Petter Ødegård

Data Anonymization for Research

Master's thesis in Communication Technology
Supervisor: Otto Jonassen Wittner
June 2019

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Information Security and Communication Technology

Petter Ødegård

Data Anonymization for Research

Master's thesis in Communication Technology
Supervisor: Otto Jonassen Wittner
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Information Security and Communication Technology

Abstract

The implementation of the General Data Protection Regulation (GDPR) causes companies to reconsider their approach for storing user data. Researchers are in need of realistic data for their analyses to provide the best results. While there are strict rules in the GDPR against sharing and storing personal data, there are exceptions that can be adopted. Anonymization with the possibility of returning to the original data is called pseudonymization. By pseudonymizing data destined for researchers in the desired way, compliance with the GDPR can still be achieved for companies.

This master thesis investigates which policies that need to be in place to comply with the GDPR. Research is devoted to finding pseudonymization methods that can process personal data fields in network traffic logs so that they are no longer identified as personal according to the GDPR. At the same time, the fields should be of value to researchers.

Multiple fields are found to contain personal data. In addition, there are fields that do not directly identify a person, but could be used in combination with other fields to single out a person. In this thesis an extensive validation process is performed to compare state-of-the-art pseudonymization techniques from a security perspective. The results show that there are measurable differences between techniques, and that some combinations work better than others. New novel pseudonymzation techniques are suggested and shown to improve the level of anonymity by 4-5%.

Sammendrag

Implementeringen av GDPR får bedrifter til å revurdere deres måte å lagre brukerdata. Forskere trenger realistiske data for å kunne oppnå best mulige resultater. Selv med strenge regler fra GDPR mot å dele og lagre personlig data finnes det unntak som kan tas i bruk. Anonymisering hvor man bevarer muligheten til å returnere til den originale dataen kalles pseudonymisering. Bedrifter kan være i samsvar med GDPR dersom data som skal gis til forskere pseudonymiseres på en ønskelig måte.

Denne masteroppgaven utforsker hvilke retningslinjer som må være til stede for å samsvare med GDPR. Forskningen går ut på å finne pseudonymiseringsmetoder som kan prosessere felt som inneholder personlig data i netverkstrafikklogger slik at de ikke lenger kan identifiseres som personlige, ifølge GDPR. Samtidig skal feltene fortsatt inneholde nyttig data for forskere.

Flere felt viser seg å inneholde personlig data. I tillegg finnes det felt som ikke direkte kan identifisere en person, men som kan kombineres med andre felt for å avsløre en person. I denne avhandlingen har en omfattende valideringsprosess blitt utført for å tillate sammenligning av moderne, aktuelle pseudonymiseringsteknikker fra et sikkerhetsperspektiv. Resultatene viser at det er målbare forskjeller mellom teknikkene, i tillegg til at noen kombinasjoner fungerer bedre enn andre. Nye pseudonymiseringsteknikker er foreslått og vist å forbedre anonymiteten med 4-5%.

Preface

This master thesis concludes the master program in Communication Technology at Norwegian University of Science and Technology (NTNU). The thesis was a suggested topic by Uninett, a Norwegian network provider, after realizing that a better understanding of the impact of the GDPR for their company was needed. As part of Uninett's openness policy, a desire is to share operational data from network and systems with researchers. The GDPR puts restrictions to this policy.

The master thesis is a two-part work: a pre-project in the autumn 2018, and the thesis work extended from the pre-project in the spring 2019. The initial approach from the autumn has been expanded by investigating fields from different logs and performing validation, in addition to the preexisting focus on the GDPR and anonymization techniques. Petter Ødegård, with supervision from Otto Jonassen Wittner at Uninett and the Department of Information Security and Communication Technology, NTNU, has conducted the research and provided the results for the thesis.

Acknowledgement

First of all, I would like to sincerely thank my supervisor, Otto Jonassen Wittner, for all the hours he has spent discussing, helping and guiding me towards my results and this report. With his knowledge of programming, logical thinking and patience, the thesis has developed from searching for interpretations of the GDPR, looking at poorly described, possible anonymization techniques and inadequate validation techniques, to a discussed proposal with valid results from a tested validation process. All the hours spent at Uninett and his constructive feedback have been invaluable to my motivation and ability to provide a final result in this master thesis. Our weekly meetings have really pushed me to always working, from start to finish, even with multiple obstacles along the way.

Secondly, I would like to thank Uninett for the possibility to work with such a relevant and interesting project, and allowing me to work with samples of their logs on their platform.

Contents

List of Figures **ix**

List of Tables **xi**

List of Acronyms **xiii**

1 Introduction **1**

1.1 Scope of the Thesis 1
1.2 Justification, Motivation and Benefits 2
1.3 Research Questions 3
1.4 Contributions 3
1.5 Thesis Outline 4

2 Background and Related Work **5**

2.1 The General Data Protection Regulation 5
2.1.1 Initial Idea 5
2.1.2 Implications for Research 6
2.2 Anonymization 6
2.3 Validation 8

3 Network Traffic Logs **9**

3.1 Internet/Transport Layer Log 9
3.1.1 IPv4 Header 10
3.1.2 IPv6 Header 12
3.1.3 TCP Header 13
3.1.4 UDP Header 15
3.2 NetFlow Log 15
3.3 Web Server Log 18
3.4 System Log 20
3.5 Summary 21

4 Anonymization **27**

4.1 Personal vs Sensitive Data 27

4.2 Anonymization vs Pseudonymization 28
4.3 Pseudonymization for Network Traffic Logs 29
4.4 Anonymization Techniques 31
4.4.1 Crypto-PAn 31
4.4.2 Tcpmkpub 31
4.4.3 PktAnon 32
4.4.4 AnonTool 33
4.4.5 SCRUB-tcpdump 34
4.4.6 Bilateral Classification 34
4.4.7 Grouping 34
4.4.8 Truncation 35
4.4.9 Generalization 35
4.4.10 Black Marker 36
4.4.11 Hashing 36
4.5 Comparing Anonymization Techniques 37
4.6 Attacks on Privacy 38
4.6.1 Injection Attack 38
4.6.2 OS Fingerprinting 39

5 Methodology **41**

6 Validation of Anonymization **43**
6.1 Initial Approach 43
6.2 Implementation Process 44
6.2.1 Parsing of PCAP Files 45
6.3 Theory 45
6.3.1 Entropy 46
6.3.2 Mutual and Normalized Mutual Information 46
6.3.3 L1-Similarity 47
6.4 Testing Approach 47
6.4.1 Log Decision Process 49
6.4.2 General Approach for Logs 50
6.4.3 Validation Process 50
6.4.4 Mapping Process 54

7 Results **59**
7.1 IPv4+TCP/UDP header Log 60
7.2 IPv6+TCP/UDP Header Log 61
7.3 NetFlow Log 62
7.4 Web Server Log 63
7.5 Syslog 63
7.6 OS Fingerprinting Prevention 64

8 Discussion **67**
8.1 Log Results Discussion 67
8.1.1 IPv4+TCP/UDP Header Log 67
8.1.2 IPv6+TCP/UDP Header Log 68
8.1.3 NetFlow Log 69
8.1.4 Web Server Log 69
8.1.5 Syslog 70
8.1.6 OS Fingerprinting Prevention Results 70
8.2 Proposal 71
8.3 SafePcap 72
8.4 Limitations 72
8.4.1 Implementation Issues 72
8.4.2 Auxiliary Information 73
8.4.3 Log Sizes 74
8.4.4 Formatting 74

9 Conclusion **75**

References **77**

Appendices

A Average entropy vs. max entropy Figures **83**

B SafePcap vs. Crypto-PAn **93**

C Comparing fields for Inter-records **97**

List of Figures

4.1 Pseudonymization mapping. 30

5.1 The general methodology for the thesis. 42

6.1 A process flow diagram. 49
6.2 Log with assigned object number and type. 51
6.3 Anonymized object selection. 56
6.4 Log feature selection. 56
6.5 Unanonymized object selection 57

7.1 Average entropy vs. max entropy for IPv4+TCP/UDP header log - hosts 60
7.2 Average entropy vs. max entropy for OS fingerprinting prevention - hosts 66

A.1 Mismappings for IPv4+TCP/UDP header log - hosts 83
A.2 Average entropy vs. max entropy for IPv4+TCP/UDP header log - web pages 84
A.3 Mismappings for IPv4+TCP/UDP header log - web pages 84
A.4 Average entropy vs. max entropy for IPv6+TCP/UDP header log - hosts 85
A.5 Mismappings for IPv6+TCP/UDP header log - hosts 85
A.6 Average entropy vs. max entropy for IPv6+TCP/UDP header log - web pages 86
A.7 Mismappings for IPv6+TCP/UDP header log - web pages 86
A.8 Average entropy vs. max entropy for NetFlow log - hosts 87
A.9 Mismappings for NetFlow log - hosts 87
A.10 Average entropy vs. max entropy for NetFlow log - web pages 88
A.11 Mismappings for NetFlow log - web pages 88
A.12 Average entropy vs. max entropy for web server log - hosts 89
A.13 Mismappings for web server log - hosts 89
A.14 Average entropy vs. max entropy for syslog - hosts 90
A.15 Mismappings for syslog - hosts 90
A.16 Mismappings for OS fingerprinting prevention - hosts 91

A.17 Average entropy vs. max entropy for OS fingerprinting prevention - web pages 91
A.18 Mismappings for OS fingerprinting prevention - web pages 92

B.1 SafePcap vs. Crypto-PAn - hosts 93
B.2 Mismappings SafePcap vs. Crypto-PAn - hosts 94
B.3 SafePcap vs. Crypto-PAn - web pages 94
B.4 Mismappings SafePcap vs. Crypto-PAn - web pages 95

C.1 How to compare fields for inter-records - IPv4+TCP/UDP header log . 97
C.2 How to compare fields for inter-records - IPv6+TCP/UDP header log . 98
C.3 How to compare fields for inter-records - NetFlow log 98
C.4 How to compare fields for inter-records - Web server log 99
C.5 How to compare fields for inter-records - Syslog 99

List of Tables

3.1 The evaluation of IPv4 header log fields 22
3.2 The evaluation of IPv6 header log fields 22
3.3 The evaluation of TCP header log fields 23
3.4 The evaluation of UDP header log fields 23
3.5 The evaluation of NetFlow log fields 24
3.6 The evaluation of web server log fields 25
3.7 The evaluation of system log fields 25

7.1 The results for IPv4+TCP/UDP header log validation. 61
7.2 The results for IPv6+TCP/UDP header log validation. 62
7.3 The results for NetFlow log validation. 63
7.4 The results for web server log validation. 63
7.5 The results for syslog validation. 64
7.6 The results for OS fingerprinting prevention validation. 65

List of Acronyms

AAPI Anonymization Application Programming Interface.

AO Anonymized Object.

AS Autonomous System.

BGP Border Gateway Protocol.

BPP Bytes per Packet.

BPS Bytes per Second.

DCCP Datagram Congestion Control Protocol.

DSCP Differentiated Services Code Point.

ECN Explicit Congestion Notification.

EU the European Union.

GDPR General Data Protection Regulation.

HMAC Hash-based Message Authentication Code.

HTML Hypertext Markup Language.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

IANA Internet Assigned Numbers Authority.

IETF Internet Engineering Task Force.

IHL Internet Header Length.

IP Internet Protocol.

JAR Java ARchive.

NTNU Norwegian University of Science and Technology.

OS Operating System.

PBKDF2 Password-Based Key Derivation Function 2.

PPS Packets per Second.

RFC Request for Comments.

SCTP Stream Control Transmission Protocol.

SHA Secure Hash Algorithm.

TCP Transmission Control Protocol.

ToS Type of Service.

TTL Time To Live.

UDP User Datagram Protocol.

UO Unanonymized Object.

URL Uniform Resource Locator.

WWW World Wide Web.

XML Extensible Markup Language.

Title:	Data Anonymization for Research
Student:	Petter Ødegård

Problem Description:

Protection of user data is a hot topic ever since the introduction of the GDPR to the European market in May 2018. User data is collected through network traffic and stored in different logs at the companies. Some companies provide researchers with logs for further analysis and want to continue with this practice. The thesis will investigate which fields of different network traffic logs are in need of anonymization and what kind of anonymization they need. A focus of this investigation is that researchers should still be provided with useful data to some extent.

A set of state-of-the-art anonymization techniques will be applied to a realistic data set and the techniques will be compared. An evaluation of the effectiveness of the anonymization techniques will be performed to observe how the techniques manage against common and probable attacks. A novel attempt to quantify the risk of personal data being revealed will be presented. Thesis work will culminate in a recommendation of best practice procedures for preparing data sets to be offered researchers.

Responsible professor:	Otto Jonassen Wittner, IIK
Supervisor:	Otto Jonassen Wittner, IIK

Chapter 1 Introduction

Data anonymization has been highly relevant since the GDPR was implemented in May 2018. The financial penalties related to breaches of the regulation meant that companies needed to evaluate their own use of stored user data. The angle of this master thesis is to focus on GDPR compliance from a researcher perspective. A researcher requests to receive network logs from a network capturing company. To be able to perform the most accurate analysis for their work, the researchers need real-world traffic, with as little manipulation as possible. How this can be accomplished within the boundaries of the GDPR will be put forward and discussed in this thesis.

Based on currently available anonymization techniques that can satisfy the GDPR, there are several combinations to look into. To be able to differentiate good and bad approaches, a validation method is implemented. In addition to different anonymization techniques, different logs will also be investigated. They contain fields with special characteristics, and can thus require different combinations of anonymization techniques.

1.1 Scope of the Thesis

The thesis describes and compares multiple methods to cover GDPR compliance when dealing with data anonymization. The logs investigated are Internet/Transport layer logs, NetFlow logs, web server logs and system logs. From the internet protocol stack only the Internet layer and Transport layer are considered. Other layers, like the Application layer and the Link layer, are out of scope. For the Transport layer, only Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are considered, meaning that protocols like Stream Control Transmission Protocol (SCTP) and Datagram Congestion Control Protocol (DCCP) are out of scope. Further, all fields of these logs will be described, and an evaluation of which fields that need anonymization is provided. Some of the logs can have various header formats, but the formatting done for this thesis is performed based on common practice and usefulness of the fields. Only anonymization techniques that are deemed appropriate for some field are included, leaving techniques like Tcpdpriv and encryption out of scope.

1.2 Justification, Motivation and Benefits

The need for data protection has been increasing ever since the introduction of the World Wide Web (WWW) to the public in August 1991 [his]. The Data Protection Directive was adopted in 1995 and was a privacy directive to protect the processing of individuals' personal data and its movement in the European Union (EU) [dir].

However, since then the internet and its applicability in today's society has exploded, and to a larger extent than first thought when the regulation was developed. More and more user data are captured based on our interaction with mobile devices, internet of things, and everyday technology. The idea of a new regulation to cope with this new world of technology was adopted in April of 2016 [gdpp] and developed into the GDPR, which was officially implemented in the EU in May 2018 [gdpo]. The GDPR puts an updated spin on storage of personal data, and provides companies with incentives to follow the guidelines: Large fines, up to 4% of gross income, and always with a minimum fee will be demanded for any breach of the GDPR, as explained in Article 83(5) [gdpk]. The GDPR is a regulation, which means that it is legally binding within the EU and other nations that cooperate with a nation within the EU. This is unlike the Data Protection Directive, which as a directive, was more of a suggestion than a legal binding [reg].

The GDPR motivates companies to more seriously consider how they deal with the data they capture and store. Since so much personal information is stored all over the internet, the damage of information about an individual leaking out could be costly. Whereas the old directive was unable to impact companies, the individuals where more often the victims. After GDPR, companies can also be on the losing end if personal data are compromised.

There are multiple reasons for data being collected. In many cases companies need the data to operate services correctly, and therefore store personal data out of necessity. Other reasons are listed in GDPR Article 6(1) [gdpj]. But for a company which has close relations to researchers and wish to support their work by providing them with data, the road to GDPR compliance is more difficult. When you want to share data with researchers you will need to store data for longer time periods, and have to consider another aspect of GDPR. Article 89 [gdpl] describes data storage for scientific or historical research, among others, and gives information as to how the process of providing researchers with data should be handled. As an introduction to the topic, there is a requirement in this article about pseudonymization to be allowed to distribute data logs to researchers. Pseudonymization differs from anonymization, and this will be covered in Section 4.2. The terms anonymization and pseudonymization are used interchangeably for the reasons explained in 4.3. For researchers there are several network traffic logs that can be of interest. They all contain different header fields, which may or may not contain personal data. To find out how this pseudonymization should be performed to be within the GDPR boundaries, and which fields in the logs need pseudonymization, are the essential motivations for this master thesis.

1.3 Research Questions

The thesis contains multiple parts, as explained in the problem description. The research questions related to the explanation is based on the effort one needs to put in to be able to get access to the original data from anonymized data. While most anonymization/encryption/hashing can be broken in theory, the interesting element is how much computer power and time is needed to perform the deanonymization process. The analysis is focused on the ability an attacker has to relate anonymized data to unanonymized data, by looking at the entropy of the log. The entropy is a number indicating how well distributed the information in a log is. The research questions are as follows:

RQ1: To what extent can state-of-the-art anonymization tools in combination anonymize personal data sufficiently to comply with the GDPR while still offering content that can tell something about user behavior to researchers?

RQ2: In network traffic logs, which are the vulnerable fields regarding personal data, and to what extent do they need to be anonymized so that no personal data can be captured?

1.4 Contributions

The contribution with this master thesis is an in-depth analysis of different levels of anonymization, for logs which, based on knowledge from literary review, have not been evaluated in this way before. Through this master thesis some anonymization methods have been developed, and are discussed in Chapter 4.

The thesis also contributes with an understanding of the GDPR in relation to user data handling in special cases. Researchers requesting data is presenting a new challenge with the new regulation, and an analysis into which parameters have to be in place for such a transaction to be allowed, is sorely needed. There are many articles that try to deal with the GDPR perspective of research and health data [MME+18], [Cha17], [SG18], [MBBvD16], [Cor18], [Lea18]. There are also several techniques existing for anonymizing different fields in logs [cry], [tcpc], [pkta]. Even a sound validation process is available [CWK$^+$08]. However, these factors have never been put together, based on knowledge from literary review, as extensively as done in this master thesis. The articles regarding the GDPR never touch technical issues, only the legal ones. Most of the anonymization techniques are old, most more than a decade old, and have not been developed with the GDPR in mind. It is rarely considered that researchers want to maintain the structure of the original data after anonymization for other fields than Internet Protocol (IP) addresses. The preexisting validation process [CWK$^+$08] has done a comparison of two anonymization techniques, but this was done solely with techniques for IP addresses. The logs considered in this mentioned validation process were also quite limited when compared to traffic logs, which are requested today and investigated in this thesis.

The thesis concludes with a proposal of how to handle pseudonymization in general, a description of which fields that contain personal information and how to handle these fields, recommendations and suggestions of how to alter the approach if more security is needed. Based on the techniques supplied there is potential to tweak the anonymization based on the needs of the individual researcher.

Implemented code for formatting a PCAP file to a text file, NetFlow formatting, web server log formatting, syslog formatting, log hashing, generalization, IP truncation, and a working implementation of the validation process from [CWK$^+$08] can be found at https://github.com/petterod.

1.5 Thesis Outline

The thesis will start with a look into background and related work in Chapter 2. Here GDPR, anonymization and validation background information will be covered. Moving on, an inspection of the log fields is provided in Chapter 3. The explanation of anonymization techniques and methodology can be found in Chapter 4 and 5, respectively. The validation process and results are covered in Chapter 6 and 7, respectively. Discussion is presented in Chapter 8, before conclusion wraps up the thesis in Chapter 9.

Chapter 2
Background and Related Work

The thesis is divided into the anonymization phase and the validation phase. Work contributing to this thesis are covered in both phases in this section. For the anonymization phase work related to IP address anonymization, Uniform Resource Locator (URL) anonymization and other general techniques are introduced. For the validation phase, the background is covered briefly, as it needs further explanation in Chapter 6. As a foundation for this, an investigation into the GDPR is first provided.

2.1 The General Data Protection Regulation

2.1.1 Initial Idea

Since 1995 the Data Protection Directive has been in place for companies to follow within the EU [dir]. But times change, and it is fair to say that the internet rules the world of today in a more significant way than it was ever possible to imagine when the WWW was introduced in 1991 [his]. As the internet is increasingly integrated into everyday life, better handling of the free-flowing data captured over the internet is sorely needed. Having been worked on for several years, the GDPR was finally put into action in May 2018 [gdpo]. With large penalties in place for companies breaching the regulation, the financial incentive is expected to motivate compliance.

Many companies end up with user data based on interaction with their customers. This information is not necessarily needed for their services, and is thus not essential to store in an unfiltered fashion. Article 6(1) [gdpj] in the GDPR specifies reasons for storing personal data. Mostly, companies are allowed to store data if the service they provide depend heavily on this information, or they have consensus from the users. Otherwise, storage of user specific data might be problematic. In addition, users have the right to erasure, as specified in Article 17(1) [gdpc]. But where does researchers enter the picture in this context?

2.1.2 Implications for Research

While providing researchers with network data is not the primary objective of a company, the situation is left an own section in the GDPR. Article 89(1) [gdpl] in the GDPR is close to addressing the needed implication in the topic covered by this thesis. Since Article 89(1) is of great importance to the understanding of data provided to researchers, a closer look at it is necessary:

> "Processing for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes, shall be subject to appropriate safeguards, in accordance with this Regulation, for the rights and freedoms of the data subject. Those safeguards shall ensure that technical and organisational measures are in place in particular in order to ensure respect for the principle of data minimisation. Those measures may include pseudonymisation provided that those purposes can be fulfilled in that manner. Where those purposes can be fulfilled by further processing which does not permit or no longer permits the identification of data subjects, those purposes shall be fulfilled in that manner".

A company delivering data to researchers therefore needs to have a working method of pseudonymizing the personal data they intend to share, and strict policies for handling both the data and the activities of the researchers are required, as also discussed in [MME$^+$18]. A practical suggestion for this is provided in Section 4.3. Article 5(1) b and e [gdpi] specifies that processing of personal data for other purposes than the original[1] done in accordance with Article 89(1) shall not be in conflict with the original purpose, and that if Article 89(1) is followed, the amount of time the person is allowed to be left identifiable and stored increases. The previously mentioned right to erasure explained in Article 17(1) is also set aside, as Article 17(3) d puts it, [gdpc] "for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes in accordance with Article 89(1) in so far as the right referred to in paragraph 1 is likely to render impossible or seriously impair the achievement of the objectives of that processing". All of this shows that as long as Article 89(1) is followed, processing personal data for other purposes than what they were originally intended for, is possible and taken into account by the GDPR.

2.2 Anonymization

The anonymization phase consists of both which anonymization techniques to use and which fields to use them on. van Dijkhuizen and van der Ham did research related

[1] "archiving purposes in the public interest, scientific or historical research purposes or statistical purposes"[gdpi].

to both topics in [DH18], from an intrusion detection system perspective. Initially they did an analysis on which fields that need protection in different layers of the internet protocol stack, before they compared anonymization techniques on a surface level. Surface level means that they did not pay much attention to the security of the techniques, but rather which attributes they maintained. The attributes range from if the technique could handle IPv6 anonymization or tunneling, did recalculation of header checksum, if it was still operational with available source code, and more. As an overview of nearly the entire topic, this article was tremendously helpful. It confirmed many anonymization tools that were already in consideration to be used in the thesis work.

Anonymization has been performed since before the GDPR was ever thought of. As several other anonymization techniques for IP addresses either incorporate it or are based on it, the method of Crypto-PAn [cry] has been essential. This IP traffic trace anonymization method was developed by Fan et al. in 2001 and is described in [XFAM01], [XFAM02], [FXAM04]. It further established prefix-preserving anonymization as a method to sanitize IP addresses, which will be discussed more closely in Section 4.4.1. The prefix-preserving idea of Crypto-PAn originated from Greg Minshall with tcpdpriv [tcpa], and was improved upon by Crypto-PAn. Later multiple anonymization techniques have been developed to either improve flaws of Crypto-PAn or try different approaches. This thesis will cover some of these techniques in more detail and also provide result comparisons between them.

One of these techniques is Tcpmkpub [tcpc]. Tcpmkpub was developed by Pang et al. in [PAPL06] in 2006 and is a trace anonymization tool. It handles anonymization at different layers of the internet protocol stack, such as the link layer, network layer and the transport layer. Crypto-PAn is partially used in their approach for IP addresses in the network layer. For this thesis the approach for the IP addresses will be further investigated in Section 4.4.2.

PktAnon [pkta] is a generic framework for profile-based traffic anonymization and was developed by Gamer et al. in [GMS08] in 2008. PktAnon approaches anonymization with a defensive transformation. This means that prior to a field being processed for anonymization, the approach for the field needs to be specified. You avoid adding original fields by accident, which should have been anonymized. The framework contains a collection of ways to modify data fields, and the ones used for fields in this thesis will be covered in 4.4.3.

AnonTool [ano] is a generic network trace anonymization framework and was developed by Foukarakis et al. in [FAP09] in 2009. The functionality of AnonTool is based on the Anonymization Application Programming Interface (AAPI) [KAA+06], and it works as the command line tool version of AAPI. This framework also allows different anonymization techniques for different fields specified by the user, and the appropriate ones are explored in 4.4.4.

SCRUB-tcpdump [scr] is a network packet trace anonymization tool and was introduced by Yurcik et al. in [YWH+07a] and [YWH+07b]. As with PktAnon and AnonTool it provides multiple anonymization methods for different fields in a tcpdump[2]. The chosen methods will be explained in 4.4.5.

Kuenning and Miller made suggestions for anonymizing URLs and filenames in [KM03]. They thought about two methods: splitting on user-defined regular expressions in the URL, and then 1) giving each substring a number starting from 1 and increasing for unique substring, or 2) adding a secret string to the substrings and hashing with MD5. The second method is similar to the method used for hashing of URLs in this thesis, which is explained in Section 4.4.11. Here the MD5 algorithm is replaced, and instead of user-defined regular expressions like Kuenning and Miller's method uses, the URLs are always handled the same way.

2.3 Validation

Perhaps the most demanding task of data anonymization is to find a reasonable way of measuring how good the anonymization is. Several articles focus on how to handle the issue, but fall short when it comes to either the wanted validation or how an implementation should work. Approaches are mostly related to speed or storage, not security. After a thorough and deep literature search for a suitable method, an article which combined network traffic logs with a mathematical approach to anonymization measurement was discovered. Coull et al. [CWK+08] worked out a method which included the needed properties: Validation of anonymization from a security perspective, methods for handling network traffic fields and a somewhat understandable step-by-step process. The validation of the chosen techniques is performed with their validation method. Neither running code nor source code implementing their method is currently available. Thus, a reimplementation was necessary for this thesis. The theory behind Coull et al.'s method is explained in Chapter 6. The validation process is so essential for the results that it needs to be explained in more detail.

[2]Tcpdump is a command-line packet analyzer, storing data in a PCAP file[tcpb]

Chapter 3

Network Traffic Logs

Before entering the enticing world of anonymization, an introduction and evaluation of each field from the headers of the different logs are in order. The task of this master thesis is to anonymize personal data according to the GDPR in addition to provide researchers with data possible to analyze. More precisely, four categories of network traffic will be investigated: Internet/Transport layer logs, NetFlow logs, web server logs and system logs. For each of these logs there are different fields contained in their headers. Each field needs to be evaluated: Does it need anonymization? What kind of anonymization? or is it safe to be left unchanged? A combination of fields should also be evaluated as this can reveal the identity of the sender even though the most revealing personal data is hidden. Some fields exist in multiple of the four mentioned log categories. They are evaluated independently for each log category, even though these fields will often be handled in the same way across log categories. An overview of the fields in each of the logs together with the evaluation will be presented. The chapter will conclude with a summary of the anonymization approach for each field.

The chapter is not meant as a deep analysis into each field, but rather a short description to understand if and what data in the fields can be personal, sensitive or used together with other fields to gain information about a person.

3.1 Internet/Transport Layer Log

For an Internet/Transport layer log there are considerations to be made before looking at the fields. In the Internet layer there are two versions of IP: IPv4 and IPv6. UDP and TCP are two transport protocol mostly used in the transport layer. The two considerations are whether IPv4 or IPv6 is being used, and whether UDP or TCP is being used. Other protocols than UDP or TCP can be chosen, but those are the protocols of which header investigation will take place. Note that payload is assumed to be removed from the transport layer.

3.1.1 IPv4 Header

The fields in an IPv4 header are explained in this subsection. Unless otherwise stated, the information about the fields is taken from Internet Engineering Task Force (IETF) Request for Comments (RFC) 791 [Inta]. In addition to the fields originally in the IPv4 header format, a timestamp is added to the header for the work of this thesis. The capturing of Internet/Transport layer data is based on the network traffic capture program *tcpdump* [tcpb]. *Tcpdump* uses *libpcap*, a portable C/C++ library for network traffic capture [tcpb], to capture copies of packets. The timestamps are provided by *libpcap* when the copies are made. This means that there is a layer of *libpcap* on top of the Ethernet layer, which is on top of the Internet layer, which again is on top of the Transport layer. Since this thesis is only concerned with the Internet and Transport layer the timestamps are retrieved to the Internet layer from this *libpcap* layer.

The evaluation of the IPv4 header is summarized in Table 3.1. Note that in addition to these fields every packet can contain IP Options. IETF RFC 791 [Inta] says that the IP Options are only necessary for some situations, but are not needed for most common communications. Together with the extra work load addition of the options would require, the IPv4 options are deemed not relevant and too time consuming to include for the analysis. One of the options allow for variable length, which means that every analyzed packet would need to be filled with the maximum length of the IPv4 options field. In addition, the amount of time needed to handle the variable length field correctly would not be appropriate compared to the useful data it would provide, especially since the amount of packets actually using this field is presumed low. IPv4 Options are thus removed from the header log.

Timestamp - The Timestamp tells the time the packet was sent. A timestamp might be used by an adversary to identify clock skew on a computer, which can indicate what Operating System (OS) the computer is running and be used to fingerprint a user with unique characteristics. This attack is further explained in Section 4.6.2. In addition, timestamps can be used in an injection attack, described further in Section 4.6.1. In this attack, if the adversary knows when a log is captured, he/she can inject large amounts of packets to get a mapping of the anonymization techniques used. A truncation of sorts of the timestamp would be a good measure to take for the timestamp to reveal as little information as possible that could be used further in an attacking scenario.

Changing the timestamp could have little value for injection attack protection if the attack is done extensively. In addition, the timestamp is important for a researcher in the analysis. These two aspects determine that the timestamps are left open. Anonymization frameworks allowing truncation of timestamps could be used, but the effect of it might impact the researchers more than it would prevent an attacker. Timestamps are useful for analysis and comparisons amidst logs.

Version - The Version field shows which version of IP the packet is: The value is always '4' in an IPv4 header log. This field does not risk exposing any personal information, and is safe to leave open.

IHL - The Internet Header Length (IHL) is the length of the IP header. It is not considered to possess any personal information, and is left open.

DSCP - The Differentiated Services Code Point (DSCP) field, in addition to the Explicit Congestion Notification (ECN), is used for Quality of Service and congestion notification, as explained in [DH18]. Both these fields were previously collectively called the Type of Service (ToS) field. The information these fields contain have proven to be revealing as they can identify types of routers, and user behavior can be exposed through user-defined fields. The field is therefore anonymized with the *constant overwrite* method from *PktAnon*, described in Section 4.4.3. It is validated together with fields vulnerable to OS fingerprinting attack, an attack discussed in Section 4.6.2.

ECN - The ECN field is explained together with DSCP.

Total Length - The Total Length field is the length of the packet, with IP header and the accompanied data. As with IHL, it is not dangerous for exposure of personal data, and is kept open.

Identification - The Identification field is used to identify a packet. In case a packet is fragmented, the identification number will show which fragments belong to the packet. [DH18] suggests that Identification could be used to fingerprint an OS since the algorithm to make the identification number is specific to the OS. What this means is that the increment of the Identification value is different depending on the OS. A method called *grouping*, which is explained in Section 4.4.7, is used. *Grouping* is only applied to the Identification field when OS fingerprinting attack (discussed in Section 4.6.2) is considered. The danger of identifying ones OS is also explored in this section.

Flags - The Flags field consist of three control flags, which are specified by three bits. The first is always zero, as it is reserved for future use. The second bit decides that you may fragment the packet if zero, and that you should not fragment when set to one. The third bit tells that the fragment is the last fragment if the bit is zero, and that there are more fragments following if the bit is one. [DH18] argue that the second bit, Don't Fragment, can be used to fingerprinting a machine or device. However, the risk of this is considered negligible for this work and Flags are left open.

Fragment Offset - The Fragment Offset specifies where in the packet the fragment should be. There is no risk of personal data being lost in this field, and so an open policy is applied.

TTL - The Time To Live (TTL) field shows the maximum time the packet is allowed to be in the internet system. It is measured in seconds, but since many packet processes takes less than a second, the value is effectively decreased by one for every new process. [DH18] points out that not all operating systems have the same default initial TTL value, and this can be used to reveal the OS of a computer, further described in Section 4.6.2. TTL is anonymized with *bilateral classification*, explained in Section 4.4.6.

Protocol - The Protocol field tells which protocol is used, e.g. TCP, UDP, DCCP, SCTP. As this is not revealing personal information and could be interesting to analyze for a researcher, the field is left unmodified.

Header Checksum - For verifying the header fields a checksum is calculated. The only issue with this field is that if some values of the header are changed due to anonymization, the checksum should be recalculated so that one can differentiate between incorrect packets and packets with anonymization applied in a correct manner [DH18]. *PktAnon* (Section 4.4.3) is chosen as checksum recalculation tool.

Source IPv4 Address - The IP Address field is the most personally revealing field in the IP header. The Source IPv4 Address is where the packets are sent from. The address might reveal your location, and it is therefore considered personal data. From IETF RFC 4291 [Inta], the format of an IPv4 address is *x:x:x:x*, where an x is one byte, called one octet. There are four octets, which can hold values between 0 and 255. In total an IPv4 address consists of 32 bits.

The anonymization techniques which applies to the IPv4 Address field are *Crypto-PAn*, *Tcpmkpub*, *PktAnon*, *AnonTool* and *Truncation*. These will all be explored in detail in Section 4.4.

Destination IPv4 Address - This field is equivalent to the Source IPv4 Address, only that it shows where packets are sent to. The same policy towards anonymization applies here.

3.1.2 IPv6 Header

Here the IPv6 header is explained. IPv6 has a different header format compared to IPv4, and effectively allows for more IP addresses than IPv4. Some fields are equivalent to IPv4 header fields, and will require a shorter explanation. The explanation on the addition of timestamps from the IPv4 section applies here as well. Unless otherwise stated, the IPv6 header fields descriptions are based on IETF RFC 2460 [Intb]. The evaluation is summarized in Table 3.2.

Timestamp - The timestamp is dealt with as explained for IPv4.

Version - This field is equivalent to the one for IPv4, with the version changed to '6'.

Traffic Class - The Traffic Class field allows for identification and distinction between different classes or priorities of IPv6 packets. According to [DH18], the field should be dealt with in the same manner as the DSCP and ECN fields of the IPv4 header.

Flow Label - The Flow Label field is used to mark sequences of packets that need special handling by routers. It is not known to be of any danger to personal information and is left open.

Payload Length - The Payload length is the length of the payload of IPv6, i.e. the part of the packet that is not in the header. As explained in the introduction of this chapter, the payload is completely removed from the packets. Considering this, the length is not containing any personal information and is unmodified.

Next Header - The Next Header field is equivalent to the Protocol field in the IPv4 header.

Hop Limit - The Hop Limit field functions and is dealt with as the TTL field in the IPv4 header.

Source IPv6 Address - As with IPv4 Addresses, the Source IPv6 Address field reveals personal data about the sender of the data. From IETF RFC 4291 [ipv], the format of an IPv6 address is *x:x:x:x:x:x:x:x*, where an x is between one and four hexadecimal numbers. There are eight xs, and each x can have 16 bits. In total an IPv6 address consists of 128 bits, compared to the 32 bits of an IPv4 address.

Some, but not all, of the techniques for IPv4 anonymization applies to IPv6 as well. *Crypto-PAn*, *PktAnon* and *truncation* are tested, while *Tcpmkpub* and *AnonTool* and are not developed to handle IPv6 addresses as of yet. The three applied techniques are explored in Section 4.4.1, 4.4.3 and 4.4.8, respectively.

Destination IPv6 Address - The destination equivalent of Source IPv6 Address.

3.1.3 TCP Header

The Transport layer of the internet is run over UDP or TCP. Here the fields of the TCP header are presented. Unless otherwise stated, the information about the header fields is as described in IETF RFC 793 [TCPd]. The evaluation is summarized in

Table 3.3. Table 3.3 provides a summary of the evaluation.

Note that as with IPv4 Options, TCP Options have been deemed both not relevant and too complex to include in this master thesis analysis. One of the options can have variable length, which proved too time consuming to handle in the best possible way. In addition the inclusion of TCP Options would mean the fields for these options be added to every packet, whether they have these options or not. The extra work needed to process options correctly would not equal the interesting information they could provide, and TCP Options are thus removed from the header log.

Source Port - The Source Port field identifies which port a packet is running from. Ports distinguishes between different services which can be run over transport protocols like TCP and UDP [ian]. The ranges for port numbers are [ian]:

– 0-1023: Well-known ports.

– 1024-49151: Registered user ports.

– 49152-65535: Dynamic and/or private ports.

By analyzing the dynamic and/or private ports, information can be gained about how these ports are distributed, which can indicate a unique user. This knowledge can also be used in combination with other fields to single out a user. A novel contribution to this thesis called *generalization* handles port number anonymization. *Generalization* is explained in Section 4.4.9.

Destination Port - The Destination Port is equivalent to the Source Port field, but this time the port specifies where the packet should end up.

Sequence Number - The Sequence Number field is used to structure segments in the correct order and discard duplicate segments [DH18]. The *grouping* method from *SCRUB-tcpdump* (explained in Section 4.4.5) is able to group the number into different partitions. This will prevent the possibility of OS fingerprinting (see Section 4.6.2) discussed in [DH18].

Acknowledgement Number - The Acknowledgement Number field tells the sequence number of the next segment the sender is supposed to get. This confirms that a packet was received correctly. Acknowledgement Number is thus closely related to Sequence Number, and adopts the same anonymization approach.

Data Offset - The Data Offset shows where the data begins in the packet, i.e. on the outside of the header. As the length of the header is decided to be without any personal information, the Data Offset field is left open.

Reserved - The Reserved field is said to be for further use, and is for now always left as zero. The field is left open as it contains no personal information.

TCP Flags - The Flags of TCP are used to indicate what is happening in the packet. The nine flags are NS (Nonce Sum), CWR (Congestion Window Reduced), ECE (ECN-Echo), URG (Urgent Pointer field significant), ACK (Acknowledgement field significant), PSH (Push function), RST (reset the connection), SYN (synchronize sequence numbers) and FIN (no more data from sender). The usage of each flag will not be discussed here. Since different OSes use the TCP protocol differently, it is possible to use the flags to do OS fingerprinting [DH18], as explained in Section 4.6.2. The *keyed random permutation* method from *SCRUB-tcpdump* allows for anonymization of the TCP Flags, and is explained in Section 4.4.5

Window Size - Window Size tells how much data the sender of a segment will accept, and effectively says the amount a sender will transmit before an acknowledgement comes [DH18]. The default window size can vary from OSes, so OS fingerprinting (Section 4.6.2) is a possibility. The *bilateral classification* method from *SCRUB-tcpdump* can handle this field and is explained in Section 4.4.5.

Checksum - The Checksum works like the IPv4 Header Checksum, and should thus take the same approach.

Urgent Pointer - As [TCPd] says, the Urgent Pointer field shows the sequence number of the octet following the urgent data. This is only showed when the URG flag in TCP Flags is set to one. The field is left open, as there is not known to be any risks of personal data loss.

3.1.4 UDP Header

The UDP header consists of *Source Port*, *Destination Port*, *Length* and *Checksum*. Most of these fields have already been covered by the TCP header and requires no further explanation. The fields use the same approach as for the TCP header. The Length field is briefly explained, with the information stemming from IETF RFC 768 [UDP]. The evaluation is summarized in Table 3.4.

Length - The Length field contains the value of the UDP header and the data belonging to the packet. It is left open, as no private information is contained.

3.2 NetFlow Log

The NetFlow log format is developed by Cisco [ver]. It aggregates packets into IP flows within a data network. Some flows are then aggregated, and this is seen in the *Flows* field when the value is greater than one. The format of a NetFlow log can vary based on the tool used to capture NetFlow data. For the purposes of this master thesis, nfdump [nfd] has been used as the capturing program. Nfdump is a tool for collecting and processing NetFlow data. The dumps collected with nfdump can have several fields, but the ones decided to provide value in this context are listed below. Unless otherwise stated, the fields not already explained through previously mentioned fields are described according to the Cisco IOS NetFlow Version 9 Flow-Record Format [ver]. Table 3.5 provides a summary of the evaluation.

Start Time - First Seen - This field is the timestamp of the first packet in the flow. It is handled the same way Timestamp is for IPv4.

End Time - Last Seen - This field is the timestamp of the last packet in the flow. It is also handled in the same way as the Timestamp field for IPv4.

Duration - Duration is just the difference between End Time and Start Time, and is in no need of anonymization.

Protocol - The Protocol field is as explained for IPv4 and IPv6, and follows the approach described for these logs.

Source IP Address - Source IP Address is already explained for IPv4 Addresses. However, since NetFlow logs include Source AS and Destination AS fields (explanation provided below), a different approach needs to be taken compared to the IP addresses of IPv4. *Truncation* (Se Section 4.4.8) of the last bits[1] of the IP address is the desired technique, and the reason for this will be explained further with the Source AS field description below.

Another factor for the IP addresses of NetFlow is that the anonymization tools are not capable of handling an nfdump file or a text file, which are the formats applicable to NetFlow in this thesis. This limits the testing of the addresses to *Crypto-PAn* (Section 4.4.1) and *truncation* (Section 4.4.8). *AnonTool* claims to handle NetFlow data, but their implementation says otherwise.

Destination IP Address - Destination IP Address is handled as explained for NetFlow Source IP Addresses.

Source Port - Source Port is handled as explained for TCP Source Port.

Destination Port - Destination Port is handled as explained for TCP Destination Port.

Source AS - The Source Autonomous System (AS) field describes the Source AS number for the exterior gateway protocol Border Gateway Protocol (BGP). BGP

[1]The last octet for IPv4 and the last 16 bits for IPv6.

exchanges information about which parts of a network are reachable [RLH05]. In these exchanges, AS numbers are being exchanged to inform how to reach a certain network. AS numbers are used for exterior routing on the internet, and an AS is a group of IP prefixes linked together with a network operator specifying a single and clearly defined routing policy for this group [HB96]. The concern for this field is that it will reveal where in the network the traffic is travelling. The IP addresses might be anonymized to the point where you are not able to recognize the prefixes. However, in combination with an AS number, the privacy provided for IP addresses is suddenly gone if you can use the AS number to learn the IP prefixes from the AS number. That is why the truncation approach mentioned in NetFlow Source IP Address is chosen. The AS number will reveal the prefixes no matter which anonymization technique is used, and it is then safer to disallow the full IP address with truncation.

This approach is chosen because the AS field is used for geolocation analysis, and thus provides important value to researchers. There is a desire to keep the AS numbers open. By truncating IP addresses and leaving AS numbers open, you remove the chance of pinpointing the exact location of the IP address, while still knowing which AS the traffic belongs to.

The other possibility is to use *black marker* (explained in Section 4.4.10) on the AS numbers altogether, if they are found to be too revealing of geolocations. The approaches from IPv4 and IPv6 logs regarding IP addresses can then be taken. Herein lies a problem that some of the techniques for IP address anonymization does not cover data in a NetFlow format. Due to time constraints required format conversion tools were not implemented, and this is as of now just a theoretical option.

Destination AS - This is the destination equivalent of Source AS, and is handled in the same way.

Input Interface Num - Input Interface Num tells which interface the traffic enters the router on. There are no risks of any personal information being lost here, and it is left open.

Output Interface Num - This field is the output equivalent of Input Interface Num and is similarly dealt with.

Packets - The Packets field shows how many packets are collected in the flow and is left open.

Bytes - The Bytes field is a value telling how many bytes are in the flow and is left open.

Flows - The Flows field is the number of flows collected in the particular flow. It requires no modification and is left open.

Flags - The Flags field is equivalent to the TCP Flags field, and is handled in the same manner.

ToS - The ToS field is equivalent to the combination of DSCP and ECN fields of IPv4 and is handled accordingly.

BPS - The Bytes per Second (BPS) field shows how many bytes are processed in the NetFlow per second and is left open.

PPS - The Packets per Second (PPS) field shows how many packets are processed in the NetFlow per second and is left open.

BPP - The Bytes per Packet (BPP) field shows how many bytes are processes in every packet in a flow and is left open.

3.3 Web Server Log

A web server log contains requests for the specified web server. There are several log formats, but the web server logs investigated here are using Common Log Format from an Apache HTTP Server [com]. This format consists of the following fields, which are described based on [com] unless otherwise stated. The evaluation is summarized in Table 3.6.

IP Address - IP addresses are already well documented, but the anonymization technique options are limited for the web server log. Both *Tcpmkpub*, *PktAnon* and *AnonTool* are basing their anonymization on a PCAP file, while the web server log comes in the format of a text file. The techniques will then not be able to recognize the IP address fields in the text file. Due to time constraints required format conversion tools were not implemented. *Crypto-PAn* and *truncation* are still applicable techniques to a web server log and explanations of these techniques can be found in Section 4.4.1 and 4.4.8, respectively.

Identification Protocol - The Identification Protocol is a field which is used to determine identity of a user of a particular TCP connection, as stated in [ide]. It can return a string identifying the owner of a TCP port number pair on the system of the current server. This field is according to [com] almost never used, often arriving with '-' already set. Since there is a high probability that this field would reveal a real identity, *black marker* is used, replacing any potential value with a '-'. *Black marker* is explained in Section 4.4.10.

Userid - This field is the Userid of the person requesting the document as determined by HTTP authentication, as stated in [com]. Because of the sensitive nature of the information - you do, after all, concede the identity of the user - black marker is applied to this field, as with the Identification Protocol.

Timestamp - The timestamp has been well established and the approach will be as in the other logs.

Request Line - This field is a combination of several fields: Request Method, the Request, and the HTTP-Version. They need to be individually handled, and explanations follow below.

Request Method - The request method specifies the method used by Hypertext Transfer Protocol (HTTP) when the request for/from the web server is sent. To be sure that a safe method is used, a check is done confirming whether or not the field equals one of the standardized request methods from HTTP. If not, then a '-' is returned as its value. The allowed request methods are GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT and PATCH. The first eight methods can be found in IETF RFC 7231 [rfca], while PATCH was defined in IETF RFC 5789 [rfcb].

Request - This field tells what resource was requested by the client [com]. Oftentimes this is a URL to a certain web page. The content of the URL is the target of the client. This field can in theory contain anything, from regular strings, to email addresses, IP addresses, usernames, phone numbers and even full names. As must be obvious, this field needs special care regarding anonymization. What most people relate to a URL, e.g. https://www.ntnu.no/studentliv/trondheim, does not contain any personal information. But you can never be certain that the next URL only contains safe strings with no relation to the person accessing it. This also goes beyond personal data and enters possibly sensitive data. The difference between personal and sensitive data is explained in Section 4.1. Therefore a strict policy must be in place.

For strings between '/', '?', '=' and '&', *hashing* is chosen as the appropriate method. The same string is hashed the same way, and in this way researchers can see where the string is the same as another, without any personal information getting lost. By splitting the components between these special characters, and hashing every component, you effectively hide personal and other information that could be contained in the URLs, while allowing researchers to analyze patterns in how these special characters are used. The hashing used will be described in more detail in Section 4.4.11.

What is important to note here is that this field may not just contain a URL.

Even if you have chosen an allowed request method (as discussed in Request method) you may end up with user-inputted data. This is another reason why the content of the URL field is handled as strictly as it is with hashing.

HTTP-Version - HTTP-version field is the version of HTTP used in the request/response by the client/server. As with request method, there is not anything dangerous in this field. When formatting the web server log, as with request method, a check is performed to see if the value is an allowed HTTP version. A '-' is returned if not. The versions that are allowed are HTTP/0.9, HTTP/1.0, HTTP/1.1, HTTP/2.0 and HTTP/3.0.

HTTP-Status Code - Perhaps the most interesting field of a web server - that is, to researchers - is the HTTP-status code. From this you will get the feedback from the web server when the URL request was made, and as such you can analyze how web servers are maintained, if the request was successful, if there are any malicious clients out to test for weaknesses, etc. [com]. From all the benefits of this field, together with the fact that no personal information can be misplaced, HTTP-status code is left open.

Object Size - The Object Size field indicates the size of the object returned to the client [com]. This can indicate what kind of URL was requested. Some URLs are only Hypertext Markup Language (HTML) pages with text, and thus require a small size, while others contain pictures, video or other applications and are larger in size. There is, however, no personal information kept in this field, and it is therefore left open.

3.4 System Log

System log (syslog) shows actions taken inside an operating system. This section addresses the fields in a syslog. The log containing the fewest fields, the system log is provided to researchers to map how systems are, and what software is, frequently used. The evaluation is summarized in Table 3.7. The following fields are the ones provided for the log analysis, and are defined in IETF RFC 5424 [rfcc].

Timestamp - As already discussed, the timestamp displays when a certain action happened, and the approach is the same as previously mentioned timestamps.

Hostname - The machine which originally sent the syslog message is specified in this field, as stated in [rfcc]. Since this can be anything, from a string to an IP address, the Hostname is hashed, see Section 4.4.11.

App-name - The App-name field is used to show which application originated the message in the system [rfcc]. This is the most interesting field in a syslog when it comes to fields that are not containing any personal information, and is left open. Researchers are then allowed to log which applications are often accessed.

Message - Message describes what action was performed by the software at the given time [rfcc]. A message can contain a wide variety of information based on the action taken. The message, for the purposes of this thesis, is typically made up of several subfields. They are Procid, Msgid, and Msg, the message itself. As the format varies with the application and the task, some of the subfields might be missing. Since there is a lot of information about the system in which the syslog is running, and the format can vary, the message should be hashed (Section 4.4.11). The hashing is done on each component separated by a space in the message. This allows researchers to observe when an action is done in several packets, without knowing what action it is. The approach for a syslog is considered a novel contribution, as no information on this has been discovered.

3.5 Summary

IP addresses need to be anonymized in some way. Several fields can be subject to OS fingerprinting attack (Section 4.6.2), and it needs to be checked how much better the anonymization of objects can get when anonymizing these fields. They include DSCP, ECN, Identification, TTL, Traffic Class, Hop Limit, Sequence Number, Acknowledgement Number, TCP Flags and Window Size. Every checksum should be recalculated after field anonymization has been done.

Ports are an interesting source of information to a network traffic analyst. They reveal which services are used. It is, however, possible to recognize single users by analyzing the ports. Many ports are well-known, and their use common. When more personal applications are run, however, there might be that one port is used only once in a log with millions of packets. This increases the possibility of recognizing a single user in the data log, and thus an anonymization of sorts should be applied. Ports, in combination with a field revealing the OS of the host, could be used to single out the host. Fields such as Identification Protocol, Userid and Request from web server logs, and Hostname and Message from syslog, are also anonymized.

What follows are tables summarizing the anonymization approaches for all log fields. The anonymization techniques will be further explained in Section 4.4. When technique names for the columns are abbreviated, C stands for Crypto-PAn, T for Tcpmkpub, P for PktAnon, A for AnonTool, S for SCRUB-tcpdump, Tr for Truncation, BC for Bilateral Classification, G for Grouping and BM for Black Marker. An 'X' in a row indicates that the anonymization technique for this field is decided. An 'A' in a row indicates that there are alternative anonymization techniques to compare the impact of the alternatives.

Table 3.1: The evaluation of IPv4 header log fields

Field	C	T	P	A	Tr	BC	G	Leave open
Version								X
Timestamp								X
IHL								X
DSCP			A					A
ECN			A					A
Total length								X
Identification							A	A
Flags								X
Fragment offset								X
TTL						A		A
Protocol								X
Header checksum			A					A
Src IPv4 address	A	A	A	A	A			
Dst IPv4 address	A	A	A	A	A			

Table 3.2: The evaluation of IPv6 header log fields

Field	Crypto-PAn	PktAnon	Truncation	Bilateral Classification	Leave open
Version					X
Timestamp					X
Traffic class		A			A
Flow label					X
Payload length					X
Next header					X
Hop limit				A	A
Src IPv6 address	A	A	A		
Dst IPv6 address	A	A	A		

Table 3.3: The evaluation of TCP header log fields

Field	PktAnon	SCRUB-tcpdump	Generalization	Leave open
Src port			X	
Dst port			X	
Seq number		A		A
Ack number		A		A
Data offset				X
Reserved				X
Flags		A		A
Window size		A		A
Checksum	A			A
Urgent pointer				X

Table 3.4: The evaluation of UDP header log fields

Field	PktAnon	Generalization	Leave open
Source port	X		
Destination port	X		
Length			X
Checksum	A		A

Table 3.5: The evaluation of NetFlow log fields

Field	PktAnon	SCRUB-tcpdump	Generalization	Truncation	Leave open
Start time - first seen					X
End time - last seen					X
Duration					X
Protocol					X
Src IP address				X	
Dst IP address				X	
Src port			X		
Dst port			X		
Src AS					X
Dst AS					X
Input interface num					X
Output interface num					X
Packets					X
Bytes					X
Flows					X
Flags		A			A
ToS	A				A
BPS					X
PPS					X
BPP					X

Table 3.6: The evaluation of web server log fields

Field	C	P	Tr	Hashing	Black Marker	Leave open
IP Address	A	A	A			
Identification Protocol					X	
Userid					X	
Timestamp						X
Request method						X
Request				X		
HTTP-version						X
HTTP status code						X
Object size						X

Table 3.7: The evaluation of system log fields

Field	Hashing	Leave open
Timestamp		X
Hostname	X	
App-name		X
Message	X	

Chapter 4 Anonymization

This chapter presents the difference between personal and sensitive information, the difference between anonymization and pseudonymization, and the theoretical solution to pseudonymization. This theoretical solution is the backbone for how anonymization is performed. Further, every anonymization technique utilized in this master thesis is explained, before IP address techniques are compared. Finally, a look into two attacks for network traffic logs is discussed.

4.1 Personal vs Sensitive Data

The GDPR requires that companies deal with personal and sensitive data from users in a secure way. Important for this discussion is what constitutes these kinds of data. Article 4(1) and 9(1) explain the meaning.

Article 4(1) on personal data [gdph]:

> 'Personal data' means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person.

Article 9(1) on sensitive data [gdpm]:

> Processing of personal data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, or trade union membership, and the processing of genetic data, biometric data for the purpose of uniquely

> identifying a natural person, data concerning health or data concerning a natural person's sex life or sexual orientation shall be prohibited.

Related to the data in the available logs here, IP addresses can be categorized as personal data, as it can reveal the location of the user. URLs can be seen as personal and/or sensitive data, as a lot of what makes you a person can be found in the URLs that you visit, such as sites you log into and searches you perform based on own interests and culture. This is information you might not want to be public, because of possible discrimination or other unpleasant experiences.

4.2 Anonymization vs Pseudonymization

When it comes to private data in the context of the GDPR, the difference between anonymized data and pseudonymized data is significant. Anonymization is the process of sanitizing data to protect the privacy of objects or persons to the point where you are unable to identify them [DH18]. Recital 26 [gdpn] of the GDPR says this about anonymized information: "Information which does not relate to an identified or identifiable natural person or to personal data rendered anonymous in such a manner that the data subject is not or no longer identifiable."

In the GDPR it is explained that the personal data needs to be manipulated - called pseudonymization in article 89(1) [gdpl] - in such a way that you can return from the pseudonymized data back to the original data. From Article 4(5) of the GDPR [gdph], pseudonymization is defined as:

> 'Pseudonymisation' means the processing of personal data in such a manner that the personal data can no longer be attributed to a specific data subject without the use of additional information, provided that such additional information is kept separately and is subject to technical and organisational measures to ensure that the personal data are not attributed to an identified or identifiable natural person.

The process of returning to the original data - the deanonymization - requires that some information is kept secret from the researchers receiving the pseudonymized data. What comes to mind when thinking of such information is perhaps a key for a decryption scheme, a salt for a hash function, etc. The interpretation of pseudonymization used for this thesis is explained in Section 4.3.

The difference in anonymized and pseudonymized data then lies in the possibility to re-identify the person associated with the data. In anonymization you destroy the original value for the fields, while pseudonymization replaces the value with a pseudonym, and allows you to re-identify the original value by some additional information.

4.3 Pseudonymization for Network Traffic Logs

By now, it has been established that pseudonymization is required, by Article 89(1) [gdpl], to comply with the GDPR. This section discusses how the pseudonymization can be performed for a company. While it is explained how a company can perform the pseudonymization process, physical code for the deanonymization of the applied anonymization techniques for this process to work is not developed. The limited time for the thesis puts a restriction on the practical solution for this. A theoretical solution will, however, be suggested.

A key issue between GDPR compliance and pseudonymization is the possibility of deanonymizing the data, as defined in GDPR Article 17(1), "the right to erasure" [gdpc]. If a user requests his or her data to be removed from storage, then the company should be able to locate this user in the pseudonymized log. Article 17(3d) [gdpc], however, opens up for an exception to the right to erasure:

> "Paragraphs 1 and 2 shall not apply to the extent that processing is necessary for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes in accordance with Article 89(1) in so far as the right referred to in paragraph 1 is likely to render impossible or seriously impair the achievement of the objectives of that processing".

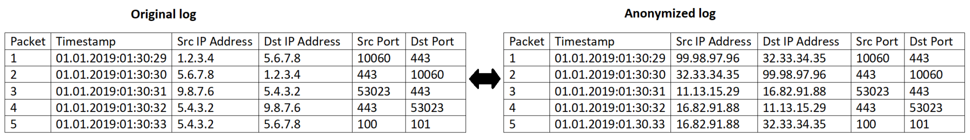
What this means is that a company needs to pseudonymize the data in case of the possibility that a person requests erasure. In the situation where this erasure would damage a researcher's results, there can be an exception to the rule. However, there are other articles of the GDPR, from Article 15 (Right of access by the data subject) [gdpa], Article 16 (Right to rectification) [gdpb], Article 18 (Right to restriction of processing) [gdpd], Article 19 (Notification obligation regarding rectification or erasure of personal data or restriction of processing) [gdpe] and Article 20 (Right to data portability) [gdpf], that effectively says that a company has to be able to reverse manipulated data provided to a researcher, even when Article 89(1) is followed. None of these articles include an exception like Article 17 does. Hence ensuring the ability to reverse pseudonymization seems to apply also for research datasets.

As Article 4(5) [gdph] of the GDPR says, you should store the key or other additional information to reverse pseudonymization safely at the data collector. The

GDPR does not, however, specify how the pseudonymization should work. The regulation only specifies that you should be able to reverse the pseudonymization with some additional information, like a key. During the search for pseudonymization techniques, very few techniques that fulfill this reversible feature was found. Crypto-PAn is really the only one that incorporate it for IP addresses.

To be able to perform validation and compare different anonymization techniques without reversal, a way to bypass this obstacle was discovered. The GDPR requirement is that it is possible to go from the pseudonymized data back to the original. One way to do this is storing the mapping between the two directly, such that the first packet of the original log is the first line of the anonymized log. This mapping relationship is shown in Figure 4.1, where the IP addresses have been anonymized in the anonymized log.

Much storage space will be needed for this, but it is the only way that makes it possible to remove personal information, perform the wanted analysis, and be within the boundaries of the GDPR, based on literary review. The mapping between the pseudonymized and the original data would be encrypted, and the decryption key stored the same way any key for pseudonymization would. It would never be used unless there is a request for deanonymization, and otherwise the mapping would be unavailable for anyone not authorized. A researcher would never be in any close connection with this mapping, and a written agreement between the researcher and the company regarding what is required of the researcher would be mandatory. At first thought, one would want a decryption key used directly on the pseudonymized data to comply with the GDPR deanonymization demand. However, losing this key would still make it possible to get back to the original data, in the same way that the proposed storage of the original and pseudonymized mapping would. Therefore, confidence is placed in this approach to work both in the thesis and later for any company in need of pseudonymization. The anonymization techniques will not need to hold any specific properties to return to the original data, as this mapping is already acquired through the described way of anonymizing the logs and storing the originals.



Original log

Packet	Timestamp	Src IP Address	Dst IP Address	Src Port	Dst Port
1	01.01.2019:01:30:29	1.2.3.4	5.6.7.8	10060	443
2	01.01.2019:01:30:30	5.6.7.8	1.2.3.4	443	10060
3	01.01.2019:01:30:31	9.8.7.6	5.4.3.2	53023	443
4	01.01.2019:01:30:32	5.4.3.2	9.8.7.6	443	53023
5	01.01.2019:01:30:33	5.4.3.2	5.6.7.8	100	101

Anonymized log

Packet	Timestamp	Src IP Address	Dst IP Address	Src Port	Dst Port
1	01.01.2019:01:30:29	99.98.97.96	32.33.34.35	10060	443
2	01.01.2019:01:30:30	32.33.34.35	99.98.97.96	443	10060
3	01.01.2019:01:30:31	11.13.15.29	16.82.91.88	53023	443
4	01.01.2019:01:30:32	16.82.91.88	11.13.15.29	443	53023
5	01.01.2019:01.30.33	16.82.91.88	32.33.34.35	100	101

Figure 4.1: Pseudonymization mapping.

Throughout this thesis, anonymization is often used as the term to remove personal information from a log even though the GDPR specifies that it should be pseudonymization. This is done because of the approach described in this section. The anonymized data is handled as if it were pseudonymized thanks to the mapping between the original and the modified data.

4.4 Anonymization Techniques

In this section the different anonymization techniques used in the thesis will be outlined. They apply to several fields, like IP addresses, ports, URLs, flags, sequence number, identification, etc.

4.4.1 Crypto-PAn

Crypto-PAn [cry] is a cryptography-based sanitization tool made by Fan et al. in [XFAM01], [XFAM02], [FXAM04]. Crypto-PAn expands on prefix-preserving anonymization used by Minshall for Tcpdpriv [tcpa]. [XFAM01] explains prefix-preserving anonymization as if two original IP addresses share a bit prefix, their anonymized mappings will also share a bit prefix. The calculation itself is based on pseudo-random permutation with a provided cryptographic key and XORing an original bit with a permuted bit. To illustrate the method of Crypto-PAn, consider this example taken from a sample trace log provided by Crypto-PAn (O_1 and O_2 are the original addresses, A_1 and A_2 are the anonymized ones): $O_1 = 24.5.0.80 \rightarrow A_1 = 100.9.15.210$, $O_2 = 24.0.250.221 \rightarrow A_2 = 100.15.198.226$. Observe that O_1 and O_2 share a prefix, 24, and this octet is anonymized equally. In addition, they share another octet, 0, which is also anonymized to the same value for both addresses. In this way the topology of the network is maintained, but permuted, by Crypto-PAn. The permutation needs a user-specified key, which makes it possible to reproduce the anonymization.

The tool yacryptopan [yac] was used to anonymize IPv6 addresses with Crypto-PAn, as the original Crypto-PAn is not expanded to anonymize these addresses.

4.4.2 Tcpmkpub

Tcpmkpub [tcpc] is the trace anonymization tool developed by Pang et al. [PAPL06]. It takes on the Link layer, Network layer and Transport layer, in addition to handling checksums. The work of interest here is the Network layer, as the Link layer is not considered, while the Transport layer and checksums are handled by other tools.

For IP Address anonymization, Tcpmkpub divides addresses into either being internal or external. This is in practice specified in a topology file where the user can set which networks are local to the network capturer and which subnets should be handled as internal. For external the approach is pretty straight forward: Crypto-PAn is used. Pang et al. argues that since Crypto-PAn is used only for external addresses, the lack of locality will make it problematic for an attacker to find one single person. There is, however, a more advanced policy for internal addresses.

For internal addresses, you avoid using prefixes which are found from the external address process. By making them separate, you prevent that by learning a known company's prefix, you can use that to find other networks which are close.

An IP address can be divided into the subnet part and the host part, where the subnet is a defined prefix of the address. For one subnet there can be multiple hosts. There is a different approach for the subnet and host portions of the internal addresses. Each defined subnet is mapped independently, and information is kept on whether two addresses are in the same subnet. An example is two addresses, 1.2.3.4 and 1.2.9.4, with the 24 leftmost bits as subnets, where the first 20 bits are a shared prefix. They may not share this prefix after the anonymization, as they are subnets handled independently. Then different mapping for host portions of a subnet is applied. For both the mapping of host portions and subnets, a pseudo-random permutation between addresses is needed. The permutations are dependent on a cryptographic key, which makes it possible to use the mapping on multiple traces without storing the mapping.

Using prefixes for internal and external addresses independently come with benefits and disadvantages. The benefit is that you avoid leaking prefix-information based on known prefixes. A disadvantage is that the anonymization will not be consistent between traces. The prefixes used for the external anonymization will not be the same every time, thus the space for internal addresses will change accordingly. This is the major drawback of Tcpmkpub.

4.4.3 PktAnon

PktAnon [pkta] is a generic framework for profile-based traffic anonymization and is based on the work of Gamer et al. [GMS08]. They look at the whole stack of captured network traffic in the form of a PCAP file. Different anonymization techniques are included with PktAnon, which makes it possible to manipulate fields from different layers. PktAnon is used in this thesis as a way to anonymize IP addresses since many of the other fields in need of anonymization are handled appropriately by other techniques. Either way, if one is in need of alternative techniques for fields, PktAnon provides multiple proposals. PktAnon is also used for its *constant overwrite* method. The techniques are applied to the chosen fields as per a user-defined Extensible Markup Language (XML) file.

An important contribution PktAnon introduces is defensive transformation. Every field which is desired after the anonymization needs to be assigned an anonymization technique. The technique in this context can also mean to just leave the field open. In this way you have to make a conscious decision on every field which is needed in the sanitized log. You then avoid forgetting to assign a technique, which could lead to an involuntary loss of private information.

For IP address anonymization, PktAnon has several methods, but the most relevant method is called *AnonHashHmacSha1*[1]. It hashes the entire address with Hash-based Message Authentication Code (HMAC) Secure Hash Algorithm (SHA)-1 and a user-defined key. This is secure, in that only IP addresses that are identical will share information. You will not learn any information about other addresses if one anonymized address is mapped back to the original. This approach limits the usefulness of the data from a researcher's point of view. Both IPv4 and IPv6 addresses are tested with *AnonHashHmacSha1*.

The method *constant overwrite* overwrites the value for the field with a user-specified constant. It is applied to the DSCP/ECN, Traffic Class and ToS fields with the value "0".

4.4.4 AnonTool

AnonTool [ano] is a command line tool for anonymizing network traces. Foukarakis et al. used the preexisting AAPI [KAA+06] to build this tool in 2009 [FAP09]. It comes with a variety of different ways to handle headers of different protocols and logs, such as HTTP, NetFlow, IP, TCP and UDP. Based on the evaluation done in Chapter 3, AnonTool is only used for IP addresses[2].

For IP addresses a method called *MAP* is used. An address is mapped to an integer. What this means in practice is that the first IP address arriving will get the value "1.0.0.1", the next one "1.0.0.2", and so on. The mapping is done such that both source and destination address is considered in the same map. The first source address will get "1.0.0.1", the first destination address "1.0.0.2". The same IP address will get the same map value. This means that by applying this anonymization technique, you will only recognize when the addresses are equal. Else, if the values are close to one another, you know that they were sent almost at the same time.

Looking at the *MAP* method from a researcher's perspective, this will provide less interesting information that some of the other techniques mentioned. The topology of the network is practically lost. But as it is a radical different approach than the others, it is worth testing if the privacy is better maintained in this scenario.

[1]PktAnon is also able to use prefix-preserving anonymization, but this is covered by Crypto-PAn.
[2]AnonTool is as of this writing only capable of handling IPv4 headers.

4.4.5 SCRUB-tcpdump

SCRUB-tcpmkpub [scr] is a network packet trace anonymization tool developed by Yurcik et al. in [YWH$^+$07a] and [YWH$^+$07b]. It implements several useful methods that can anonymize relevant data fields. The methods used for this thesis are presented below.

SCRUB-tcpdump has a method called *grouping* which groups a value into smaller partitions. Each of these partitions will be represented by one value. This method is used for Sequence/Acknowledgement Number, dividing the values into four groups. The groups are made from the following ranges:

$$[0 - 2^{10}], [2^{10} + 1 - 2^{20}], [2^{20} + 1 - 2^{30}], [2^{30} + 1 - 2^{32}]$$

Another useful method SCRUB-tcpdump provides is *bilateral classification*. The method is applicable for ports and Window Size from the relevant fields in this thesis. What it does is split the field values into either value A or value B. For the validation process bilateral classification is chosen as the anonymization technique for Window Size. The values for the Window Size below 10000 are set to 0, while 10000 are set to 65535.

The last method used from SCRUB-tcpdump is *keyed random permutation*, used on TCP Flags. It takes a user-defined key, which allows for the same random permutation between traces if the same key is applied.

4.4.6 Bilateral Classification

The idea of *bilateral classification* stems from SCRUB-tcpdump, covered in Section 4.4.5. SCRUB-tcpdump restricts the use of this technique to Ports and Window Size, while other use cases for it is implemented in this thesis. Both TTL and Hop Limit are handled in this manner. TTL and Hop Limit have field sizes of 8 bits, which gives 255 possible values. Values below 128 are classified as 0, and 128 and above as 255. This limits the usefulness of the fields for a researcher, but divides the possible default values for OSes evenly, based on [Lyo08, p. 185].

4.4.7 Grouping

Grouping in this context is expanded from the idea of SCRUB-tcpdump discussed in Section 4.4.5. For SCRUB-tcpdump the method is limited to Sequence/Acknowledgement Numbers, TCP Flags and TTL[3] related to our relevant fields. The method appears to be applicable to the Identification field as well, so an implementation for

[3]Although the TTL feature does not seem to work properly.

this field was done for the thesis. It divides the 16-bit space of the Identification field into eight equally large groups, where each group consists of 8192 values. The values between 0 and 8191 are classified as 8191, between 8192 and 16383 as 16383, and so on. All values above 57343 are grouped as the maximum value, 65535. The grouping prevents an OS fingerprinting attack, explained in Section 4.6.2, by not revealing how the Identification value is assigned and incremented for each packet from an OS. One drawback is that the usefulness of the data decreases.

4.4.8 Truncation

Truncation is a technique where you remove a part of a value, in our case some bits of a header field [SLL06]. Truncation can remove left-most bits, right-most bits, or even parts in the middle. All fields can be truncated, so the part of the evaluation is to observe which fields that have to use such a technique. For use in this thesis truncation of IP addresses has been used to measure the benefits of this approach to privacy compared to the prefix-preserving anonymization. The last octet of an IPv4 address has been set to '0'. For IPv6 addresses the last 16 bits are set to '0'.

4.4.9 Generalization

The term *Generalization* is defined in [BÅØ05] and makes values more general, less specific. By doing this it gets harder to distinguish the objects if their values are within each other's generalization criteria. It is a form of truncation, but since it has characteristics which only apply to ports in this implementation, it is chosen to be named its own technique in this thesis. The implementation is used exclusively for ports and is a novel contribution of this thesis.

The technique is based on how the ports are divided. Slagell et al. used this technique for ports as well [SLL05], but divided ports into either ephemeral (greater than or equal to 1024) or non-ephemeral (smaller than 1024). In doing this you lose much of the interesting data that ports provide. The ports from 0 to 1023 are defined as well-known system ports. They contain publicly acknowledged operations. Ports between 1024 and 49151 are known as registered user ports, with specific tasks distributed in advance. The remaining ports, however, from 49152 to 65535, are dynamic and/or private. This means that their purpose is not predefined, and so the use of any specific port in this range will effectively provide researchers with little information. Since any two ports in this range are virtually equivalent, they can be truncated in a way, called generalization. In the new implementation, every port in this range is rounded to the nearest hundred, such that it is impossible to read into how these ports are assigned for private use. The port number ranges are as defined by Internet Assigned Numbers Authority (IANA) in [ian].

4.4.10 Black Marker

Black marker, as explained in [SLL06], is a rather radical technique where the entire value of a field is replaced with a constant value, like a '-' character or a zero value, and thus removing every possibility of ever getting useful information out of the field. For the thesis, the black marker is already applied to Identification Protocol and Userid when the web server log is formatted.

4.4.11 Hashing

Hashing is a method where you replace a random length value with a fixed length value. The hash value is chosen such that by just inspecting it, you cannot get back to the original value. Hashing algorithms have three properties [RS04]:

Preimage-resistance - It is computationally infeasible to find x such that h(x) = y when knowing just y.

2nd-preimage resistance - It is computationally infeasible to find an x' after x is inputted, where x' is unlike x, but h(x) = h(x').

Collision resistance - It is computationally infeasible that x and x' hash to the same output when they are unlike, meaning that h(x) = h(x').

Hashing in the context of this thesis is done for three fields: Request from a web server, and Hostname and Message from a syslog. The Request field is most commonly a URL, which will be used as the term for the Request field further in this explanation of hashing. When a Request is built like a string i.e. not a URL, it is split on spaces, where each component after the split is hashed. The Hostname is hashed directly. The syslog messages are split based on spaces and every component is hashed. URLs on the other hand, are handled differently. The approach is reminiscent of that from Kuenning and Miller in [KM03]. The method chosen here is always splitting on '/', '?', '=' and '&', such that also queries (e.g. in a search) are handled securely. Every component after this split is hashed. The hashing is performed by the algorithm Password-Based Key Derivation Function 2 (PBKDF2) [pbk] with HMAC-SHA256 as a pseudorandom function. This sets the output to 32 octets (which are 256 bits). For PBKDF2 you input a string, set a salt and an iteration count. Based on the pseudorandom function underneath (here HMAC-SHA256), the PBKDF2 function is performed, where octets are XORed together. How many octets are XORed is defined by the iteration count. These XORed octets are then outputted as the number of octets defined by HMAC-SHA256.

This algorithm, compared to other hash algorithms, has one advantage: It is designed to be slow, based on the iteration count. The speed at which an attacker can test words in a hash algorithm is crucial for the system to not reveal the hashed data, and so this property is highly wanted for our approach.

When this hashing is applied to the data, it is done deterministically. Every matching string is hashed the same way, so as to provide researchers with data about which terms appear several times. The key to this is one predetermined salt. For the data that is hashed to still be useful, you have to be able to recognize when the same term reappears. However, with the deterministic approach comes a large risk. As will be explored more in Section 4.6.1, an attacker is able to exploit such an approach. By injecting data in a time frame that he/she is about to receive data from, the attacker can create a dictionary of values of his/her choosing and their anonymized version. Certain parameters need to be in place for the deterministic hashing to work, and they are discussed in Section 4.6.1.

URLs can sometimes contain an IP address. Since the address space of IP addresses is relatively small, a hash function is not secure to handle IP address anonymization. However, in the setting of a URL, hashing will work. The reason is that the space from which values are chosen is vastly larger, since values can take on several forms, like email-addresses or, in fact, all strings imaginable. The complete IP addresses will in this case be hashed, not based on octets, which is the case for other prefix-preserving methods.

By using this hashing algorithm, you could "hide" the original search words and directories of web pages in a large set of bits, thus making it impossible to go from the hashed part back to the original part, based on *preimage-resistance*. In here lies a challenge with regards to the GDPR. Since you should be able to reverse the pseudonymization you performed on the data, this approach seems to fall outside of our use case. However, by the approach described in Section 4.3, you get a one-to-one mapping between the original and the anonymized packets, bypassing the issue of hashing.

4.5 Comparing Anonymization Techniques

The anonymization techniques that are being utilized in this thesis have now been presented. To better understand how they provide anonymity, a comparison between them is needed. Only IP address anonymization is considered for comparison. Crypto-PAn, Tcpmkpub, PktAnon, AnonTool and truncation are compared.

Crypto-PAn preserves the topology of the network in a strict manner and needs a user-defined key to preserve the same mapping between logs. An octet is mapped to the same anonymized octet, no matter at which position in the address it resides. A deanonymization for this approach will expose all the octets of the address, which will be similar through the whole log. This is the disadvantage of Crypto-PAn.

Tcpmkpub tries to improve on Crypto-PAn by dividing into external and internal addresses. Crypto-PAn is used for external addresses, while the internal addresses are handled per defined subnet specified by the user. Each subnet is permuted pseudo-randomly based on a user-defined key. This leaves more security for internal addresses, which are more likely to be exposed by an attacker for known companies. Exposing one internal address will not lead to knowledge gained on the rest of the internal network, which is an issue with Crypto-PAn. Similarly, by gaining knowledge of external network, you learn nothing about the internal addresses.

PktAnon uses hashing of the complete value with HMAC SHA1. IP addresses will only be comparable if they match completely. The method needs a user-defined key to keep the hashing consistent between logs.

The AnonTool method has similarities with the method from PktAnon in that you gain no information about another address if you expose one. It is a mapping performed based on the sequence the addresses appeared in. The topology of the network is obfuscated with this method.

When observing their properties, it is thought that truncation of IP addresses will yield the most secure anonymization. In this way you disallow the researchers of seeing a complete address, in effect removing the octet which most easily distinguish users. The method is for this reason not providing researchers with specific data. However, the network topology is still maintained, which cannot be said for the methods of PktAnon and AnonTool, and so truncation add data with different characteristics.

4.6 Attacks on Privacy

There are several attacks that applies to log data, but the focus of this thesis is on injection attack and OS fingerprinting attack, as they seem most appropriate to the logs currently investigated. A wide variety of attacks is discussed by King et al. in [KLS09].

4.6.1 Injection Attack

Injection attack[4] is where an adversary injects packets with known characteristics into a network to try to recognize his/her own patterns in network data which has been anonymized [BST+10]. By doing this the adversary can build up a dictionary of words that he/she knows the anonymized version of, risking the privacy of the

[4]Reminiscent of a known-plaintext attack for cryptosystems [BST+10].

whole anonymization process. This attack is fairly difficult to defend against. Since the anonymization needs to be deterministic for researchers to get any effective data out, the adversary is able to recognize patterns. It is also difficult to spot that the person or computer adding this traffic to the network is in fact malicious. Since prefix-preserving anonymization through Crypto-PAn and partly Tcpmkpub is applied to IP addresses for the fields in the thesis analysis, the adversary can obtain knowledge as to which prefixes translates to which anonymized prefixes. This weakness is discussed further in Section 8.4.2.

Companies providing network logs to researchers need to be aware of such an attack. The information shared with the researchers regarding the particular log need to be limited. The company should not share which network the log was captured on. Likewise, researchers should not be able to request specific networks in fear of them having an injection attack prepared on their requested network. The time in which the traffic log is captured should not be shared in advance, but be randomly selected. As stated in the GDPR Article 32(1) [gdpg], it should be highly unlikely that the anonymization approach is reversed by the wrong people with equipment and technology of today. When the researcher knows neither the time nor the place, the work he/she has to put in to be able to inject the correct captured log is very large.

In addition to these practical suggestions, Brekne et al. discussed countermeasures for injection attacks in [BÅ05], which goes beyond this master thesis. The countermeasures are to employ non-static pseudonyms for IP addresses, employ mandatory sampling at the monitoring sensors, which effectively increases cost of performing injection attacks, and detecting and preventing packet injection attempts by removing malformed packets.

4.6.2 OS Fingerprinting

When an adversary is analyzing the anonymized and original log, as in the case of this thesis' validation process (see Section 6.4.3), not only injection attack is possible. The adversary can also try to fingerprint the OS of a user. As mentioned in Section 3, multiple data fields in the different logs can be used in OS fingerprinting. King et al. [KLS09] explains fingerprinting as "the process of matching attributes of an anonymized object against attributes of a known object to discover a mapping between anonymized and unanonymized objects.". This means that the adversary recognizes certain values in a field which would only stem from a particular OS. By doing this, the adversary is able to shorten the list of Unanonymized Object (UO)s which can map to the Anonymized Object (AO) in question. This mapping is explained in Section 6.4.3. In the rare case that an obscure OS is recognized, it would also be easy to single out this computer and the person operating it.

Several of the fields need anonymization based on this threat, even though they do not directly contain any information which is deemed personal or sensitive. The methods used for these types of fields can be found throughout Section 4.4.

What is explained above might seem like a farfetched scenario. The chances of identifying a single person based on the OS are small. However, from a security perspective worst-case scenarios need to be considered, and one has to consider the situation where the adversary has some information regarding certain fields beforehand. If the adversary is after a particular person, and already is in possession of the OS, this can be used. When situations call for it, every opportunity might be exploited, and it is better to be overprotective than to lose valuable private data.

Chapter 5 Methodology

The methodology used for the thesis is based on both qualitative and quantitative methods. A qualitative method comments on the suitability of a solution, while a quantitative method utilizes measurements to either recommend or discourage a solution.

The general approach for the master thesis is best described as using design science. Design science is a framework which fits the approach of a scientific experiment because it emphasizes iterations of smaller subtasks, which may or may not provide suitable results [Wie14]. If the results fail to meet expectations, then another iteration is performed, and the knowledge gained from the results is used to get a better understanding of the problem and what to do next time. During a scientific experiment, new knowledge is the desired goal. Based on research questions for such an experiment the results will tell if the answers to these questions are satisfactory. During this process, emphasis should also be put on how the results are provided, as there is never just one single possible approach for a problem. Design science introduces several terms which make it easier to define the progression of the process, where it will end up and how to get there. An iterative approach for both the complete process and subtasks will make evaluation in relation to the end goal more feasible.

In design science, a goal is that artifacts give new knowledge, which can be used to create solutions to challenges, in the context of the artifact. An artifact can be anything human made, intended in some way to solve a task. The context can e.g. be how the artifact is used, like a design.

Figure 5.1 shows the general approach for the methodology used for the process in Chapter 6, with inspiration for the figure from [OLSB09]. A problem is identified after a literature research, and an artifact is created in the context of the problem. Another literature research is needed to find the best way to evaluate the artifact. Then the artifact is evaluated. The evaluation results in new knowledge of the problem, which in this thesis would be the significance of an anonymization technique. The process might be iterated based on other available solutions and the adequacy of the technique. Subtasks from the artifact include which fields to use in a log.

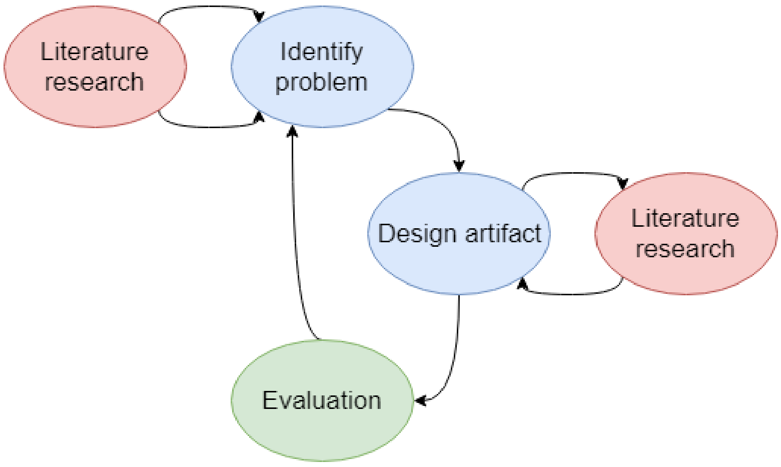


Figure 5.1: The general methodology for the thesis.

Chapter 6 Validation of Anonymization

This chapter presents the complete process for validating an anonymization technique in this thesis. First, an overview of the initial approach is described. A large part of the thesis work has consisted of an implementation process, and this is addressed next in this chapter.

The focus then shifts to the actual validation process. After anonymization is performed on the logs, a method to validate how good the anonymization is, is needed. What is interesting to calculate, is how easy it is to single out a person in the anonymized log. Coull et al. explained a sensible method in [CWK$^+$08]. Theory is presented to understand the mathematical procedure for the validation, before the testing approach will describe the validation process.

The testing approach in Section 6.4 addresses the whole process for validation of an anonymization technique on a log. The section initially explains the log decision process and a general approach for the logs. Then the validation of logs is presented, before an explanation of the deanonymization/mapping process between an anonymized log and an original log ends the chapter in Section 6.4.4.

6.1 Initial Approach

The work is based on studying GDPR Articles to understand what is needed to succeed with proper anonymization. By literary review, it became clear that finding anonymization techniques which could change the personal data, while still being able to reverse it, is wanted. The literary review was a large part of the first couple of months, as an understanding of the GDPR, anonymization techniques and validation method was required.

The concept of the thesis is to manipulate personal data so that researchers can get useful information out of a network traffic log, while the company providing the data complies with the GDPR. In many of the logs available for further inspection, IP addresses play a central role. In addition, IP address are the most personal data sensitive field, bar Request, Identification Protocol and Userid from web server logs, and Hostname and Message from syslogs. The initial step was to investigate the possibilities for IP addresses. Then followed research into evaluation of the other personal data sensitive fields. Special care was put into URL and Port evaluation. The appropriate solutions are covered in Section 4.4.

6.2 Implementation Process

In this section, the implementation process for the thesis is explained. First, code for incorporating the theory from Section 6.3 was implemented. The next step was how the data should be handled from the captured files. A general idea from the start was that Internet/Transport layer logs, with IPv4-, IPv6-, TCP- and UDP-header logs, were captured in PCAP files. Crypto-PAn (Section 4.4.1) needs a text file as input to perform the anonymization. The challenge was then how to fit this PCAP data into both the anonymization techniques, like Crypto-PAn, and the validation method. A PCAP parser called *pkts.io* [pktb] provided the solution. By using *pkts.io*, java code was produced which splits each recognizable header field into a new array, making manipulation of the data much simpler. Some issues with *pkts.io* is discussed in Section 6.2.1.

After formatting the log files, the whole validation process from [CWK$^+$08] described in Section 6.4.3, was implemented. This implementation was offered several challenged, as when one implemented function worked, another provided a problem. For this implementation to work correctly, an iterative process for checking every step of the validation implementation was performed with Crypto-PAn as a test anonymization technique.

While Coull et al.'s [CWK$^+$08] validation process was being implemented, codes for formatting NetFlow logs, web server logs and syslogs were developed. The logs were formatted to include the fields described in Chapter 3. Each of these three logs were obtained as a text file, and the output of the formatting was a text file as well. The formatting mostly consisted of formatting IP addresses and timestamps so they could be compared, and removing fields that were deemed out of scope for the chosen formats[1]. In addition, the novel contribution of *generalization*, as well as the code for this thesis' interpretation of *hashing*, *truncation*, *grouping* and *bilateral classification*, was implemented in this phase. Finally, the anonymization frameworks mentioned in Section 4.4 were configured to run correctly. The successful implementation of both

[1]These logs, like web server log or NetFlow log, have many possible formats with other fields than the ones focused on in Chapter 3. The most common formats were chosen, and their respective fields were covered in Chapter 3

the validation process and the mentioned anonymization techniques and frameworks allowed for the results to be produced.

6.2.1 Parsing of PCAP Files

As mentioned, parsing PCAP files with *pkts.io* [pktb] allows for these files to fit the layout of the anonymization techniques, that is, a text file with one line being one packet, and fields of the packet separated by a tabulator. A lot of functionality is already implemented in the *pkts.io* tool. For many fields in IPv4-, IPv6-, TCP- and UDP-headers, the methods are easy to utilize without further work with the code. However, there are also some fields missing. The provided code for *pkts.io* made it possible to add methods to get fields with some tweaking of the code.

The fields added to the *pkts.io* version of IPv4 are *DSCP*, *ECN* and *TTL*. *Traffic Class*, *Flow Label* and *Hop Limit* are added to the *pkts.io* version of IPv6. The fields added to the *pkts.io* version of TCP are *Reserved*, the *NS flag*, *Windows Size*, *Checksum* and *Urgent Pointer*. TheUDP *Checksum* is also added. The additional code is supplied in pkts-core/src/main/java/io.pkts.packet/ for the following interfaces: IPv4Packet.java, IPv6Packet.java, TCPPacket.java, UDPPacket.java. The classes that inherited the methods of these interfaces, and thus also had code added, were IPv4PacketImpl.java, IPv6PacketImpl.java, TcpPacketImpl.java and UdpPacketImpl.java from the directory pkts-core/src/main/java/io.pkts.packet.impl/.

To get these changes to work, the project *pkt-core*, where all the changes are performed, needs to be exported as a Java ARchive (JAR) file. This JAR file must be added to the repository from which *pkts.io* is cloned, and added as a dependency in the Maven project containing the configuration of the PCAP formatting program selected for the thesis. This PCAP formatting program formats a PCAP file to include the fields specified in Chapter 3 for IPv4-, IPv6-, TCP- and UDP-headers

6.3 Theory

This section is heavily dependent upon mathematical theory and formulas. Specifically, four equations are essential for the validation process: Entropy, mutual information, normalized mutual information and L1-similarity. Each of the formulas will first be described in this chapter, before they are applied in the validation process in Section 6.4.3. The term *object* is important for the next subsections and the validation process in Section 6.4.3. An object is a collection of log records/lines where certain fields have identical value. Some fields may also have deterministic values. How objects are created is explained thoroughly at the start Section 6.4.3.

6.3.1 Entropy

Entropy is an essential part of measuring the anonymity of a log, which Coull et al. explains in [CWK+08]. Entropy is a value for how evenly distributed the data in a log is. Serjantov and Danezis suggest in [SD02] that the entropy value can be seen as the number of bits of additional information needed by an attacker to identify a person. The equation for entropy is

$$H(X) = -\sum_{x \in X} p(x) \lg p(x), \tag{6.1}$$

where lg is lg_2, and *p(x)* is the probability of observing value x for random variable X. The probability of an instance is used to produce the entropy value. From this follows that the probability for every value in every field for each object is calculated (Objects are discussed more in Section 6.4.3). This is done to obtain the entropy value used further in Section 6.4.3 to effectively deanonymize a log.

Consider an example were we look at an IPv4+TCP/UDP log as explained in Section 3. Say that the log has ten packets with Source IP Address *1.2.3.4*. These packets constitute an object, called object A. X is the random variable for source port values of A. Between the ten packets of object A, seven packets have 80 as the source port, two have 443 and one have 115. Hence when observing a source port of object A randomly the probability of seeing port 80 is p(80)=0.7, port 443 is p(443)=0.2 and port 115 is p(115)=0.1. The entropy calculated for the source port field of object A is thus 1.15677965 with Equation (6.1). *log N* is the maximum entropy for a variable, where N is the number of values investigated, in this example 10. This means that the maximum entropy for this example would be 3.32192809, indicating that the field is not evenly distributed, but not completely dominated by one value either. The minimum value for entropy is 0. This occurs when *p(x)* is 1 for a certain value of x, and 0 for all others.

6.3.2 Mutual and Normalized Mutual Information

The next equation is mutual information. Mutual information is used in this process to see if fields are dependent of each other. As explained in [CWK+08], a comparison directly between original and anonymized data will not give a good result when the anonymization modifies the value of the original to the point where you no longer can see any similarity. For this a method of recognizing fields that statistically are almost identical is needed. In this context the mutual information equation is used.

The equation is as follows:

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \lg \frac{p(x,y)}{p(x)p(y)}. \tag{6.2}$$

Equation (6.2) will find to what extent X and Y are independent of each other. To get a more intuitive measure of this number, the normalized value of the mutual information is calculated, like this:

$$\overline{I(X;Y)} = \frac{I(X;Y)}{min(H(X),H(Y))}, \tag{6.3}$$

where *H(X)* and *H(Y)* is the entropy of variable X and variable Y, respectively. This value will be used between fields of an object to evaluate if two fields can be grouped together based on them being almost identical. This method is called Feature Selection and will be explained in more in Section 6.4.3.

6.3.3 L1-Similarity

L1-Similarity, based on [CWK+08], is computed as Equation (6.4) shows. This is done to find how similar the variables X and Y are. You look at the values for both variables and find the probability for every value in both X and Y.

$$sim(X,Y) = 2 - \sum_{z \in X \cup Y} |P(X = z) - P(Y = z)| \tag{6.4}$$

When the two variables are the same, the probabilities will match up, and the similarity will end up as 2. If they have no similarities, all probabilities for the values in X will add up to 1, and the same will happen for the values in Y. The summation of the equation will then be 2, and the L1-similarity is subsequently 0. In Section 6.4.3 this is used to find the similarity between the fields of an AO and a UO.

6.4 Testing Approach

The rest of this chapter is devoted to testing the different anonymization technique combinations with the validation process of Coull et al. [CWK+08]. The complete process of the testing approach is shown in Figure 6.1. Aside from the step of running anonymization techniques, every step of the process in this figure is a program implemented for this thesis. They are used to format, anonymize and validate fields, logs and anonymization techniques. The AddingRecords and FeatureSelection programs use the method explained in Section 6.4.3, while the Validation program is a combination of Section 6.4.3 and 6.4.4.

How the testing of the logs is performed after the implementation of the validation code is now considered. What is worth noting here is that the whole testing approach is done from an adversary perspective. In this scenario, the adversary has access to both the anonymized and the original log, and wants to perform a mapping (deanonymization) to see which anonymized data belongs to which unanonymized person. This scenario is of course exaggerated, but a lot of information on different values for packets are known, and by knowing a small amount, the adversary could in theory perform an attack like the testing approach describes. This includes the scenario described in Section 4.6.2 where an attacker knows the OS of the victim, which decreases the number of possible mappings.

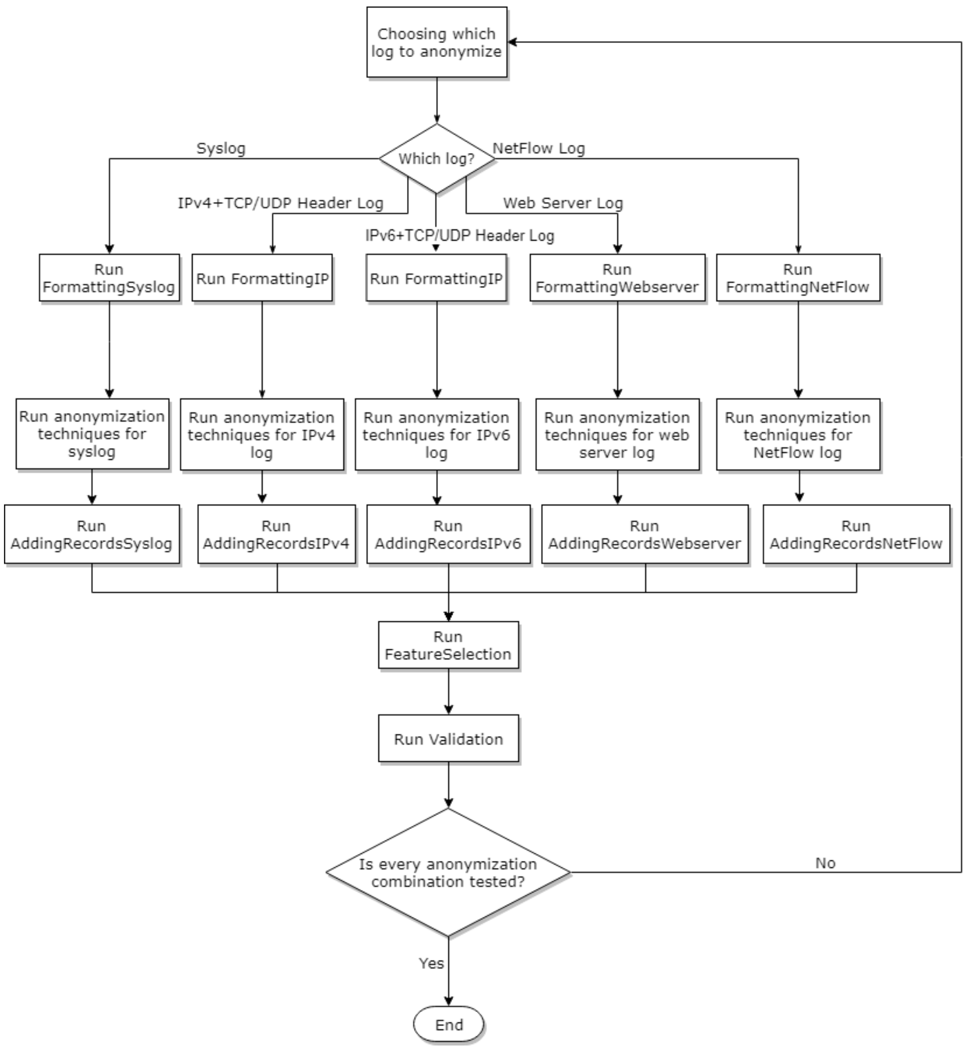


Figure 6.1: A process flow diagram.

6.4.1 Log Decision Process

The initial step is to select the type of the log. The type of log is decided between IPv4+TCP/UDP header log, IPv6+TCP/UDP header log, NetFlow log, web server log and syslog. As explained previously, different logs contain different fields with information. Which fields are being investigated decides which techniques are used. The way to anonymize each log is detailed in Section 3 with the anonymization techniques from Section 4.4. Some fields will have several techniques that are applicable, making it possible to test different combinations. Many fields will not require anonymization, however, and are left in their original state.

6.4.2 General Approach for Logs

The log fields for the different logs were studied closely in Section 3. Based on the fields in each log, different approaches were needed to anonymize the data. This section will describe the general approach for all logs, in addition to the evaluation of each field given in Section 3.

Note that Internet/Transport layer logs are divided into logs containing IPv4+TCP/UDP data and IPv6+TCP/UDP data. This is because a joint log for both IPv4 and IPv6 would require multiple fields with no values as all the packets would need all fields, regardless of IPv4 or IPv6. The validation process needs all fields of the log to work properly, and several fields with null value would dramatically influence the Feature Selection process, which is described in Section 6.4.3.

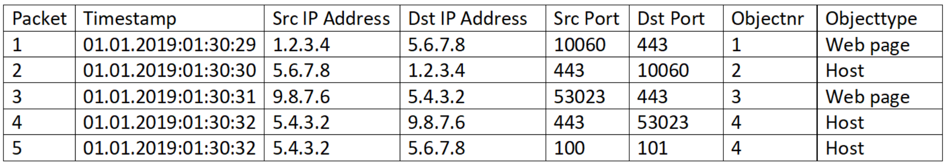
In the logs, there are a few fields that with certainty can contain personal information, where anonymization is a must: IP address, Request (URL), Identification Protocol, Userid, Hostname and Message. As mentioned, there are plenty of other fields in risk of OS fingerprinting, and they might also be in need of anonymization. The approach for the anonymization, regardless of the log, is as follows: If the log contains one of the must-anonymize fields, the different alternatives for them will be compared to start with, e.g. IP addresses. Then, if the log contains fields at risk of OS fingerprinting, anonymization will be performed in two ways:

1. All of these fields will be put together with one of the anonymization techniques for the must-anonymize fields. An example is that these fields are anonymized together with Tcpmkpub for IP addresses and generalization for ports.
2. Each one of the OS fingerprinting fields will be individually anonymized together with the must-anonymize fields, to spot which OS fingerprinting anonymization is impacting the results the most. An example is that Identification is anonymized with grouping in the same logs that uses Tcpmkpub anonymization for IP addresses and generalization for ports.

6.4.3 Validation Process

When the anonymization of the appropriate fields of the log is done, the validation of the particular combination is needed. This means to check which combinations hide the personal data in the best way, based on entropy explained in Section 6.3.1. The validation process explained here is the validation process of Coull et al. [CWK$^+$08] interpreted and implemented for this thesis.

The validation has several steps. First the log is split into different objects. Based on the article Coull et al., the idea is to group the objects as either hosts or web pages. From the information available from the logs, it is possible to group objects on IP addresses and ports. A check will be made to see if the packet is using ports 80 or 443 as the destination port. Port 80 and 443 are used for HTTP and Hypertext Transfer Protocol Secure (HTTPS) respectively, and thus indicate web page activity. Every unique IP address with destination port 80 or 443 is grouped as a web page. Then, every unique IP address which is not defined as a web page, is grouped as a host. The outcome is several objects, grouped as either hosts or web pages, and assigned a unique object number. With reference to figure 4.1, the original log from this figure is assigned object number and object type in Figure 6.2. Packet 4 and 5 are from the same Source IP Address, and thus grouped to the same object. Packets 1 and 3 have Destination Port 443, but different Source IP address, which groups them into separate web page objects. In the validation, hosts and web pages are separated, and results are provided independently, as they have different characteristics. Since an anonymized host cannot be an unanonymized web page, based on how they are grouped, a comparison between them is not needed.



Packet	Timestamp	Src IP Address	Dst IP Address	Src Port	Dst Port	Objectnr	Objecttype
1	01.01.2019:01:30:29	1.2.3.4	5.6.7.8	10060	443	1	Web page
2	01.01.2019:01:30:30	5.6.7.8	1.2.3.4	443	10060	2	Host
3	01.01.2019:01:30:31	9.8.7.6	5.4.3.2	53023	443	3	Web page
4	01.01.2019:01:30:32	5.4.3.2	9.8.7.6	443	53023	4	Host
5	01.01.2019:01:30:32	5.4.3.2	5.6.7.8	100	101	4	Host

Figure 6.2: Log with assigned object number and type.

The grouping is done to better compare unanonymized and anonymized logs. The goal for an attacker is to be able to single out one or several people, based on their IP address or a combination of other fields. By grouping you single out unique IP addresses and their activity. This makes it easier to map between an AO and a UO, and in this way recognize a user's behavior in the data log. Recall that this approach is from an adversary's point of view.

As a note, remember that syslogs contain neither IP addresses nor ports (Section 3.4). For these logs, unique App-names are used to split the log into objects.

Inter- and Intra-Records Process

By now all the packets of an object are grouped together based on the grouping characteristics of a host or a web page. When the grouping into objects is done, additional information for each object is calculated. For every field of a packet in an object, a comparison is calculated between the field value for this packet and the previous packet within the object. The comparison depends on the field type. For example, the IP address of packet *i* is compared to the address of packet *i-1* with XOR-operation, while timestamps are compared with minus operation. A summary of how the fields for every log are compared is provided in Appendix C. The calculated results are then added to the packet record as new fields. The fields added from this comparison are called inter-record fields [CWK$^+$08].

Next intra-records are added if the log requires it. An intra-record is a comparison between fields internally for a packet. The comparison is only carried out for fields that make sense comparing, that is, they have the same data type. For example an Internet/Transport layer log contains both Source and Destination IP Address, as well as the IP address fields provided from inter-records. These fields are compared and added to the packet records as new fields.

Inter- and intra-records are added to be able to differentiate objects from one another more clearly. The implementation of intra-records is based on logs that originally contain more than one field which can be compared[2]. Every packet will have two of every field after adding inter-records, but to compare the original and inter-record fields for every pair seems excessive. Intra-records for IP addresses and ports are added for IPv4+TCP/UDP and IPv6+TCP/UDP header logs since they originally include source and destination fields. For a NetFlow log, intra-records for IP addresses, ports, AS numbers and interfaces are added to a packet. Web server logs and syslogs do not originally contain any pair of fields that makes sense to compare.

In the coded implementation of this process, the input is two text files - one with the original log and another with the anonymized log. the output is the two same text files with object numbers, object types, inter-records and intra-records added.

Feature Selection Process

In the Feature Selection process, you compare the fields of an object. The process is explained in [CWK$^+$08]. This is to minimize the actual validation of the objects, explained further below in this section in the Object Anonymity process, since it lets you compare fields that are independent of each other.

You first compare two and two fields. Fields are grouped together if they are dependent on each other more than some threshold. This dependency is calculated with normalized mutual information from Equation (6.3), explained in Section 6.3.2.

[2]It does not make sense to compare a port value and the value for TCP flags, as the value you are left with does not indicate a natural relation between the two values.

The threshold set for this implementation is 0.99, as this strongly suggests dependency among the fields. If a field is grouped into more than one group, the shared groups are further grouped together. The fields that does not reach the threshold for any combinations are left by themselves. In this way you end up with groups that are independent of each other. These groups are now called features.

There is ambiguity in [CWK$^+$08] as to whether an object or the entire log is checked for Feature Selection. As objects are compared with one another in the end, it seems reasonable to perform the Feature Selection on an object basis.

In the coded implementation of this process, the type of log to be used is first specified. Then the input is the original log and the anonymized log produced from the Inter- and Intra-Records process, and the output is the same two text files with Feature Selection performed on their fields. This effectively groups all fields into independent features.

Object Anonymity Process

By now the Feature Selection process is completed, and the fields of the packets are grouped into features, for both the original and the anonymized log. The most comprehensive process will be explained here. The method is still as explained in [CWK$^+$08].

The first step is to look at the L1-similarity, from Section 6.3.3, for the first feature of an AO. You compare an AO and every UO by pair based on their L1-similarity for this feature. This is eventually done for every feature of the AO. For this AO, a probability is calculated from the total sum of L1-similarity for all UOs in this feature. The UO with the highest probability will be the UO which is most similar to the AO for this feature. Equation (6.5) shows how this calculation is performed.

$$P(X_{i,A} = U_j) = \frac{sim(F_{i,A}, F_{i,U_j})}{\sum_{\forall U_k} sim(F_{i,A}, F_{i,U_k})}, \tag{6.5}$$

where $F_{i,A}$ is the ith feature of AO A, F_{i,U_j} is the ith feature of UO U_j. $sim(F_{i,A}, F_{i,U_j})$ is the L1-similarity from Equation (6.4) for $F_{i,A}$ and F_{i,U_j}. The denominator sums the L1-similarity for A and all UOs for feature F_i. $P(x_{i,A} = U_j)$ constitutes the probability for feature i for the combination of A and U_j. A probability is calculated for the combination of A and each U_j. After the probability is computed for every A and U_j combination for one feature, the entropy of the feature for A is calculated. The entropy is computed from Equation (6.1) in Section 6.3.1. What you end up with, is the entropy value for this feature of the AO. The value will indicate if there is an even distribution of probabilities (i.e. there are no values that stand out; high entropy) or if there is a peak in the probabilities (i.e. one value dominates; entropy close to zero). This process is continued for every feature of an AO.

When all feature entropy values have been computed for an AO, the feature entropies are added together. This constitutes the total entropy for the AO, and is shown in Equation (6.6). The described process is continued for every AO in the log.

$$H(A) = \sum_{i=1}^{l} H(X_{i,A}), \tag{6.6}$$

where l is the number of features for AO A, *H(A)* is the entropy for A, and $H(X_{i,A})$ is the entropy calculated for feature i of A.

After all the AOs have its entropy computed, the average entropy for the AOs is determined. In addition, the maximum entropy for the log is calculated. That is done by finding the AO with the highest number of features, and calculating $l \lg N$, where l is number of features and N is the total number of AOs. This assures that the maximum amount of entropy possible for the log is shown. The average entropy and maximum entropy form the foundation of the results in Chapter 7.

6.4.4 Mapping Process

A mapping process is applied in the Article of Coull et al. [CWK+08] which enables an adversary to match AOs with UOs. As it is not directly specified in [CWK+08] how the mapping of an AO and a UO is performed, an interpretation of the process is applied in this thesis.

The mapping between an AO and a UO is done in several steps. The first step is to calculate the entropy for every AO in the log. This process is explained in Object Anonymity Process in Section 6.4.3. You end up with the entropy value for all AOs. You select the AO with the lowest value. This AO is comprised of the entropy values of the features which the AO consist of. You select the feature of the AO with the lowest entropy value. This feature is chosen because of the probabilities of all the UOs. The UO with the highest probability in this field is chosen as the correct mapped object. A check is done to see if the AO in fact matches the UO. If not, it is counted as a mismapping. The final step of an iteration is to remove both the AO and the UO from the objects to be mapped, and store the mismapping value. This process is repeated until there are no more AOs and UOs to map.

In the coded implementation of this process, the input is the two text files produced from the Feature Selection process, and the output is a csv file with the results. A pseudoalgorithm is provided below, where X is the AO with lowest entropy, F the field with lowest entropy, and Y the UO with highest probability.

```
$misMapping \leftarrow 0$
while $anonymizedObjectList$ is not empty do
  Calculate entropy value for every $AO$
  $X \leftarrow 0$
  $F \leftarrow 0$
  $Y \leftarrow 0$
  for every $AO$ do
    if $AO$ is min then
      $X \leftarrow AO$
    end if
  end for
  for every $field$ in $X$ do
    if $field$ is min then
      $F \leftarrow field$
    end if
  end for
  for every $UO$ in $F$ do
    if $UO$ is max then
      $Y \leftarrow UO$
    end if
  end for
  if $X \neq Y$ then
    $misMapping \leftarrow i + 1$
  end if
  Add $misMapping$ to $misMappingList$
  Remove $X$ from $anonymizedObjectList$
  Remove $Y$ from $unanonymizedObjectList$
end while
```

What follows is an example of one iteration of the process with fictitious objects, features and values. It illustrates the mappings process clearly.

1. First the AO with the lowest entropy is chosen. As can be seen in Figure 6.3, AO2 has the lowest entropy and will be investigated further.

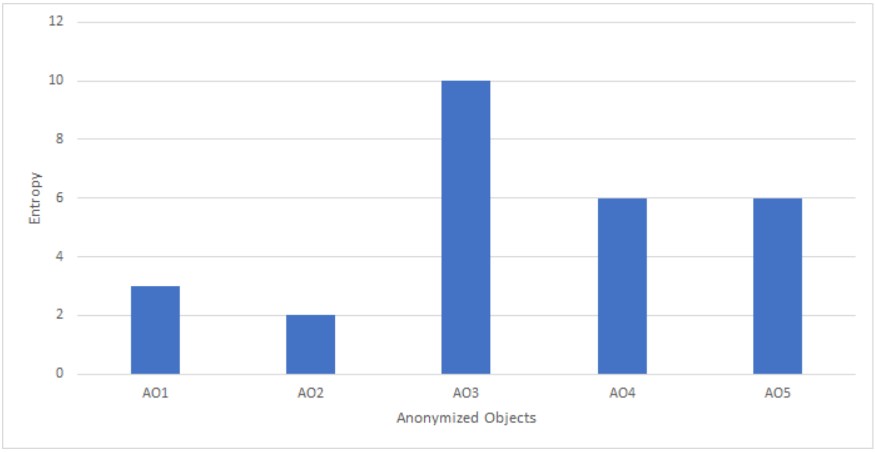


Figure 6.3: Anonymized object selection.

2. When AO2 is chosen as the AO with lowest entropy, the entropy of all the features of this object is considered. The feature with the lowest value is chosen, which in Figure 6.4 is F1.

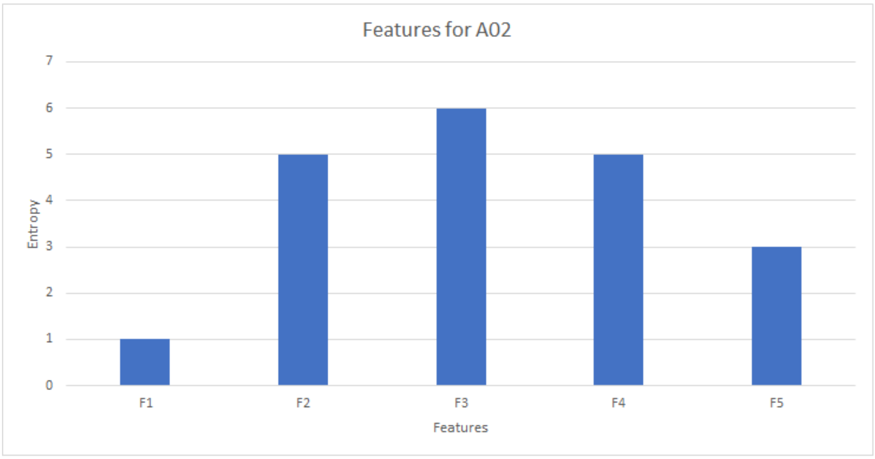


Figure 6.4: Log feature selection.

3. Now that F1 is chosen as the lowest feature, the probability distribution that F1 is based on is considered, and the goal is to find the UO with the highest probability. In Figure 6.5, it is shown that UO2 has the highest probability. This means that AO2 is mapped to UO2 by F1. In the end the two mapped objects will

be compared to see if they match. This is to find the number of mismappings. If they do not match, then the mismapping number is increased.

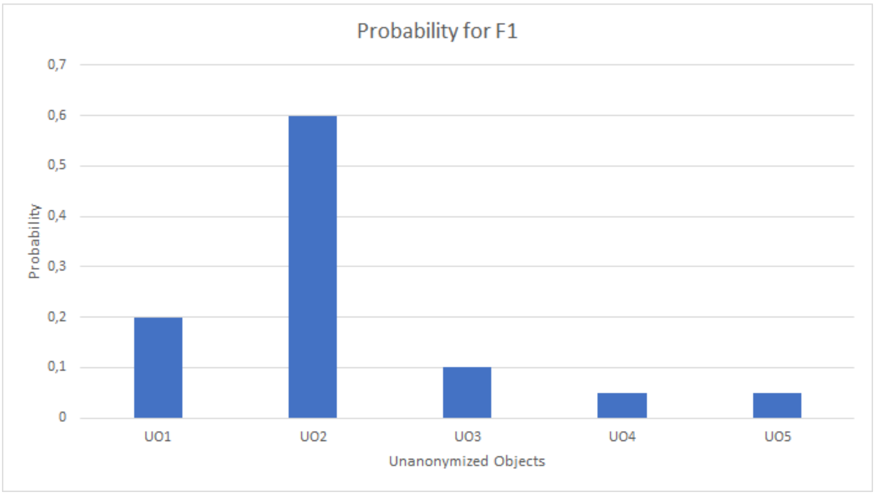


Figure 6.5: Unanonymized object selection

Chapter 7
Results

The methodology of the master thesis and the validation process are now explained, and this chapter will present the results from the performed work. In the mapping process of a log, three values are measured for every mapping between an AO and an UO: The average entropy before the mapping, the maximum entropy before the mapping, and the amount of mismappings after the mapping. This chapter provides results in the form of a percentage from the average entropy before the first mapping normalized by max entropy before the first mapping. The complete mapping process for every log and object type can be found in Appendix A. This will display how the average and maximum entropy decreases as the mapping progresses, and optimally you will see different graphs for different anonymization technique combinations. The complete mapping process line charts are further discussed in 8.4.2. Unless otherwise stated, the techniques from the log evaluation in Chapter 3 are applied, validated and providing the results for each log.

By looking at the results in Appendix 7.1, it can be seen that the average entropy of an AO is approximately 400 at the start of the mapping process for an IPv4+TCP/UDP header log. This number is a measure of how many bits an attacker has to brute-force to guess the correct combination of the information inside the object, as suggested by Serjantov and Danezis in [SD02]. Parts of this information is public knowledge or the space in which values can come from is small (true/false). Not all of the 400 bits would be a complete guess, but it gives a quantitative number as to how hard it is, from an adversary's point of view, to obtain the information.

When anonymization is applied to a log, the entropy should be higher as the ability to guess the combination in the log should decrease. You would think that the entropy with no anonymization should be zero. This is not the case for this validation process. The explanation is that multiple objects have the same values in fields, which makes the entropy for the field more than zero. As explained in Section 6.4.3, the average entropy number is measured by calculating the entropy for each field of an object, adding these field entropies together, and then finding the average among all objects. The entropy of one field is based on the probability distribution of possible objects that can map to the values of our object in question. If there are more than one object with the same values as our object, the field entropy will be more than zero, and the average entropy for all objects will thus also be more than zero.

7.1 IPv4+TCP/UDP header Log

Execution of the tests for IPv4+TCP/UDP log was done on 693 packets with 92 hosts and 39 web pages. Figure 7.1 shows the average entropy and maximum entropy as AOs are mapped to UOs for the entire log, for host objects. The remaining mapping process line charts for the result of web pages, as well as mismapping line charts for both object types, are available in Appendix A.1 through A.3. Average entropy normalized by max entropy before the first mapping for the different anonymization technique combinations are shown in Table 7.1 as percentages. For both hosts and web pages the combination of Tcpmkpub for IP address and generalization for ports, as well as PktAnon for IP address and generalization for ports, yield the highest percentage closest to the maximum possible value. At the other end, just using Crypto-PAn or AnonTool for IP addresses gives the lowest score for both object types. In general, it can be seen that the percentage of entropy when anonymization techniques are applied is much lower for web pages than for hosts. PktAnon and PktAnon with generalization stand out on the number of mismappings, both for hosts and web pages.

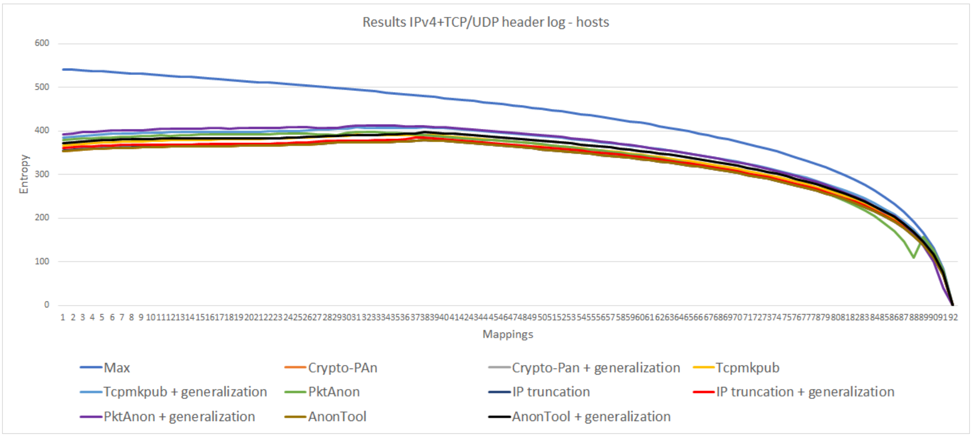


Figure 7.1: Average entropy vs. max entropy for IPv4+TCP/UDP header log - hosts

Table 7.1: The results for IPv4+TCP/UDP header log validation.

IPv4+TCP/UDP	Hosts	Web pages	Mismapping hosts	Mismapping web pages
Nr	92	39		
Crypto-PAn	65,28 %	53,96 %	0	2
Crypto-PAn + generalization	68,67 %	57,54 %	0	5
Tcpmkpub	67,76 %	55,29 %	0	2
Tcpmkpub + generalization	71,06 %	58,74 %	0	5
PktAnon	69,86 %	56,42 %	28	17
PktAnon + generalization	72,48 %	59,22 %	27	18
IP truncation	65,49 %	54,31 %	0	2
IP truncation + generalization	66,48 %	55,61 %	3	2
AnonTool	65,28 %	53,96 %	0	2
AnonTool + generalization	68,67 %	57,50 %	0	5

7.2 IPv6+TCP/UDP Header Log

The IPv6+TCP/UDP header log tests were performed on a log of 822 packets, with 50 hosts and 43 web pages. The mapping process line charts for the results, as well as mismapping line charts, are available in Appendix A.4 through A.7. Average entropy normalized by max entropy before the first mapping for the different anonymization technique combinations are shown in Table 7.2 as percentages. From the table the combination of PktAnon and generalization is the best of the ones tested. Crypto-PAn by itself gets the short end of the stick. These observations apply both to hosts and web pages.

Table 7.2: The results for IPv6+TCP/UDP header log validation.

IPv6+TCP/UDP	Hosts	Web pages	Mismapping hosts	Mismapping web pages
Nr	50	43		
Crypto-PAn	52,95 %	57,68 %	0	0
Crypto-PAn + generalization	57,22 %	63,30 %	3	7
IP truncation	53,78 %	57,80 %	2	0
IP truncation + generalization	54,99 %	59,68 %	2	7
PktAnon	57,85 %	60,52 %	28	9
PktAnon + generalization	60,23 %	64,90 %	31	13

7.3 NetFlow Log

A total of 300 packets, 143 hosts and 14 web pages were inspected for the NetFlow log results. The number of packets is so low because the number of distinct users in a NetFlow log is aggregated. this means that there are many distinct IP addresses which translates to many objects. The NetFlow log is limited in number of tests because of two reasons. The first is that maintaining the AS numbers leaves truncation as the only viable option for IP address anonymization, as discussed in Section 3.2. However, a different approach would be to anonymize the AS numbers, and perform IP address anonymization with all available techniques. This introduces the second reason. As also explained in Section 3.2, the format of the NetFlow log does not match the required format of the anonymization techniques. The mapping process line charts for the results, as well as mismapping line charts, are available in Appendix A.8 through A.11. Average entropy normalized by max entropy before the first mapping for the different anonymization technique combinations are shown in Table 7.3 as percentages. The results show that adding generalization to ports increases the entropy from just truncating IP addresses.

Table 7.3: The results for NetFlow log validation.

NetFlow	Hosts	Web pages	Mismapping hosts	Mismapping web pages
Nr	143	14		
IP truncation	59,44 %	44,26 %	0	0
IP truncation + generalization	60,35 %	46,94 %	2	0

7.4 Web Server Log

The results for the web server log are based on a log with 1500 packets and 123 host objects. The web server log would optimally also have results for the PktAnon and hashing combination, but as explained in Section 3.3, the format of the web server is not compatible with the one of PktAnon. The mapping process line charts for the results, as well as mismapping line charts, are available in Appendix A.12 and A.13. Average entropy normalized by max entropy before the first mapping for the different anonymization technique combinations are shown in Table 7.4 as percentages. The table indicates that both the Crypto-PAn and hashing combination and the IP truncation and hashing combination yield good, and equivalent results, both for the normalized value and mismappings.

Table 7.4: The results for web server log validation.

Web server	Hosts	Mismapping hosts
Nr	123	
Crypto-PAn + hashing	74,47 %	7
IP truncation + hashing	74,47 %	7

7.5 Syslog

The results for the syslog are based on a log with 5000 packets and 101 host objects. As explained in Section 6.4.3, the syslog is special compared to the other logs. Here there are no IP addresses to group into different objects. This is why the log is so disproportionate in size and objects compared to the other logs. An ideal validation would include more objects, but the validation code is too slow to do this in practice. Validation of 101 objects over 5000 packets takes approximately four hours to complete. The mapping process line charts for the results, as well as mismapping line charts, are available in Appendix A.14 and A.15. Average entropy normalized by max entropy before the first mapping for the different anonymization technique combinations are shown in Table 7.5 as percentages. The results suggest that just using hashing for the syslog messages in this log will not provide particularly good security when considering entropy measure.

Table 7.5: The results for syslog validation.

Syslog	Hosts	Mismapping hosts
Nr	101	
Hashing	51,76 %	0

7.6 OS Fingerprinting Prevention

The fields evaluated to contain personal data have now been validated. The remaining fields, the fields which are vulnerable to OS fingerprinting explained in Section 4.6.2, are validated in this section. The discussion from Chapter 3 showed that the fields DSCP, ECN, Identification and TTL from the IPv4 header, Traffic Class and Hop Limit from the IPv6 header, Sequence Number, Acknowledgement Number, TCP Flags and Window Size from the TCP header, and the TCP Flags and ToS field of NetFlow, all could be used to learn information on the running OS.

Validation has been performed on the IPv4+TCP/UDP header log regarding OS fingerprinting. Suggestions have been made in Chapter 3 for how to deal with the other logs as well, but are not validated because of limitations in the anonymization tools. The IPv6+TCP/UDP header log is not validated for OS fingerprinting attacks because SCRUB-tcpdump, used for obfuscating Sequence/Acknowledgement Number, TCP Flags and Window Size, does not manipulate IPv6 header. For the NetFlow log, the format is not compatible with the PCAP format which SCRUB-tcpdump demands.

Note that for the results, the checksums have not been recalculated, even though this would be a more correct approach. PktAnon is chosen as the tool to perform the recalculations. However, the techniques implemented specifically for this thesis, like generalization, are based on text file data. Thus recalculating the checksums for the PCAP file, formatting it to text file and then adding more anonymization leaves the recalculation done to the PCAP file futile. The recalculation needs to be performed after all anonymization techniques have been applied, but time constraints have restricted the implementation of formatting a text file back to PCAP format. This would have allowed for recalculation of checksums to be performed after anonymization with methods in a text format (like generalization).

Below, in Table 7.6, the average entropy normalized by max entropy before the first mapping for the different anonymization technique combinations are shown as percentages. T stands for Tcpmkpub, G for generalization. *T + G* describes the approach where no OS fingerprinting anonymization is applied. *T + G + OS* describes the results when anonymization for DSCP/ECN, Identification, TTL, Sequence/Acknowledgement Number, TCP Flags and Window Size is applied. The other entries show when each of these fields is added to Tcpmkpub and generalization. Figure 7.2 shows how the average and max entropy changes as the mappings are performed for hosts. The mapping process line charts for the results of web pages, as well as mismapping line charts for both object types, are available in Appendix A.16 through A.18.

It is observed that the normalized value increases with addition of all the fields vulnerable to OS fingerprinting, and that almost all individual anonymizations also increases the value. However, by applying anonymization to TCP Flags, the normalized average entropy decreases. For the individual technique results, it is observed that the Identification and DSCP/ECN fields gives the highest entropy, almost at the level of anonymization with all OS fingerprinting fields applied. By applying anonymization to Identification and DSCP/ECN individually to Tcpmkpub and generalization, the number of mismappings increases.

Table 7.6: The results for OS fingerprinting prevention validation.

IPv4+TCP/UDP	Hosts	Web pages	Mismapping hosts	Mismapping web pages
Nr	92	39		
T + G	71,06 %	58,74 %	0	5
T + G + OS	76,04 %	63,55 %	33	21
T + G + DSCP/ECN	74,00 %	61,56 %	35	22
T + G + Identification	73,82 %	59,84 %	32	12
T + G + TTL	71,27 %	59,46 %	4	8
T + G + Seq/Ack	71,84 %	59,38 %	0	6
T + G + TCP Flags	69,76 %	57,05 %	0	6
T + G + Window Size	71,66 %	59,32 %	0	9

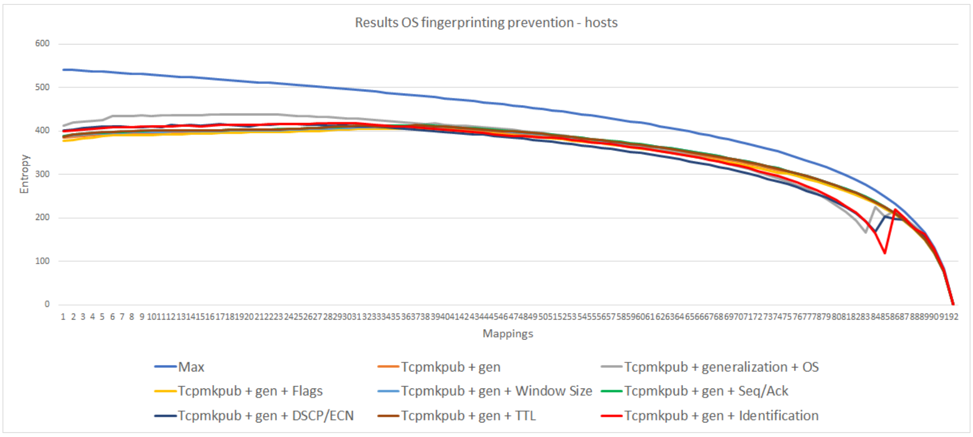


Figure 7.2: Average entropy vs. max entropy for OS fingerprinting prevention - hosts

Chapter 8 Discussion

By now all topics have been covered, and much practical work has been put into making the implementations and validations of both fields, logs and anonymization techniques as valuable as possible. Consider again the work process of the thesis in Figure 6.1. Every step is developed as an individual program[1], which makes the process flexible with regards to both debugging and usage. One example is when truncation is applied for IP addresses. The flexible programs allow for inter- and intrarecords as well as object categorization before any anonymization is performed. This is needed for IP truncation, as the specific objects would have been truncated and the logs would not be comparable to the other technique tests. Addresses 1.2.3.4 and 1.2.3.5 would both become object 1.2.3.0 and grouped together, which would give wrong results compared to the other techniques. With applying truncation after adding the records and dividing objects, the original objects are maintained, but the last part of the object IP addresses is removed.

In this chapter the results will first be discussed, followed by a proposal for best practice. A look at a commercial alternative is presented and compared with one of the tools tested in the thesis. Finally, limitations and future work will be explained.

8.1 Log Results Discussion

In this section the results of the different logs will be discussed. One by one an evaluation provides further insights into the results, what they mean, and what could have been done differently.

8.1.1 IPv4+TCP/UDP Header Log

Looking at the results for IPv4+TCP/UDP log from Section 7.1 there are observations which are interesting to investigate closer. From Section 4.5 it was discussed how

[1]Apart from some anonymization techniques.

truncation of IP addresses would destroy the most specific parts of an address, to the disappointment of researchers. However, when observing the results, IP truncation turns out to be just slightly better than Crypto-PAn and AnonTool, and worse than both Tcpmkpub and Pktanon. For the tests where generalization of ports is added to IP address anonymization, IP truncation scores lowest of all. These observations apply to both hosts and web pages.

The combinations with the highest average entropy are Tcpmkpub + generalization and PktAnon + generalization. Both have average entropy over 70 % of maximum entropy for hosts, and approximately 59 % for web pages. The big difference between these two is that while the Tcpmkpub combination manages to map every object for hosts, the PktAnon combination makes the validation method miss 27 objects, nearly 30 % of the observed objects. While this in theory is a good and wanted result, it brings up the question of why PktAnon stands out in the way it does. It uses hashing on the complete IP address to anonymize the address as opposed to the other techniques, which might have an impact. However, the hashing would then make it similar to AnonTool, and these two fields should have approximately the same results. Somehow, PktAnon manages to both increase the entropy and number of mismappings for hosts and web pages compared to AnonTool. Another peculiarity with the results of PktAnon is displayed in Figure 7.1. At the end of the mappings process, approximately for mapping nr 88, PktAnon dips in entropy. The reason might be that all the objects which easiest are mapped have been removed, leaving the more challenging objects to map behind. This dip does not reappear when generalization for ports is added to PktAnon.

The results from the IPv4+TCP/UDP header log indicates that PktAnon + generalization is the best combination when measured in entropy. The recommended approach is nevertheless Tcpmkpub + generalization. The reason is that this combination scores almost as high as the PktAnon combination, but is built on a more sound anonymization. The number of mismappings of the PktAnon combinations suggest that there might be something incorrect with the results, and such, Tcpmkpub + generalization is a safer bet. Another conclusion from this log is that adding generalization to Source and Destination Ports, by applying the novel method suggested in this thesis, generally increases the entropy, mostly by 3-4 %. It does not have much impact on mismappings, with only a slight increase for some of the logs. Generally, the entropy decreases for web pages, likely due to the smaller number of objects observed.

8.1.2 IPv6+TCP/UDP Header Log

The IPv6+TCP/UDP header log results shown in Table 7.2 are more limited than the IPv4+TCP/UDP header log, but still carry some interesting results. PktAnon

+ generalization is again the highest scoring combination, accompanied by a large number of mismappings. for Crypto-PAn and truncation by themselves, truncation scores the highest, while the tables are shifted when generalization for ports is added. For IPv6 addresses the truncation is performed on the last 16 bits. This could be increased to cover more of the host part of the address. The last 64 bits of and IPv6 is generally the host part, called Interface ID in IETF RFC 4291 [ipv].

The entropies for hosts and web pages are closer for this log, as the number of hosts and web pages are almost equal. Even though there are still more hosts than web pages, the entropy results for the latter are higher than for the former. This might imply that characteristics of the web pages are harder to distinguish. Generalization of ports still increases the entropy.

8.1.3 NetFlow Log

For the NetFlow log there are only two combinations tested. As already mentioned in Section 7.3, the choice of leaving AS numbers open, together with formats not matching, leaves the NetFlow log with few possibilities to test.

The alternatives left are then truncation of IP address, and truncation in combination with generalization of ports. For NetFlow log the addition of generalization only slightly increases the entropy, which has been the case when adding generalization to truncation for the other logs. This suggests that truncation combined with generalization is an insignificant combination, regardless of log.

Generally speaking, there is a gap between hosts and web pages, which further increases the suspicion that the number of hosts or logs plays a rather important part in the entropy results. The results for web pages are particularly low, likely because there are so few web page objects.

What could be done for the NetFlow log is to decide that AS numbers cannot be kept in their original form, even though researchers would prefer to keep it that way. This would allow for more better IP anonymization techniques than truncation, which in turn could use more of the generalization's potential.

8.1.4 Web Server Log

There are two tests performed for the web server log. As explained in Section 7.4, the format stops other IP address anonymization techniques from being utilized. In addition, there is only the Request field, apart from IP address, that needs anonymization in this log.

The peculiar result is that both hashing with Crypto-PAn and truncation added yield the same results. Both combinations are shown to score relatively high compared to results from other logs, and provide some mismappings as well. One reason for the equal results could be that the hashing dominates the log. The hashing in the Request field is so essential in dividing the objects in the logs that the anonymization of the IP addresses completely drowns. Consider Feature Selection explained in Section 6.4.3. It is possible that the Request field and the IP address fields are grouped together into one feature in both cases. Then the other fields would operate as features equally for both cases, resulting in similar output from the validation process.

8.1.5 Syslog

The result of Syslog from Section 7.5 shows that hashing of the syslog is not very effective regarding entropy. There are no mismappings, and the average entropy before the first mapping is only slightly above half of the maximum entropy. In the syslog Timestamps, Hostname, App-name and Message are considered. Timestamps could be individual and fairly simple to divide the objects on. The Hostname is the same for the log as it is the common host for the observed system, ergo the entropy for this field will be low. The App-names are the ones dividing the logs into different objects on, and will provide a hit or miss with entropy. That leaves the Messages. Many of the Messages contain similar parts since certain actions are performed again on a system. However, there always seems to be some values changing in the Messages, presumably the identity of the action. This means that practically all messages are hashed differently; you never get a scenario where a message can be recognized in another object than the one it resides in. To distinguish between objects is easier compared to other logs, since so much information is individual. In addition, the small number of fields in the syslog means that every field contributes more to the average than for other logs.

8.1.6 OS Fingerprinting Prevention Results

The OS fingerprinting prevention results from Table 7.6 show that applying all techniques will increase the entropy with 5 %. Curiously, the TCP Flags anonymization decreases the entropy. This leads to questions of whether the method used is correctly applied, and if another method for the flags would be better. For further validation of the OS fingerprinting fields, a different method should be chosen to anonymize TCP Flags.

Another curious observation is that DSCP/ECN anonymization leads to the highest entropy when only adding individual fields to Tcpmkpub and generalization is considered. The DSCP/ECN fields are only replaced by a "0" value, which for most packets would be the current value anyway. Identification anonymization is also proven to provide good results, even though it removes much information for analysis. Both DSCP/ECN and Identification anonymization lead to many mismappings, indicating that they impact the security of the log in a large way.

Consider Figure 7.2, and observe the dip in average entropy when all field anonymization is applied at approximately mapping 83. The dip is likely caused by the DSCP/ECN and Identification field anonymization, as the figure shows that both these fields display similar behavior. Why this happens for the two field combinations is hard to explain, but it might indicate that at this point, all the objects which are easy to map are removed from the log, and the ones harder to differentiate are left.

8.2 Proposal

By now different logs with their own combinations of fields and anonymization have been investigated and discussed. Multiple conclusions can be drawn. The pseudonymization suggestion from Section 4.3 is a foundation for the proposal. A one-to-one mapping between anonymized and original logs needs to be stored securely at the company, with access restrictions and agreements with researchers in order.

With regards to IP address anonymization, avoid using IP truncation, as it removes a researcher's possibility of analyzing the whole network topology, and the entropy results compared to the other methods tested are not significantly better. Tcpmkpub seems like the best tool to use, as it combines sound theory and the second best results. PktAnon provides slightly better results than Tcpmkpub, but includes a few uncertainties and peculiarities. Generalization for Source and Destination Ports by the method suggested in this thesis offers a steady increase in entropy with 3-4 % in most cases.

To increase the entropy for a NetFlow log, one would have to decide that AS numbers need anonymization, which would free up IP addresses to be anonymized with other technique than truncation.

Hashing of URLs does provide a web server log with relatively high entropy in combination with both IP address anonymization tested, even though the same cannot be said about the hashing applied to the syslog. The nature of a syslog might make it challenging to increase the entropy.

For OS fingerprinting prevention, applying anonymization to DSCP/ECN, Identification, TTL, Sequence/Acknowledgement Number and Window Size increases the entropy, but only DSCP/ECN and Identification does so in a significant manner, at least for hosts. The technique used for TCP Flags should be avoided, while the DSCP/ECN field seems safe to be anonymized. The Identification anonymization might remove too much information from a researcher, and should only be applied if necessary.

8.3 SafePcap

By now all anonymization technique combinations are documented and results discussed. The anonymization techniques are chosen since they are based on some information theory or cryptography. In addition, there exists a commercially available anonymization software called SafePcap [saf]. SafePcap proclaims that it makes files in the PCAP format GDPR compliant, with a demo version available for free since April 4th, 2019. A test was performed with this tool on a PCAP file and measured against a PCAP file with Crypto-PAn applied to the IP addresses. These test results are provided in Appendix B.1 and B.4.

The reason not much emphasis was put on SafePcap in this thesis [2] is that it has several flaws. Firstly, the maximum size of a PCAP file to test the demo version on is limited to 50 kB, which equates to a rather small log to perform serious analysis on. This can be seen by the results as the log inside this size limit equates to just 28 hosts and 12 web pages. Secondly, and more crucially, is that none of the anonymization techniques they use are documented. There is not even a summary of which fields are being manipulated. In the future, if a commercially available tool is to be made public and reliable, this kind of documentation needs to be in place. Blindly relying on a tool to do the work for you might be wise for the unwise, but when security is of concern, maximum care needs to be taken.

8.4 Limitations

Anonymization techniques have been compared, results have been discussed, and a proposal has been presented. During the thesis work there have been some stumbling blocks, however, and they will be discussed in this section.

8.4.1 Implementation Issues

One problem that emerged after the implementation was completed was that the normalized mutual information for some features proved to be greater than one. Since a normalized value is supposed to be between zero and one, this was a peculiar and problematic issue. Several steps were considered to try to fix this problem. As explained the implementation was done based on an understanding of the formulas in the article by Coull et al. [CWK+08]. The implementation had been tested on smaller data sets and proven to be correct, but presented suspiciously many results which had

[2] That is, apart from the late release date of the working demo version.

more than 0.99 in normalized information value, with some of them being between 1.0000000000000000298 and 1.03. The implementation of the entropy validation method is based on getting two arrays of strings from the user. By calculating the entropy for some of the actual arrays from the network data by hand, a comparison to the calculations made by the program was made. The results proved to be almost identical. The difference originated from a rounding error which was quickly fixed.

Another problem was discovered after calculating mutual information of two fields by hand, which the program had shown to have normalized mutual information more than 1. After calculating the mutual information of two fields, which the program had shown to have normalized mutual information of more than 1, by hand, the problem was discovered. To get the joint distribution of two arrays, the arrays are merged together to be able to get their probability. However, the implemented merging process allowed for a tiny error to occur, and thus fixing this merging removed the results with too high normalized mutual information. Some of the results still could be more than 1, but those were typically of the format 1.0000000000000001, that is, 15 zeroes of precision.

8.4.2 Auxiliary Information

Considering the validation process described in section 6.4.3 based on Coull et al. [CWK$^+$08], a part has been left for future work. When mapping all the AOs to their respective UO, one should take into consideration the new information learned by the last mapping. In the case of prefix-preserving anonymization of IP addresses, explained with Crypto-PAn in Section 4.4.1, the information learned in one mapping would make the job of the next mapping potentially easier, as you have gained information about the anonymization of some of the prefixes in the log. This information would then be used in the next iteration. If the next chosen AO - that is, the one with the lowest entropy - has some IP address prefixes which already is mapped, you know which objects are able to match this knowledge. The information learned is called auxiliary information.

Consider an example. If you learned that the IP address *192.52.XX.XX* is anonymized as *234.61.XX.XX*, then this information is stored. You continue mapping the next AO. When an AO now has *234.61* as its two first octets, then you know that the only UOs which could equal this AO are the ones with an IP address beginning with *192.52*. This also applies to other fields, depending on which anonymization technique is used for that particular field. Auxiliary information could be used to identify weaknesses in an anonymization technique, which Brekne et al. did in [BÅØ05] and [BÅ05].

Time restrictions for this master thesis prevented the implementation for how to gain the auxiliary information. What needs to be said is that by using the auxiliary information from each mapping between AO and UO, the entropy curves presented in Chapter A would in all likelihood be lower. The auxiliary information would also increase knowledge of how the different anonymization techniques would fare after every mapping.

8.4.3 Log Sizes

As per the pre-project of this master thesis, a goal was to test the chosen approach on real-world sized logs. However, as the work has progressed, this has not been deemed manageable. The code created to perform the validation is too slow to fully function in this time frame on logs of very large sizes. Validation test on IPv4+TCP/UDP header log with 693 packets, 92 hosts and 39 web pages finishes in approximately 49 minutes. The logs tested in the thesis are for the most part consisting of a few hundred packets. A more realistic log would consist of hundreds of thousands of packets, perhaps even millions. The validation results still indicate which combinations of anonymization approaches will give the best privacy, but the scale is significantly smaller than optimal. As already discussed in this chapter, the characteristics of the different techniques will still make it possible to recommend the better solutions. A future work could be to improve the implementation of the validation process with multithreading used when programming the implementation. This will allow for actions which are not dependent on each other to be performed in parallel.

8.4.4 Formatting

By now it should be established that not all formats of either logs or anonymization frameworks are working properly. Firstly, some of the frameworks are not equipped to deal with IPv6 header. AnonTool is said to handle NetFlow data, but this functionality does not seem to work properly. In addition, the working implementations of the novel contributions are not optimal when considering their formats. While techniques like *hashing* and *generalization* have been implemented for this thesis to fit into the chosen approach, they have been implemented for manipulation of a text file. This applies to the validation process as well. While the implementations are working, it is inconvenient that many of the anonymization frameworks use a PCAP file. Future work would be to implement correctly a conversion from text file into a PCAP file, thus making it possible to apply the techniques of the frameworks for a web server log or a NetFlow log. This would also allow for checksum recalculation after text file anonymization technique have been applied.

Chapter 9

Conclusion

The topic of this master thesis has been GDPR compliance for sharing personal data while researchers still can perform valuable analyses on personal data that has been processed. An attempt to measure the level of anonymization from different anonymization techniques is provided. A novel proposal as to which fields need anonymization, along with which techniques to use on these fields is presented. The techniques used are within the boundaries of the GDPR, where a theoretical solution is suggested for complying with this new regulation. By storing the anonymized and the original logs as a one-to-one mapping, the demand that pseudonymization should be reversible with additional information is met.

Tcpmkpub is found to be the most suitable anonymization technique for IP addresses. The novel contribution of generalization of port numbers increases the anonymity of a log, while still preserving useful data for researchers. Generalization increases the entropy with approximately 3-4 % for most logs. IP addresses, along with Request, Identification Protocol and Userid in a web server log, and Hostname and Message from a syslog, are evaluated to be the most personal data sensitive fields. Hashing is chosen as the method for both the Request field from web server log, and the Hostname and Message field from syslog. While the combination of hashing Request field and anonymize IP addresses is shown to provide good anonymity for a web server log (approximately 75 % of maximal entropy value), the hashing for a syslog does not produce satisfactory results (approximately 52 %). For extraordinary situations OS fingerprinting protection with several fields is shown to increase the anonymity and number of mismappings, but this also reduces the usefulness of the data log for researchers.

Future works from this thesis could be that truncation of timestamps might have benefited the anonymization. This would have removed a useful piece of information from the analysis to a researcher, but a further investigation into the balance between truncating timestamps and researchers utilizing it, could be conducted. Auxiliary information would possibly have decreased the results, making the results in this thesis somewhat optimistic compared to what actually would be the case in a more real scenario. Hence implementing the auxiliary information process is left as future work. The validation could have benefited from having larger logs, which would have given a more realistic comparison of the techniques. For this reason, increasing the speed of the validation process by optimizing the implementation code would be needed, and is a desired future work. Another future work would be to implement novel anonymization solutions for syslogs, as the approach in this thesis is insufficient by the entropy measure used in this thesis.

References

[ano] AnonTool. https://www.ics.forth.gr/dcs/Activities/Projects/anontool.html. Accessed: 2019-05-17.

[BÅ05] Tønnes Brekne and André Årnes. Circumventing ip-address pseudonymization. In *Communications and Computer Networks*, pages 43–48, 2005.

[BÅØ05] Tønnes Brekne, André Årnes, and Arne Øslebø. Anonymization of ip traffic monitoring data: Attacks on two prefix-preserving anonymization schemes and some proposed remedies. In *International Workshop on Privacy Enhancing Technologies*, pages 179–196. Springer, 2005.

[BST$^+$10] Martin Burkhart, Dominik Schatzmann, Brian Trammell, Elisa Boschi, and Bernhard Plattner. The role of network trace anonymization under attack. *ACM SIGCOMM Computer Communication Review*, 40(1):5–11, 2010.

[Cha17] Gauthier Chassang. The impact of the eu general data protection regulation on scientific research. *ecancermedicalscience*, 11, 2017.

[com] Log Files. https://httpd.apache.org/docs/1.3/logs.html. Accessed: 2019-05-17.

[Cor18] Marc Cornock. General data protection regulation (gdpr) and implications for research. *Maturitas*, 111:A1, 2018.

[cry] Crypto-PAn. https://www.cc.gatech.edu/computing/Networking/projects/cryptopan/. Accessed: 2019-05-17.

[CWK$^+$08] Scott E Coull, Charles V Wright, Angelos D Keromytis, Fabian Monrose, and Michael K Reiter. Taming the devil: Techniques for evaluating anonymized network data. 2008.

[DH18] Niels Van Dijkhuizen and Jeroen Van Der Ham. A survey of network traffic anonymisation techniques and implementations. *ACM Computing Surveys (CSUR)*, 51(3):52, 2018.

[dir] DIRECTIVE 95/46/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL. https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:31995L0046. Accessed: 2019-05-25.

[FAP09] Michael Foukarakis, Demetres Antoniades, and Michalis Polychronakis. Deep packet anonymization. In *Proceedings of the Second European Workshop on System Security*, pages 16–21. ACM, 2009.

[FXAM04] 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.

[gdpa] GDPR Article 15. https://gdpr-info.eu/art-15-gdpr/. Accessed: 2019-06-01.

[gdpb] GDPR Article 16. https://gdpr-info.eu/art-16-gdpr/. Accessed: 2019-06-01.

[gdpc] GDPR Article 17. https://gdpr-info.eu/art-17-gdpr/. Accessed: 2019-05-25.

[gdpd] GDPR Article 18. https://gdpr-info.eu/art-18-gdpr/. Accessed: 2019-06-01.

[gdpe] GDPR Article 19. https://gdpr-info.eu/art-19-gdpr/. Accessed: 2019-06-01.

[gdpf] GDPR Article 20. https://gdpr-info.eu/art-20-gdpr/. Accessed: 2019-06-01.

[gdpg] GDPR Article 32. https://gdpr-info.eu/art-32-gdpr/. Accessed: 2019-05-25.

[gdph] GDPR Article 4. https://gdpr-info.eu/art-4-gdpr/. Accessed: 2019-05-25.

[gdpi] GDPR Article 5. https://gdpr-info.eu/art-5-gdpr/. Accessed: 2019-05-26.

[gdpj] GDPR Article 6. https://gdpr-info.eu/art-6-gdpr/. Accessed: 2019-05-25.

[gdpk] GDPR Article 83. https://gdpr-info.eu/art-83-gdpr/. Accessed: 2019-05-25.

[gdpl] GDPR Article 89. https://gdpr-info.eu/art-89-gdpr/. Accessed: 2019-05-25.

[gdpm] GDPR Article 9. https://gdpr-info.eu/art-9-gdpr/. Accessed: 2019-05-25.

[gdpn] GDPR Recital 26. https://gdpr-info.eu/recitals/no-26/. Accessed: 2019-05-26.

[gdpo] Official Journal of the European Union. https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=OJ:L:2016:119:FULL. Accessed: 2019-05-25.

[gdpp] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). https://eur-lex.europa.eu/eli/reg/2016/679/oj. Accessed: 2019-05-25.

[GMS08] Th Gamer, Chr Mayer, and Marcus Schöller. Pktanon–a generic framework for profile-based traffic anonymization. *PIK-Praxis der Informationsverarbeitung und Kommunikation*, 31(2):76–81, 2008.

[HB96] John Hawkinson and Tony Bates. Guidelines for creation, selection, and registration of an autonomous system (as). Technical report, 1996.

[his] History of The Web. https://webfoundation.org/about/vision/history-of-the-web/. Accessed: 2019-05-25.

[ian] Service Name and Transport Protocol Port Number Registry. https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt. Accessed: 2019-05-25.

[ide] Identification Protocol. https://tools.ietf.org/html/rfc1413. Accessed: 2019-05-17.

[Inta] Internet Protocol 1981. https://tools.ietf.org/html/rfc791#section-3.1. Accessed: 2019-05-11.

[Intb] Internet Protocol, Version 6 (IPv6) Specification. https://tools.ietf.org/html/rfc2460#section-3. Accessed: 2019-05-12.

[ipv] IP Version 6 Addressing Architecture. https://tools.ietf.org/html/rfc4291. Accessed: 2019-05-31.

[KAA$^+$06] Dimitris Koukis, Spyros Antonatos, Demetres Antoniades, Evangelos P Markatos, and Panagiotis Trimintzios. A generic anonymization framework for network traffic. In *2006 IEEE International Conference on Communications*, volume 5, pages 2302–2309. IEEE, 2006.

[KLS09] Justin King, Kiran Lakkaraju, and Adam Slagell. A taxonomy and adversarial model for attacks against network log anonymization. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1286–1293. ACM, 2009.

[KM03] Geoff Kuenning and Ethan L Miller. Anonymization techniques for urls and filenames. *TR UCSC-CRL-03-05, University of California at Santa Cruz*, 2003.

[Lea18] Nathan C Lea. How will the general data protection regulation affect healthcare? *Acta medica portuguesa*, 31(7-8):363–365, 2018.

[Lyo08] Gordon "Fyodor" Lyon. Nmap network scanning : Official nmap project guide to network discovery and security scanning. volume 1. Insecure.Com LLC, 2008.

[MBBvD16] Menno Mostert, Annelien L Bredenoord, Monique CIH Biesaart, and Johannes JM van Delden. Big data in medical research and eu data protection law: challenges to the consent or anonymise approach. *European Journal of Human Genetics*, 24(7):956, 2016.

[MME$^+$18] Miranda Mourby, Elaine Mackey, Mark Elliot, Heather Gowans, Susan E Wallace, Jessica Bell, Hannah Smith, Stergios Aidinlis, and Jane Kaye. Are 'pseudonymised'data always personal data? implications of the gdpr for administrative data research in the uk. *Computer Law & Security Review*, 34(2):222–233, 2018.

[nfd] NFDUMP. http://nfdump.sourceforge.net/. Accessed: 2019-05-15.

[OLSB09] Philipp Offermann, Olga Levina, Marten Schönherr, and Udo Bub. Outline of a design science research process. In *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*, page 7. ACM, 2009.

[PAPL06] Ruoming Pang, Mark Allman, Vern Paxson, and Jason Lee. The devil and packet trace anonymization. *ACM SIGCOMM Computer Communication Review*, 36(1):29–38, 2006.

[pbk] PBKDF2. https://tools.ietf.org/html/rfc8018. Accessed: 2019-05-24.

[pkta] PktAnon. http://www.tm.uka.de/software/pktanon/. Accessed: 2019-05-17.

[pktb] pkts.io. https://github.com/aboutsip/pkts. Accessed: 2019-05-13.

[reg] Difference between a Regulation, Directive and Decision. https://www.usda-eu.org/eu-basics-questions/difference-between-a-regulation-directive-and-decision/. Accessed: 2019-05-25.

[rfca] Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. https://tools.ietf.org/html/rfc7231#section-4.3. Accessed: 2019-06-02.

[rfcb] PATCH Method for HTTP. https://tools.ietf.org/html/rfc5789. Accessed: 2019-06-02.

[rfcc] The Syslog Protocol. https://tools.ietf.org/html/rfc5424#section-6. Accessed: 2019-05-17.

[RLH05] Yakov Rekhter, Tony Li, and Susan Hares. A border gateway protocol 4 (bgp-4). Technical report, 2005.

[RS04] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *International workshop on fast software encryption*, pages 371–388. Springer, 2004.

[saf] SafePcap. https://omnipacket.com/safepcap. Accessed: 2019-05-15.

[scr] SCRUB-tcpdump. http://scrub-tcpdump.sourceforge.net/index.php. Accessed: 2019-05-24.

[SD02] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In *International Workshop on Privacy Enhancing Technologies*, pages 41–53. Springer, 2002.

[SG18] Galit Shmueli and Travis Greene. Analyzing the impact of gdpr on data scientists using the infoq framework. 2018.

[SLL05] Adam J Slagell, Yifan Li, and Katherine Luo. Sharing network logs for computer forensics: A new tool for the anonymization of netflow records. In *Workshop of the 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks, 2005.*, pages 37–42. IEEE, 2005.

[SLL06] Adam J Slagell, Kiran Lakkaraju, and Katherine Luo. Flaim: A multi-level anonymization framework for computer and network logs. In *LISA*, volume 6, pages 3–8, 2006.

[tcpa] Tcpdpriv. http://fly.isti.cnr.it/software/tcpdpriv/. Accessed: 2019-05-17.

[tcpb] tcpdump & libpcap. https://www.tcpdump.org/. Accessed: 2019-05-11.

[tcpc] Tcpmkpub. http://www.icir.org/enterprise-tracing/tcpmkpub.html. Accessed: 2019-05-17.

[TCPd] Transmission Control Protocol 1981. https://tools.ietf.org/html/rfc793#section-3.1. Accessed: 2019-05-13.

[UDP] Transmission Control Protocol 1981. https://tools.ietf.org/html/rfc768. Accessed: 2019-05-13.

[ver] NetFlow Version 9 Flow-Record Format. https://www.cisco.com/en/US/technologies/tk648/tk362/technologies_white_paper09186a00800a3db9.html. Accessed: 2019-05-17.

[Wie14] Roel J Wieringa. *Design science methodology for information systems and software engineering.* Springer, 2014.

[XFAM01] Jun Xu, Jinliang Fan, Mostafa Hamed Ammar, and Sue B Moon. On the design and performance of prefix-preserving ip traffic trace anonymization. Technical report, Georgia Institute of Technology, 2001.

[XFAM02] Jun Xu, Jinliang Fan, Mostafa H Ammar, and Sue B Moon. Prefix-preserving ip address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In *10th IEEE International Conference on Network Protocols, 2002. Proceedings.*, pages 280–289. IEEE, 2002.

[yac] Yet another Crypto-PAn implementation for Python. https://github.com/keiichishima/yacryptopan. Accessed: 2019-05-31.

[YWH$^+$07a] William Yurcik, Clay Woolam, Greg Hellings, Latifur Khan, and Bhavani Thuraisingham. Scrub-tcpdump: A multi-level packet anonymizer demonstrating privacy/analysis tradeoffs. In *2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops-SecureComm 2007*, pages 49–56. IEEE, 2007.

[YWH$^+$07b] William Yurcik, Clay Woolam, Greg Hellings, Latifur Khan, and Bhavani Thuraisingham. Toward trusted sharing of network packet traces using anonymization: Single-field privacy/analysis tradeoffs. *arXiv preprint arXiv:0710.3979*, 2007.

Appendix A

Average entropy vs. max entropy Figures

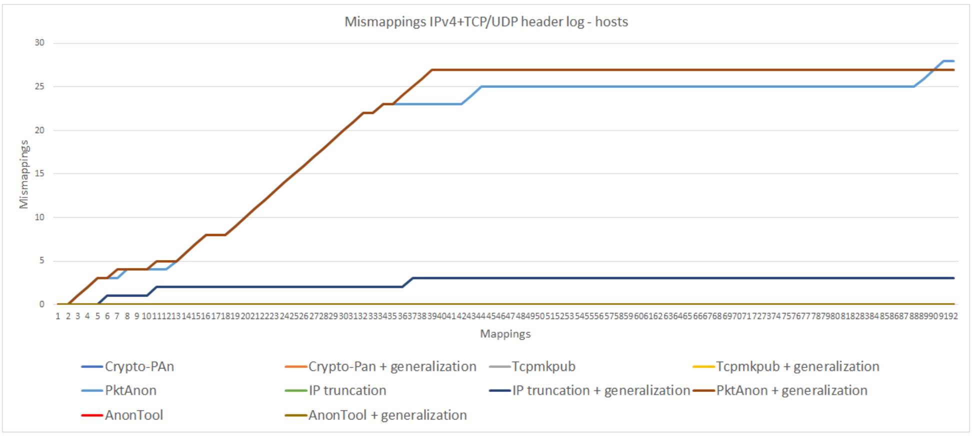


Figure A.1: Mismappings for IPv4+TCP/UDP header log - hosts

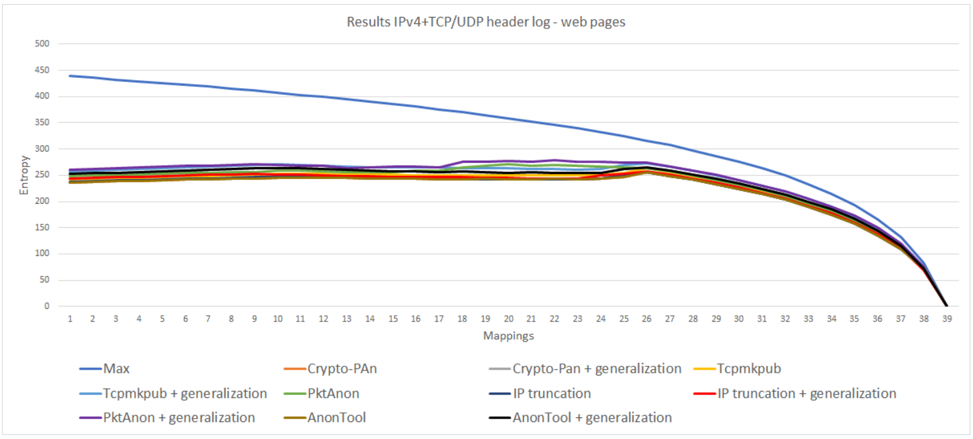


Figure A.2: Average entropy vs. max entropy for IPv4+TCP/UDP header log - web pages

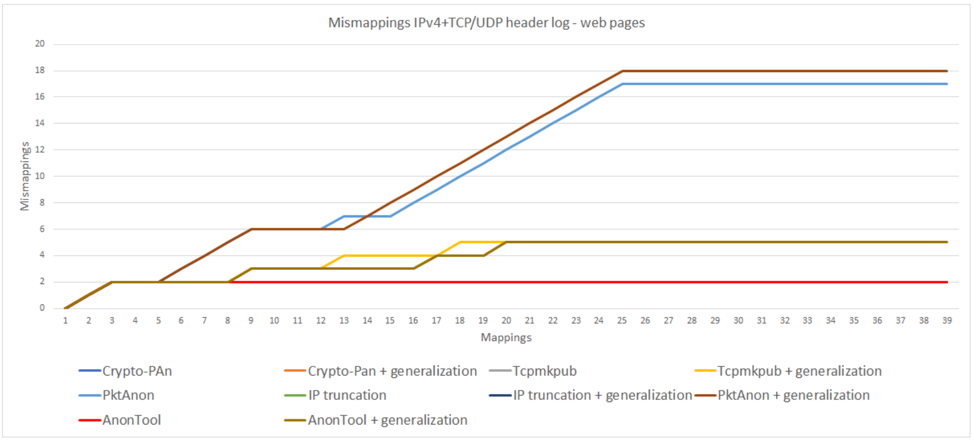


Figure A.3: Mismappings for IPv4+TCP/UDP header log - web pages

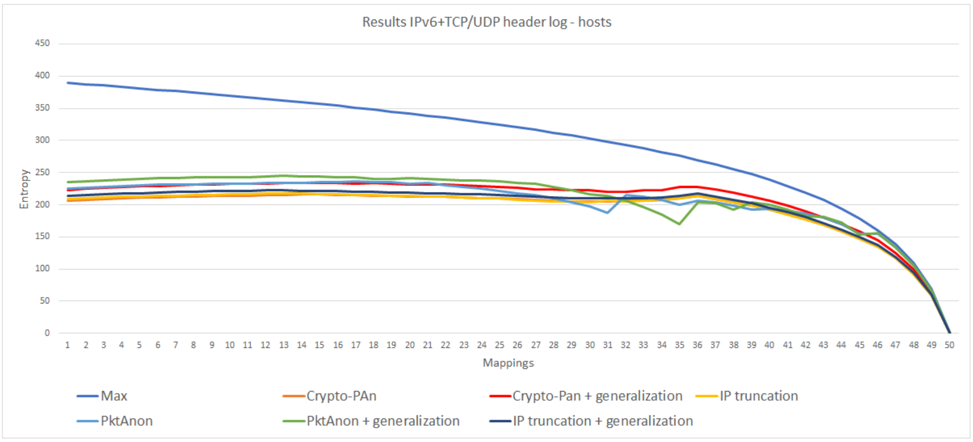


Figure A.4: Average entropy vs. max entropy for IPv6+TCP/UDP header log - hosts

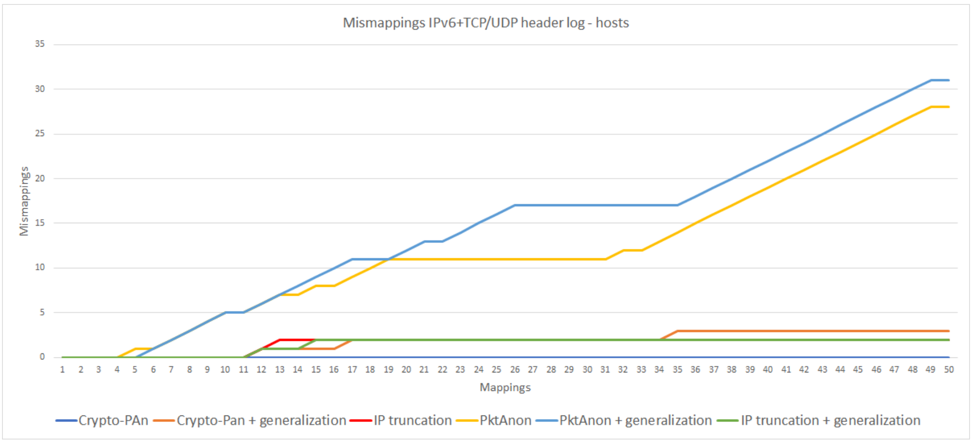


Figure A.5: Mismappings for IPv6+TCP/UDP header log - hosts

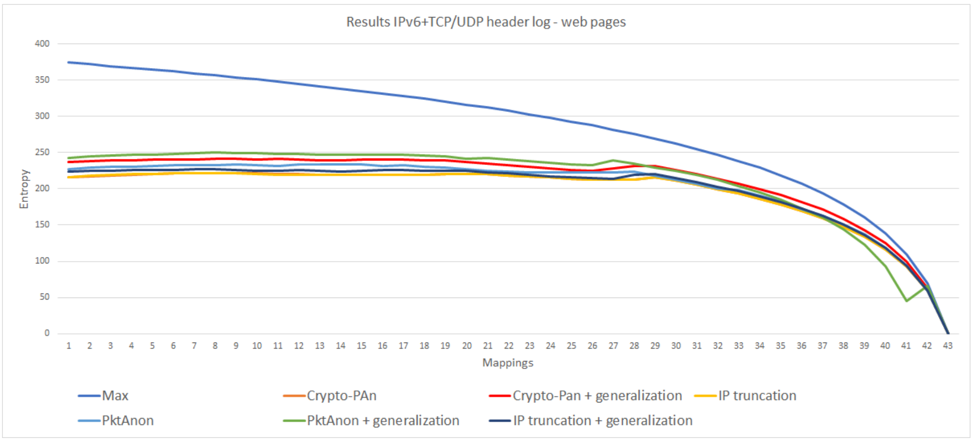


Figure A.6: Average entropy vs. max entropy for IPv6+TCP/UDP header log - web pages

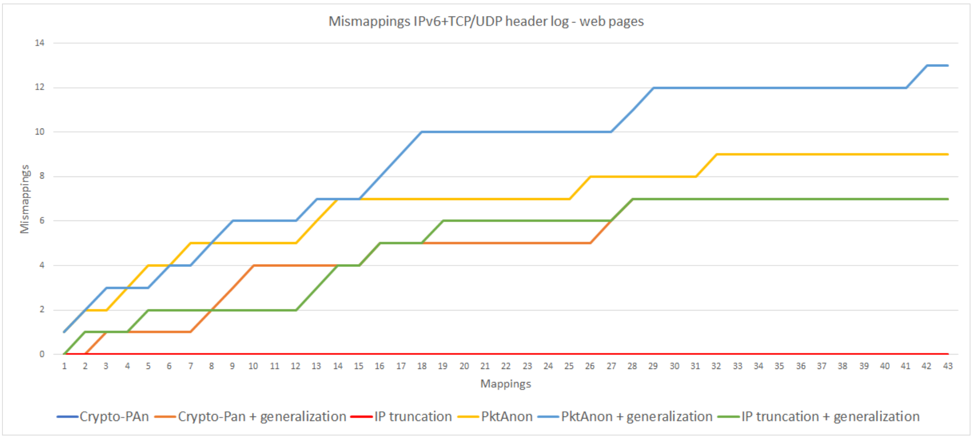


Figure A.7: Mismappings for IPv6+TCP/UDP header log - web pages

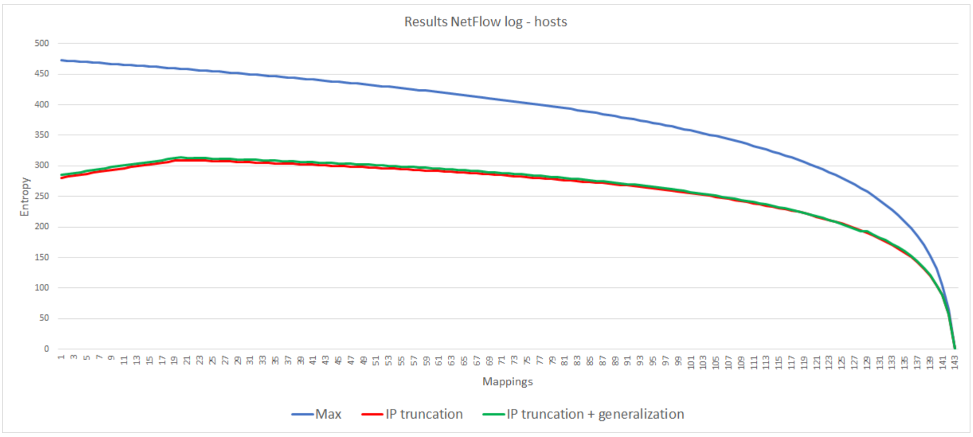


Figure A.8: Average entropy vs. max entropy for NetFlow log - hosts

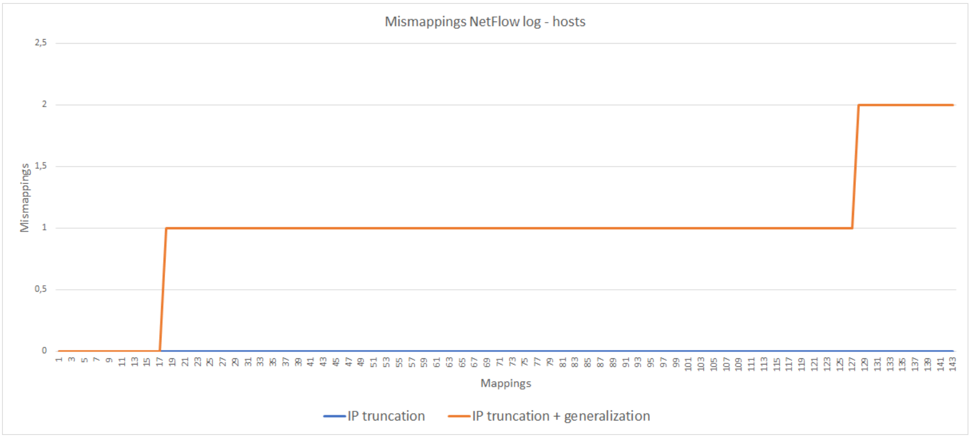


Figure A.9: Mismappings for NetFlow log - hosts

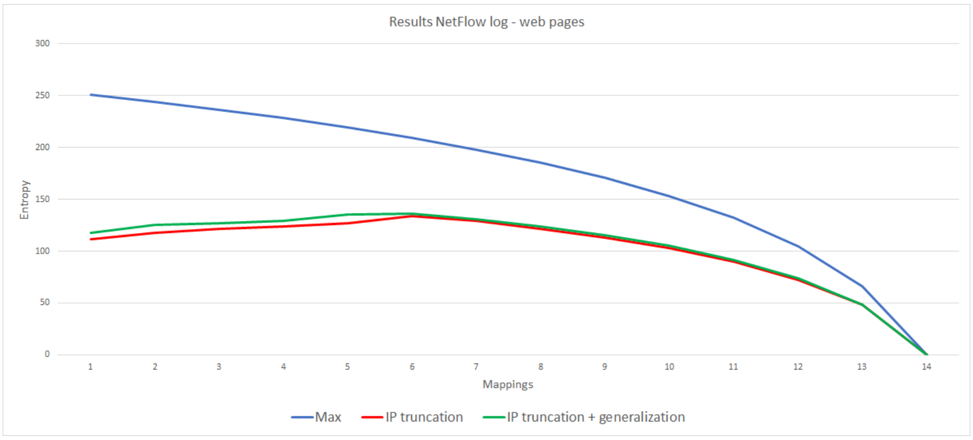


Figure A.10: Average entropy vs. max entropy for NetFlow log - web pages

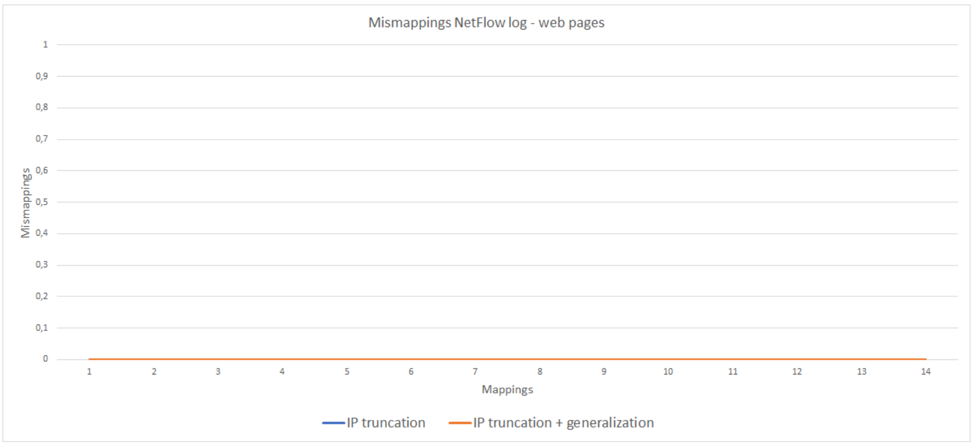


Figure A.11: Mismappings for NetFlow log - web pages

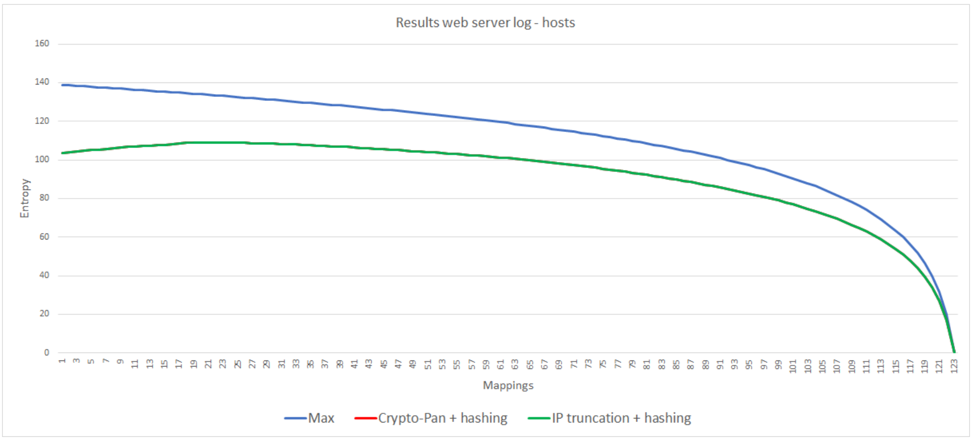


Figure A.12: Average entropy vs. max entropy for web server log - hosts

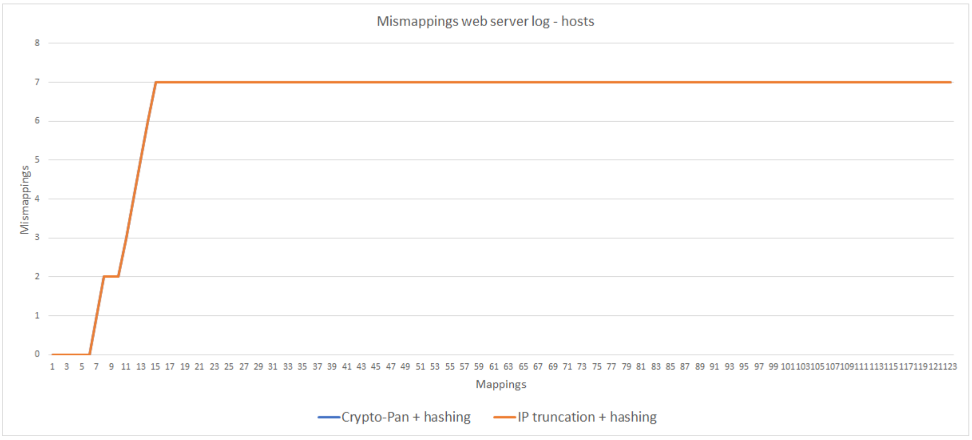


Figure A.13: Mismappings for web server log - hosts

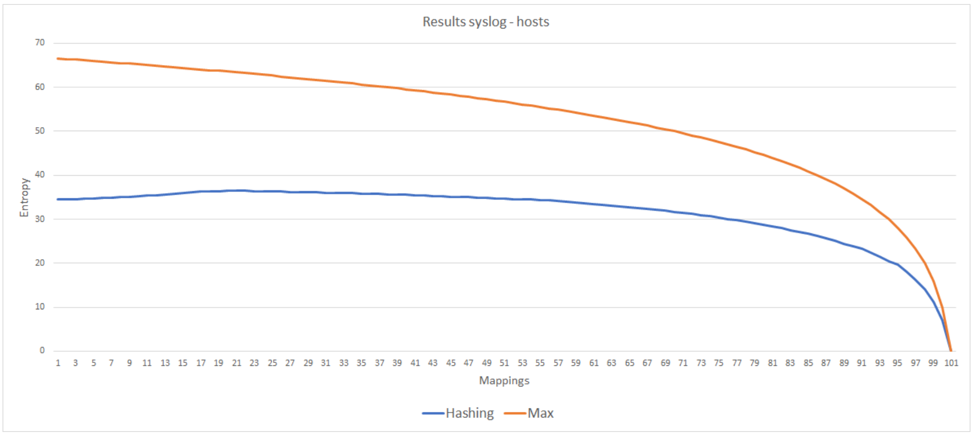


Figure A.14: Average entropy vs. max entropy for syslog - hosts

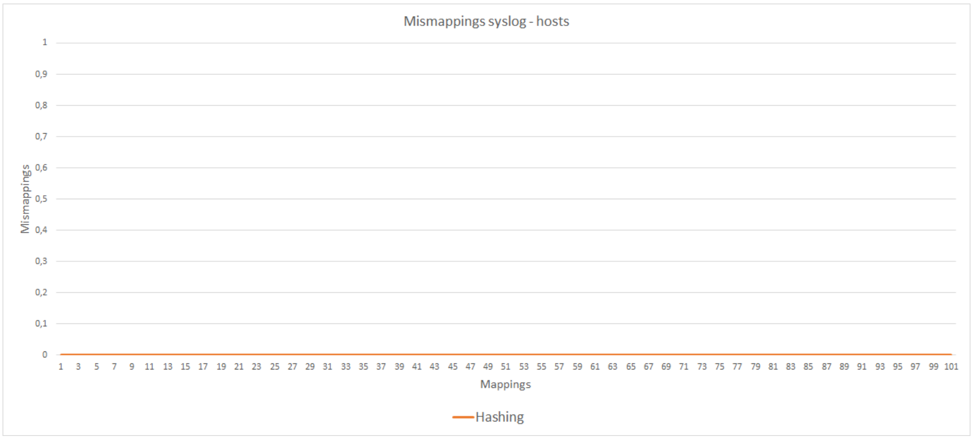


Figure A.15: Mismappings for syslog - hosts

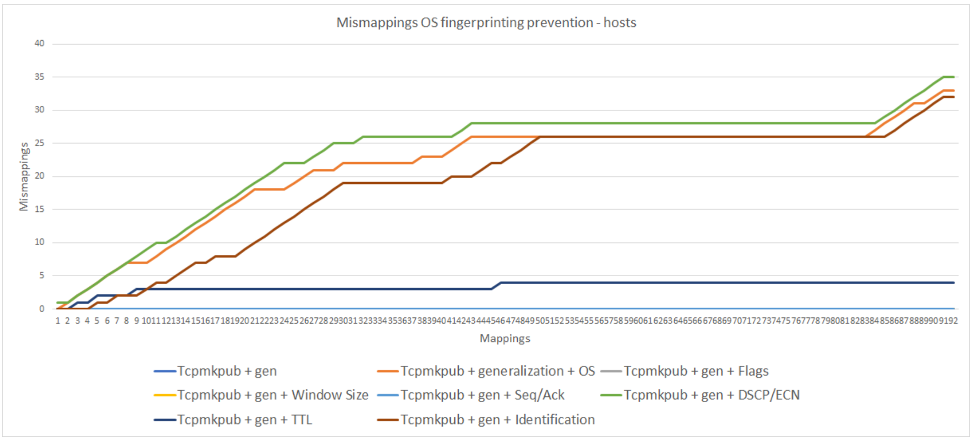


Figure A.16: Mismappings for OS fingerprinting prevention - hosts

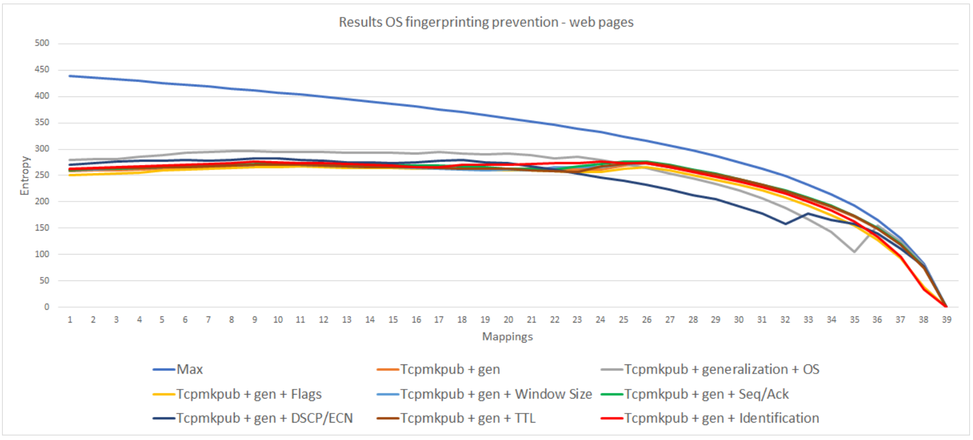


Figure A.17: Average entropy vs. max entropy for OS fingerprinting prevention - web pages

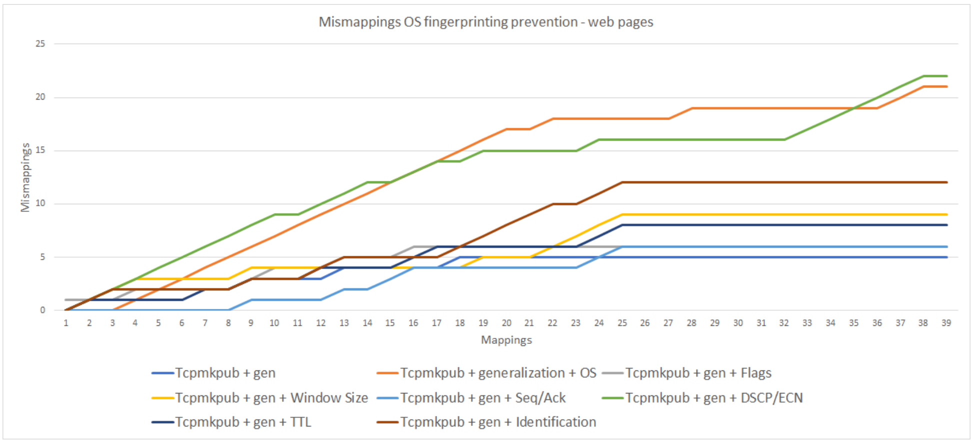


Figure A.18: Mismappings for OS fingerprinting prevention - web pages

Appendix B
SafePcap vs. Crypto-PAn

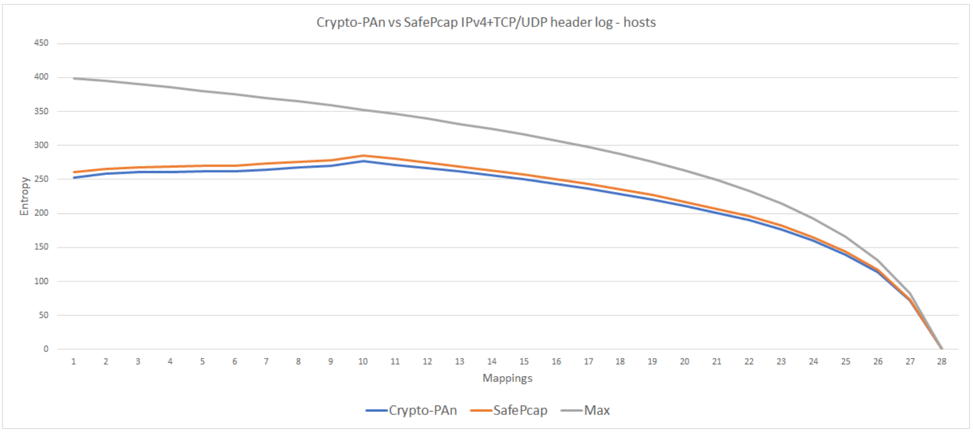


Figure B.1: SafePcap vs. Crypto-PAn - hosts

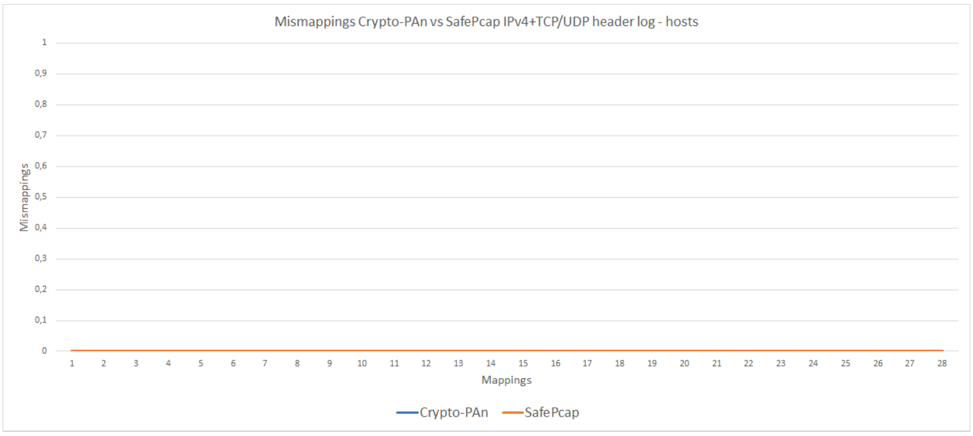


Figure B.2: Mismappings SafePcap vs. Crypto-PAn - hosts

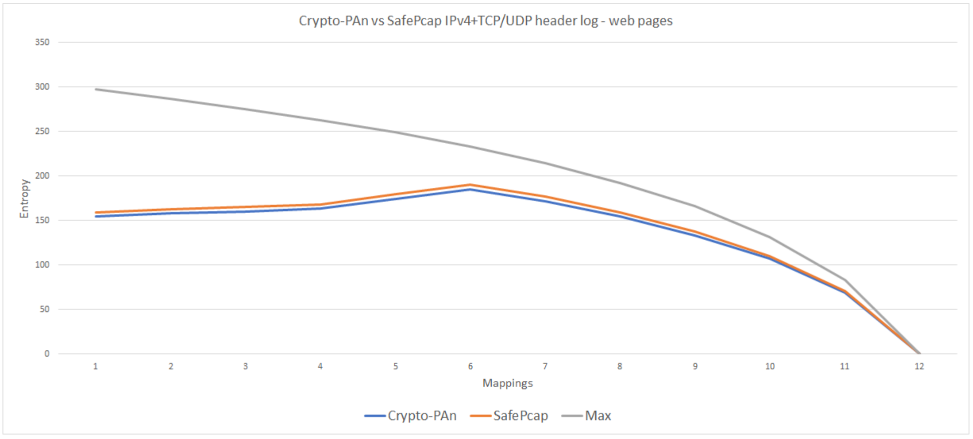


Figure B.3: SafePcap vs. Crypto-PAn - web pages

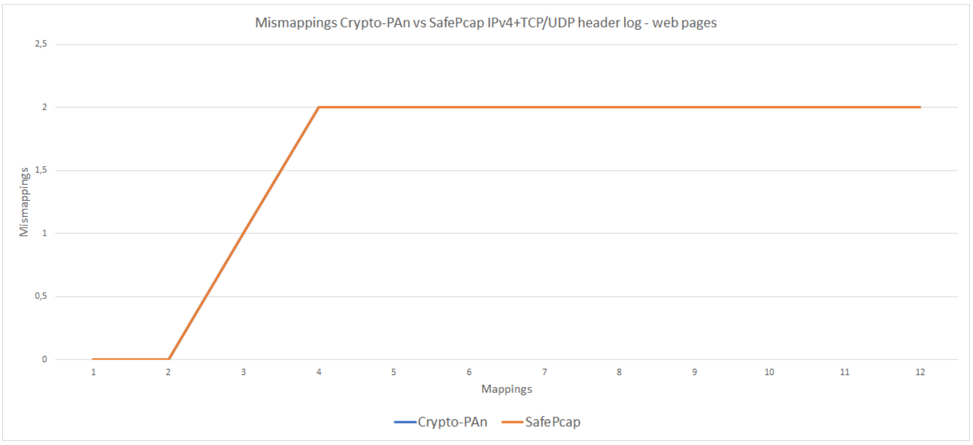
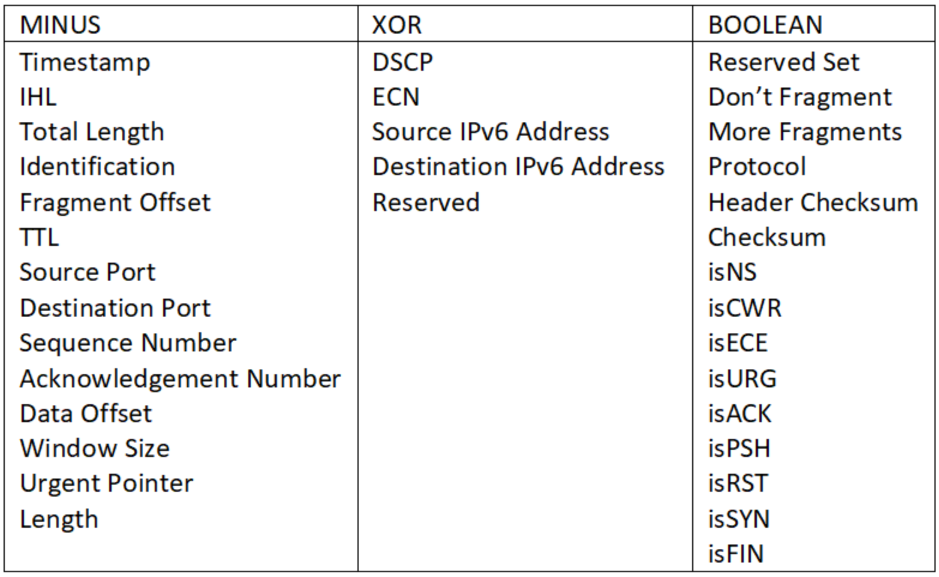


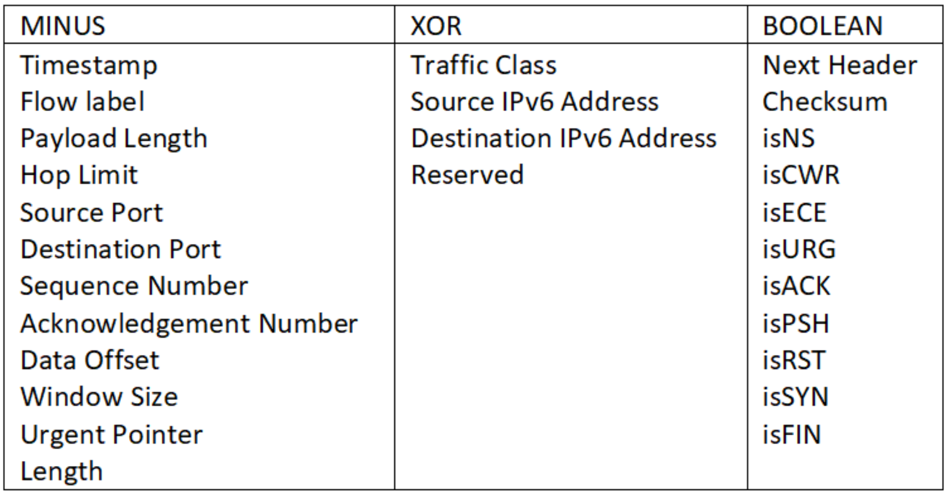
Figure B.4: Mismappings SafePcap vs. Crypto-PAn - web pages

Appendix C
Comparing fields for Inter-records



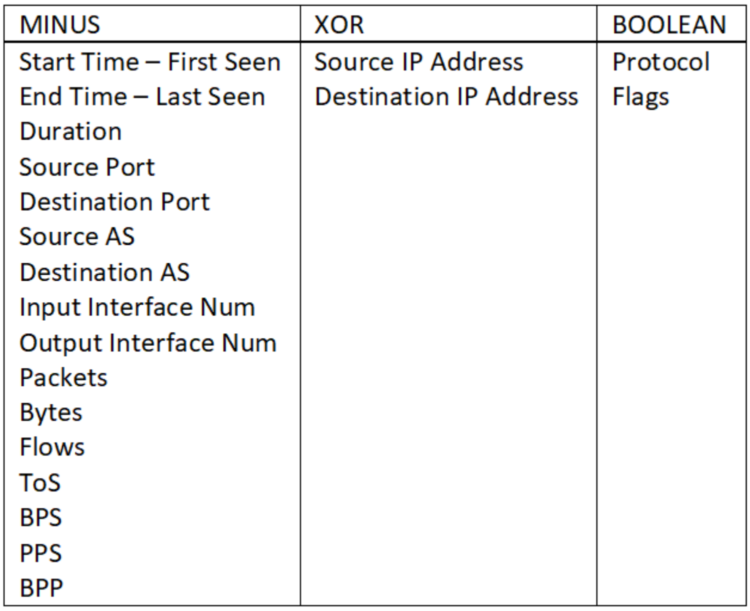
MINUS	XOR	BOOLEAN
Timestamp	DSCP	Reserved Set
IHL	ECN	Don’t Fragment
Total Length	Source IPv6 Address	More Fragments
Identification	Destination IPv6 Address	Protocol
Fragment Offset	Reserved	Header Checksum
TTL		Checksum
Source Port		isNS
Destination Port		isCWR
Sequence Number		isECE
Acknowledgement Number		isURG
Data Offset		isACK
Window Size		isPSH
Urgent Pointer		isRST
Length		isSYN
		isFIN

Figure C.1: How to compare fields for inter-records - IPv4+TCP/UDP header log



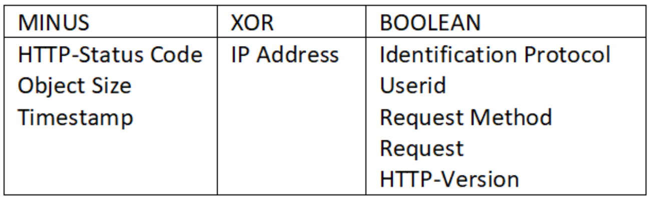
MINUS	XOR	BOOLEAN
Timestamp	Traffic Class	Next Header
Flow label	Source IPv6 Address	Checksum
Payload Length	Destination IPv6 Address	isNS
Hop Limit	Reserved	isCWR
Source Port		isECE
Destination Port		isURG
Sequence Number		isACK
Acknowledgement Number		isPSH
Data Offset		isRST
Window Size		isSYN
Urgent Pointer		isFIN
Length		

Figure C.2: How to compare fields for inter-records - IPv6+TCP/UDP header log



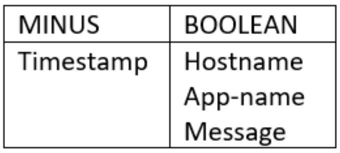
MINUS	XOR	BOOLEAN
Start Time – First Seen	Source IP Address	Protocol
End Time – Last Seen	Destination IP Address	Flags
Duration		
Source Port		
Destination Port		
Source AS		
Destination AS		
Input Interface Num		
Output Interface Num		
Packets		
Bytes		
Flows		
ToS		
BPS		
PPS		
BPP		

Figure C.3: How to compare fields for inter-records - NetFlow log



MINUS	XOR	BOOLEAN
HTTP-Status Code	IP Address	Identification Protocol
Object Size		Userid
Timestamp		Request Method
		Request
		HTTP-Version

Figure C.4: How to compare fields for inter-records - Web server log



MINUS	BOOLEAN
Timestamp	Hostname
	App-name
	Message

Figure C.5: How to compare fields for inter-records - Syslog

Petter Ødegård

Data Anonymization for Research