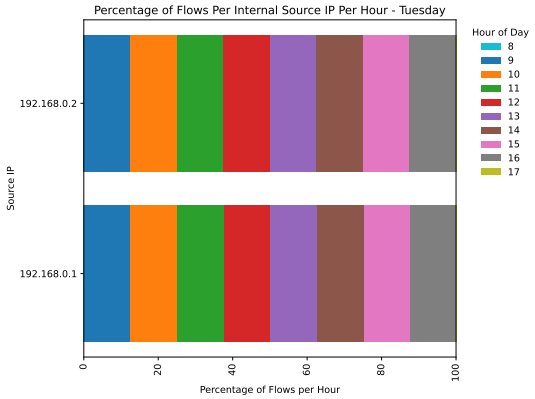


Figure 5.4: Flows per internal IP per hour for Tuesday after applying Virtual NAT and diffusion

## 5.2 Analysis of Anonymizing PCAP Data

Table 5.3 summarizes the results of the experiments on the PCAP data.

As seen in the table many priority-3 alerts are triggered as a side effect of modifying the packets for each anonymization policy. Upon investigation these were found to be primarily alerts about invalid checksums and mismatched HTTP requests and responses. While these problems could be addressed instead, due to time limitations with the project, only priority-1 and priority-2 alerts are compared in the analysis.

| Policy | No. Suricata Alerts | | |
|---|---|---|---|
| | Priority-1 | Priority-2 | Priority-3 |
| No Anonymization | 7 | 29 | 391 |
| Remove URLs & Emails | 7 | 29 | 1,682 |
| Virtual NAT | 5 | 23 | 18,950 |
| Flow Diffusion | 7 | 29 | 674 |

Table 5.3: Summary of results for policy evaluations on the PCAP data

### 5.2.1 Analysis of Policy 4: Removing URLs and Email Addresses

Using the metrics as validation the anonymization procedure successfully removed all the query strings from the HTTP paths and email addresses from the HTTP payloads.

In terms of utility when Suricata was run on the anonymized PCAP identical priority-1 and priority-2 alerts were generated, suggesting no drop in IDS performance. This likely means that the query strings and email addresses are not being used for detecting malicious behaviour. However to know whether this is the analysis should be repeated on a much more comprehensive PCAP file with more types of threat. If the results hold, then for privacy this is a good thing as it means this sensitive data can be deleted from the dataset and avoid any risk of disclosure.

Linking back to the threat model, this would make it difficult if not impossible for Bob or Catherine to determine the search queries their colleagues made. Removing email addresses hampers Eliza from discovering who Software Enterprises is in communication with. It also means Alice doesn't have an easy success for her research as there is less PII in the dataset to uncover.

### 5.2.2 Analysis of Policy 5: Virtual NAT for PCAP

After virtual NAT was applied the flow data was regenerated using the CICFlowMeter tool to verify the IP addresses in the flows had the same properties as when it was applied directly to the flow data in section 4.1.4.

In terms of utility Suricata produced fewer alerts with 5 priority-1 alerts and 23 priority-2 alerts. The differences between the alerts in the baseline and anonymized are summarized in Table 5.4. It's not clear why changing the internal IP addresses would reduce the number of TLS handshake failure alerts as TLS errors are explicit messages such as failure to agree on a cipher or unable to verify a certificate. Further

investigation would be needed to understand the reasons for it. This suggests that modifying the IP addresses using virtual NAT affects the datasets utility in subtle ways, despite most of the alerts being triggered as before.

| Alert | Baseline Count | Virtual NAT Count |
|---|---|---|
| INFO TLS Handshake Failure | 8 | 1 |
| HUNTING Observed Interesting Content-Type Inbound (application/x-sh) | 1 | 0 |
| POLICY HTTP POST invalid method case outbound | 0 | 1 |

Table 5.4: Difference between baseline and Virtual NAT Suricata alerts

In theory virtual NAT would provide similar privacy enhancements as it does with the flow data (section 5.1.4, as there would only be two internal machines in the network. However due to the complex nature of PCAP data further analysis is required to demonstrate whether this is the case.

### 5.2.3 Analysis of Policy 6: Flow Diffusion for PCAP

After flow diffusion was applied the flow data was regenerated using the CICFlowMeter tools to verify the timestamps in the flows had the same properties as if it had been directly applied to the flow data in section 4.1.5.

In terms of utility Suricata produced the same number of priority-1 and priority-2 alerts as the baseline (table 5.3. Similar to the analysis when removing URLs and email addresses this experiment should be repeated on a much larger and comprehensive PCAP file to verify these results hold more generally. If they do then this indicates that the timestamp isn't being used by Suricata to detect malicious traffic.

Similarly to virtual NAT, in theory flow diffusion provides similar privacy enhancements as it does to flow data, however further analysis is required to verify this.

## 5.3 Evaluation: Can network traffic be anonymized?

The requirements of the project, section 3.1, listed four objectives. The first sought to investigate what sensitive information exists in a network dataset. While the metrics computed in section 3.5 present some sensitive data they are by no means exhaustive.

Additionally due to the dataset being synthetic it is likely that a real world dataset contains much more sensitive information that require new metrics to detect it.

The second objective sought to devise anonymization methods to protect the identified sensitive information. While techniques such as virtual NAT and flow diffusion were developed, there is a much wider design space of anonymization methods that could be more suitable. Combined with the size and complexity of network datasets, the small number of methods considered in this thesis are best viewed as a starting point to developing more sophisticated anonymization policies. As van Dijkhuizen et. al. stated, an analysis of field sensitivity of all protocols is required [8].

The third objective sought to develop measures of whether a given anonymization policy improves privacy. With the flow data, existing privacy metrics such as $k$-anonymity and $l$-diversity could be easily calculated. However it's not always obvious what values of $k$ and $l$ are suitable, and satisfying the conditions are not proof of privacy. While the anonymization policies applied to the flow data generally saw improvement of these metrics, there isn't enough information to say with confidence that they guarantee privacy. With the PCAP data it wasn't possible to apply these measures in the same manner because the data is not in a suitable form as each packet may have a completely different structure and sensitive fields. It proved to be challenging to find or devise suitable privacy metrics to measure the anonymization policies against, and in the end this task was left as an open problem.

The final objective sought to develop measures of utility of the anonymization policies for network intrusion detection. The classification model developed for the flow data proved to be too simplistic as the investigation concluded, the accuracy largely depended on a single attribute of the flows which wasn't anonymized. Due to time constraints it wasn't possible to investigate more sophisticated models.

Using Suricata for the PCAP data was a more convincing method for evaluating utility, however modifying the PCAP files such that they didn't trigger false alarms proved to be challenging. PCAP data is very sensitive to modification, particularly due to the large number of checksums in each communication layer. The results of the experiments did not convincingly show that utility was hampered or preserved, with the main weakness being the relatively small amount of data the policies were evaluated against. The challenge is that even an hour of PCAP data is over 1GB in size and modifying all the packets to apply the anonymization is a computationally expensive process. With a condensed dataset where a small PCAP contains many examples of benign and malicious activity the analysis could be performed much more efficiently.

# Chapter 6

# Conclusion

When protecting sensitive information in a dataset the all too common approach is to naively apply popular anonymization algorithms and hope that it is sufficient. However such 'anonymized' datasets are often shown to be insecure and the data reidentified. The goal of this project was to investigate through a rigorous and scientific process whether it is possible to anonymize a network traffic dataset to protect sensitive information while maintaining its utility for network intrusion detection.

Anonymizing datasets, especially complex ones such as network traffic, is subtle and complex. It's imperative to know what sensitive data should be protected and who the data is to be protected from. This can be achieved by developing metrics to understand what sensitive information is in the dataset and devising a threat model to understand potential adversaries.

Once the sensitive data and threats have been identified, the process of devising anonymization algorithms and policies can begin. However, evaluating the efficacy of anonymization policies is also subtle and challenging. There are no one-size-fits-all measures of privacy and utility, and the ones that do exist, such as $k$-anonymity, are subjective and must be interpreted in the context of the dataset being anonymized.

Although the anonymization policies devised to protect the dataset in this project were not effective enough to mitigate the threat model, the primary contribution is developing and demonstrating a scientific methodology for devising and evaluating anonymization policies for network datasets. One of the greatest limitations is the dataset is not realistic enough. A real world dataset is likely to contain more sensitive information and require a thorough threat analysis. Repeating this work using a real world corporate dataset would undoubtedly shed new light on what policies, algorithms, and metrics are required to successfully anonymize network traffic datasets.

# Bibliography


[1] Martin Burkhart. *Enabling collaborative network security with privacy-preserving data aggregation*, volume 125. ETH Zurich, 2011.

[2] Martin Burkhart, Daniela Brauckhoff, and Martin May. On the utility of anonymized flow traces for anomaly detection, 2008.

[3] Cicflowmeter. `https://www.unb.ca/cic/research/applications.html#CICFlowMeter`. Accessed: 2021-07-30.

[4] Henry Clausen, Gudmund Grov, Marc Sabate, and David Aspinall. Better anomaly detection for access attacks using deep bidirectional lstms. In Éric Renault, Selma Boumerdassi, and Paul Mühlethaler, editors, *Machine Learning for Networking*, Lecture Notes in Computer Science, pages 1–18. Springer International Publishing, March 2021.

[5] Scott Coull, Fabian Monrose, Michael Reiter, and Michael Bailey. The challenges of effectively anonymizing network data. pages 230 – 236, 04 2009.

[6] Scott Coull, Charles Wright, Angelos Keromytis, Fabian Monrose, and Michael Reiter. Taming the devil: Techniques for evaluating anonymized network data. 01 2008.

[7] Scott E Coull, Charles V Wright, Fabian Monrose, Michael P Collins, Michael K Reiter, et al. Playing devil's advocate: Inferring sensitive information from anonymized network traces. In *Ndss*, volume 7, pages 35–47, 2007.

[8] Niels Van Dijkhuizen and Jeroen Van Der Ham. A survey of network traffic anonymisation techniques and implementations. *ACM Comput. Surv.*, 51(3), May 2018.

[9] Gints Engelen, Vera Rimmer, and Wouter Joosen. Troubleshooting an intrusion detection dataset: the cicids2017 case study. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 7–12, 2021.

[10] Jinliang Fan, Jun Xu, Mostafa H. Ammar, and Sue B. Moon. Prefix-preserving ip address anonymization: measurement-based security evaluation and a new cryptography-based scheme. *Computer Networks*, 46(2):253–272, 2004.

[11] Kiran Lakkaraju and Adam Slagell. Evaluating the utility of anonymized network traces for intrusion detection. In *Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks*, SecureComm '08, New York, NY, USA, 2008. Association for Computing Machinery.

[12] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115, 2007.

[13] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. L-diversity: privacy beyond k-anonymity. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 24–24, 2006.

[14] Mergecap. `https://www.wireshark.org/docs/man-pages/mergecap.html`. Accessed: 2021-07-30.

[15] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125, 2008.

[16] Ruoming Pang, Mark Allman, Vern Paxson, and Jason Lee. The devil and packet trace anonymization. *SIGCOMM Comput. Commun. Rev.*, 36(1):29–38, January 2006.

[17] pycryptopan. `https://pypi.org/project/pycryptopan/`. Accessed: 2021-07-30.

[18] Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. 1998.

[19] Scapy. `https://scapy.net/`. Accessed: 2021-07-30.

[20] Iman Sharafaldin, Arash Habibi Lashkari, and Ali Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. pages 108–116, 01 2018.

[21] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers and Security*, 31(3):357–374, 2012.

[22] Adam Slagell and William Yurcik. Sharing computer network logs for security and privacy: a motivation for new methodologies of anonymization. pages 80 – 89, 10 2005.

[23] Snort. `https://www.snort.org/`. Accessed: 2021-07-30.

[24] Splitcap. `https://www.netresec.com/?page=SplitCap`. Accessed: 2021-07-30.

[25] Suricata. `https://suricata.io/`. Accessed: 2021-07-30.

[26] Mi-Ja Woo, Jerome Reiter, Anna Oganian, and Alan Karr. Global measures of data utility for microdata masked for disclosure limitation. *Journal of Privacy and Confidentiality*, 1, 04 2009.

# Appendix A

# Fields in the ClCIDS17 Flow Dataset

The complete list of 84 fields (Flow ID plus 83 features) in the flow dataset are:

'Flow ID', 'Src IP', 'Src Port', 'Dst IP', 'Dst Port', 'Protocol',
'Timestamp', 'Flow Duration', 'Total Fwd Packet', 'Total Bwd packets',
'Total Length of Fwd Packet', 'Total Length of Bwd Packet',
'Fwd Packet Length Max', 'Fwd Packet Length Min',
'Fwd Packet Length Mean', 'Fwd Packet Length Std',
'Bwd Packet Length Max', 'Bwd Packet Length Min',
'Bwd Packet Length Mean', 'Bwd Packet Length Std', 'Flow Bytes/s',
'Flow Packets/s', 'Flow IAT Mean', 'Flow IAT Std', 'Flow IAT Max',
'Flow IAT Min', 'Fwd IAT Total', 'Fwd IAT Mean', 'Fwd IAT Std',
'Fwd IAT Max', 'Fwd IAT Min', 'Bwd IAT Total', 'Bwd IAT Mean',
'Bwd IAT Std', 'Bwd IAT Max', 'Bwd IAT Min', 'Fwd PSH Flags',
'Bwd PSH Flags', 'Fwd URG Flags', 'Bwd URG Flags', 'Fwd Header Length',
'Bwd Header Length', 'Fwd Packets/s', 'Bwd Packets/s',
'Packet Length Min', 'Packet Length Max', 'Packet Length Mean',
'Packet Length Std', 'Packet Length Variance', 'FIN Flag Count',
'SYN Flag Count', 'RST Flag Count', 'PSH Flag Count', 'ACK Flag Count',
'URG Flag Count', 'CWR Flag Count', 'ECE Flag Count', 'Down/Up Ratio',
'Average Packet Size', 'Fwd Segment Size Avg', 'Bwd Segment Size Avg',
'Fwd Bytes/Bulk Avg', 'Fwd Packet/Bulk Avg', 'Fwd Bulk Rate Avg',
'Bwd Bytes/Bulk Avg', 'Bwd Packet/Bulk Avg', 'Bwd Bulk Rate Avg',
'Subflow Fwd Packets', 'Subflow Fwd Bytes', 'Subflow Bwd Packets',
'Subflow Bwd Bytes', 'FWD Init Win Bytes', 'Bwd Init Win Bytes',
'Fwd Act Data Pkts', 'Fwd Seg Size Min', 'Active Mean', 'Active Std',

'Active Max', 'Active Min', 'Idle Mean', 'Idle Std', 'Idle Max',
'Idle Min', 'Label'.