University of Amsterdam

Master's programme
System and Network Engineering

Research Project 2

Online event registration with minimal privacy violation

A study into Privacy Enhanced Filtering techniques

Author: Niels van Dijkhuizen
niels.vandijkhuizen@os3.nl

Supervisor: Jeroen van der Ham
jeroen.vanderham@ncsc.nl
National Cyber Security Centre NL

July 9, 2015

Abstract

In order to detect network threats, traffic inspection needs to be performed. Due to privacy concerns, results from these inspections can only be shared with semi-trusted parties after filtering and anonymisation has been applied. In this literature study, the effects of anonymisation on intrusion detection systems is performed. To understand the current state and future of network anonymisation systems, existing tools and frameworks are compared. This comparison gives us a clear view on the problematic areas of these systems. An important requirement of such a system, is that it can operate at line-rate speeds. This paper proposes two models for an anonymisation system and an approach to come to a set of privacy policies that have a close relation to the targeted threat rules.

Contents

1 Introduction

In the past 20 years, computer networks have become more widespread, interconnected and prominent to our society. This resulted among others in a growing need within various disciplines related to computer networking, to register and store real-world network traffic. To give some examples: computer network and security research need this kind of data in order to learn more about the behaviour of the network and its anomalies. System engineers and network operators need it to resolve operational incidents or problems. Organisations need to provide evidence that they comply with the current state of IT infrastructure security to avoid data leakage (the proposed Dutch law *Meldplicht datalekken*[1] illustrates this). And most important to this research project are network forensic investigators and incident handlers that need to monitor and analyse patterns of computer crime incidents.

The Dutch *Nationaal Detectie Netwerk*[2] (NDN) is a cooperation of public sector organisations that aims for better and faster detection of digital hazards and risks, by sharing threat information. In order to do this in an efficient way, an automated network threat detection and registration system is required.

1.1 Privacy concerns

A major reason why network traffic logs (often called network traces) are often not shared with third parties, is because of the concern that confidential and private information can be inferred from it. Network traffic logs or traces are built from data packet headers which exist of fields that might contain sensitive information. One of these sensitive fields is the IP address, which reveals the source and destination of each communication. This information makes it possible to determine, for example, which device was browsing a certain web server at a certain point in time [13, 39, 54].

A distributed network threat detection and registration system would typically contain Intrusion Detection System (IDS) nodes at tactical positions in the targeted networks. A common model for such a distributed system is the Hub-and-Spoke information exchange model as illustrated in Figure 1. Patterns of intruders or other network threats can not always be found by only using the packet headers, therefore the IDS must inspect all the packet data (headers and payload) to match threats [1]. This form of inspection is called Deep Packet Inspection (DPI).

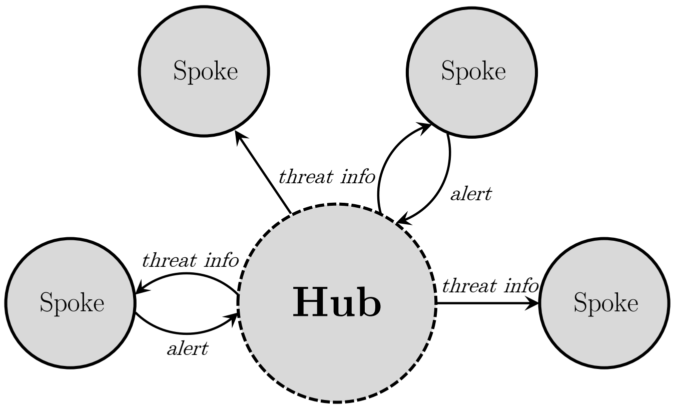

Figure 1: TAXII's Hub-and-Spoke information exchange model [10]

DPI often raises an even higher concern about confidentiality and privacy at cooperating parties / spokes, due to the deep level of inspection. For example, the content of an email could trigger a threat alarm. However, this type of data is regarded to be very privacy sensitive information [38].

[1]https://www.eerstekamer.nl/wetsvoorstel/33662_meldplicht_datalekken_en
[2]http://www.rijksoverheid.nl/documenten-en-publicaties/publicaties/2014/03/17/nationaal-detectie-netwerk.html

Monitoring network traffic or being able to query a system for a certain event often is not possible due these privacy concerns. The same issues on legal and ethical level are true for sharing complete data sets as Van Rijswijk-Deij illustrates for NRENs [48]. The purpose of this research project is to take away some objections on privacy with network detection systems by analysing anonymisation, pseudonymisation and filtering techniques.

1.2 Scope

This project is limited to anonymisation and filtering of packet headers and payload within protocols from the TCP/IP suite. Figure 2 illustrates the positioning of some of these protocols.

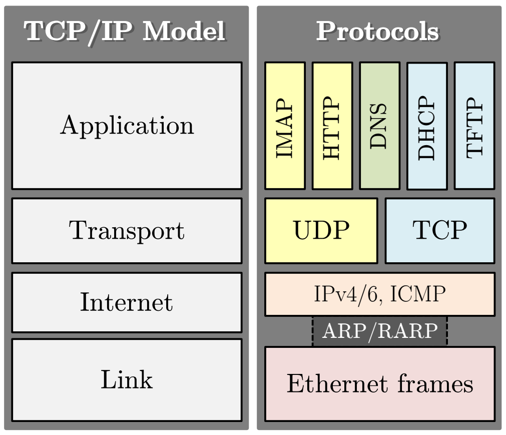

Figure 2: The TCP/IP model (*left*) and some of its protocols (*right*)

The anonymised network traffic techniques, tools and frameworks considered in this project are meant for network threat and anomaly detection as described in the introduction. After processing the network data, it may also be stored in order to make sharing and analysis possible at later times.

This project tries to indicate the feasibility of an anonymisation system, mainly in terms of processing speed. This is relevant because current link speeds[3] may impose technical limitations to such a system. Both software and hardware solutions have their strengths and weaknesses regarding processing of network data. This project will also touch upon choices in this regard.

1.3 Research Questions

There is an urgent need for open and verifiable systems that give access to useful network data, without the above mentioned privacy violation. The main research question can therefore be formulated as following:

Is it possible to create a system that indicates network threats with minimal privacy violation?

In order to answer this main research question, some sub-questions need to be formulated during the research.

- Which privacy sensitive information is present in network protocols?
- Which methods are available for filtering the privacy sensitive data?
- To what extent will anonymisation techniques influence the utility of a distributed threat detection system?
- Is it feasible for an anonymisation system to process data at current line-rate speeds?

[3]10Gbps to 100Gbps are quite common backbone speeds nowadays.

1.4 Structure

Chaper 2 will first of all explain how anonymisation techniques can be applied to network traffic. The most important concepts are covered and type of transformations to achieve anonymity will be explained. The effects and need for anonymisation within the TCP/IP layers is illustrated on a per field basis.

In chapter 3 some of the known attacks on anonymisation schemes are covered. The goal of these attacks is to infer structures or characteristics of objects, to unravel more information, untill sensitive data can be extracted from it. Some defences with this regard are mentioned as well.

To understand the current state and future of network anonymisation systems, chapter 4 will describe the development of related tools and frameworks from the past 20 years and compare these them. In this chapter, their weaknesses are highlighted. Due to the growing speeds of networks, this chapter will conclude with some of the options in performance scalability.

2 Anonymisation and Transformation

Anonymisation is the process of removing identifying particulars or details from a dataset for statistical or other use[4]. In relation to this paper, anonymisation is specificly about removing privacy sensitive information from network traffic. This is done in order to safely analyse network traffic for network threats.

Pang et al. point out in [37] that it is crucial to prevent users of network trace data to determine: (*i*) the true identities of specific hosts such that an audit trail of a certain user could be formed, (*ii*) identities of internal network hosts such that services could be mapped to specific hosts, and (*iii*) used security practices within the organisation which could leverage an attack.

As Brekne et al. denote the fine distinction between anonymisation and pseudonymisation in [6]. Anonymisation tries to achieve the state of being not identifiable within a set of subjects, while pseudonymisation is the replacement of an actual identity by an alternate identity (a *pseudonym*). A pseudonym must thus be uniquely identifiable. Bijective mappings (one-to-one correspondence) enable this.

Section 2.1 will explain some of the more important concepts used in the literature of network trace anonymisation. Section 2.2 will cover most of the network anonymisation and pseudonymisation primitives described by the literature. The effects and need for anonymisation within the TCP/IP layers is illustrated on a per field basis in section 2.3.

2.1 Concepts

Coull et al. define some important concepts in [11] that are referred to in section 4:

> "Pseudonym consistency requirement" is the requirement for consistently anonymised hardware and network addresses within and preferably across multiple traces. Without consistent anonymisation, metrics like '*the number of distinct hosts in a dataset*' or '*characterisation of connections*' can only be applied to individual traces instead of applying them to a dataset as a whole.
>
> In order to maintain utility of the network traces, it is required to keep header information of the link-, internet, and transport layers. This is called "header requirement". If this information would be destroyed the result would hold data of limited value to threat detection. For example the reassembly of TCP streams can no longer be performed, which has effect on application layer parsing.
>
> "Port number assumption" is the mapping of port numbers to well-known application layer services, which is regularly used in protocol classification schemes. Even though port numbers are not always mapped to their well-known application layer services, these

[4]Oxford definition: http://www.oxforddictionaries.com/definition/english/anonymize

services can often reliably be identified by their unique behaviour and by the use of timings and size information.

Pang et al. [38] describe the *filter-in principle*, which states that an anonymisation policy should explicitly define which fields have to stay in the clear. All other fields will be anonymised automatically. The approach of Gamer et al. [17] called "Defensive Transformation" even goes one step further. Here, in the transformation process a duplicate of the original packet chain is made. Initially this duplicate contains empty fields which must be filled with data from the original packet adjusted by the defined anonymisation primitive. There is a special primitive for non-transformations. This prevents accidental transfers of sensitive data. In his paper a more general interpretation of Defensive Transformation will be used: Information that is unknown will not be processed.

A straightforward approach to IP addresses anonymisation, is to map an original IP address to a (pseudo)random IP address in a one-to-one fashion. This however, destroys the prefix relationships among the IP addresses. This is undesirable in situations where such relationship is important (i.e. in grouping of end-points). "Prefix-relation-preserving" (often called *Prefix-preserving*) IP address permutation solves this problem [45]. Slagell [44] defines this mathematically as follows: Let τ be a permutation on the set of IP addresses, and let $P_n()$ be the function that truncates an IP address to n bits. Then τ is a prefix-preserving permutation of IP addresses if $\forall\ 1 \leq n \leq 32$:

$$P_n(x) = P_n(y) \qquad \Longleftrightarrow \qquad P_n(\tau(x)) = P_n(\tau(y)).$$

2.2 Primitives for achieving Anonymity and Pseudonymity

Replacement
Simple one-to-one replacement of a field to a new value of the same type, is a way to anonymise data. When one doesn't know the exact location or value of certain data in a network trace, this method becomes infeasible. Regular expressions however provide enough flexibility to cover both the matching and transformation of most data types. Regular expression matching is furthermore fast and can be effectively used in deep packet inspection [16]. *Regexp* have been used in several anonymisation tools as described in [38, 27, 52, 16, 32].

Reordering
'*Reordering*' or '*shuffeling*' as it is sometimes called, is the rearrangement of pieces of data (e.g. within a field). An example of this is the `AnonShuffle` primitive, used in the PktAnon tool [35].

Filtering / Data removal
By completely removing the data, information is *cut* out of a field, essentially shortening the amount of data. This is called '*truncation*' [7] or '*shortening*' [17], which does affect related field lengths. Therefore another popular way to destroy data, is by overwriting it with zeros (often referred to as '*black marking*' [44]). This can either be done completely, or partially (by Gattani referred to as '*gray marking*' [18]).

Generalisation / Reduction of Accuracy / precision degradation
'*Generalisation*' is a form of transformation where data is replaced by more general data [7]. Effectively this is often realised by '*partitioning*' information [45] (also called '*grouping*' [53] or '*binning*' [30]). An example of this is grouping of TCP/UDP port numbers into either ephemeral (>1023) or non-ephemeral (<1024) ports, by assigning a fixed value to both categories [44].
'*Precision degradation*' [14] or '*Reduction of accuracy*' [53] is comparable with selective black marking the least significant information of a data field. It is often applied to make time stamps less specific. Another example is rounding of numeric values.

Enumeration
'*Enumeration*' can generally be applied to well-ordered sets. It typically starts with a random value for the first field it has to transform and choose a greater value for each following field. This is a suitable transformation for timestamps, as they keep order, but lose precision or distance [30].

Random substitution
'*Randomising*' data of sensitive fields provides unlinkability between observations like it does with data removal [7]. It is not deterministic and will therefore yield different results on different runs. There are several ways to apply randomisation. One example is applying a bit-stream of colored noise as used in the PktAnon tool [35]. Another example is random time shifting [53]. It is however important that the used (pseudo-)random number generator (*PRNG*) is not predictable, otherwise an adversary might reveal parts of the original data by using data analysis.

Cryptographic permutation
Since the *permutation* of a set is a bijective function, it implicates pseudonymisation. A permutation can only be reversed by someone who knows how the permutation is applied. A cryptographic block cipher is a convenient type of permutation, since it needs a key as parameter to do and undo the permutation. Now one only has to know the used block cipher, including its settings and the appropriate key to repeat or undo the anonymisation. Using a cryptographic permutation makes sense for larger fields, like IPv6 addresses. It does not for small fields, because there are no strong block ciphers for them (e.g. the small IPv4 field consist of only 32 bits) [45].

Hashing
Using cryptographic hash functions for anonymisation is possible and useful for binary data and text. A difference with cryptographic permutation is that hash functions in reality have collisions, and therefore only provide semi-pseudonymisation (they are not bijective). When a field which has to be anonymised is shorter than the resulting hash, the outcome has to be truncated. Otherwise the packet analysis might get confused about the data structure. Truncated hashes have even more collisions and dictionary attacks on anonymised fields with 32 bits or less are very practical (e.g. a rainbow table containing the hashes of all IPv4 addresses) [45].

Keyed Hashing
Hash Message Authentication Codes ('*HMACs*') are specially constructed hashes generated with help of a cryptographic hash function in combination with a secret key. As with normal hash functions, collisions can occur, therefore this is not usable as pseudonymisation function. However, HMACs have better resistance to chosen plain text attacks then regular hashes [45]. Any cryptographic hash function may be used to calculate a HMAC [3].

Remarks
Several varieties of the earlier described primitives exist. They can use specific encryption or hashing functions and have different names in different tools. To give some examples: '*partial reverse truncation*' or '*byte-wise SHA1 hashing*'.

When anonymisation primitives are implemented, one should take care for correct, compatible and to some degree meaningful transformation. For example: an IPv4 address is represented by 32 bits, there are however, IP addresses with special meaning like: ⟨`0.0.0.0`⟩, ⟨`255.255.255.255`⟩, ⟨`127.0.0.1`⟩, and ⟨`169.254.x.x`⟩. Additionally every subnet has a network and a broadcast address which depend on the context. Comparable situations exist for other type of fields.

After anonymisation primitives have been applied, TCP sequence/acknowledgment numbers (if used) and IP packet length fields should be corrected to reflect the proper data lengths. Then all relevant checksums should be recalculated [38]. If this action is skipped, packets might be seen as incorrect by the IDS.

2.3 Field sensitivity

Sections 2.3.1, 2.3.2 and 2.3.3 will cover the link, internet and transport layer fields that are related to security and privacy. For each of the relevant fields a brief description of its purpose is given. What information could be derived from a field and whether a field can be abused for malicious purposes is described. Finally when anonymisation has effect on the IDS, this is will also be indicated.

2.3.1 Link Layer

The link layer is the lowest layer of the TCP/IP suite. It deals with the protocols and methods that operate on the physical link of a host. Since IDS probes often reside at local area networks, Ethernet is assumed for this layer. Figure 3 shows the structure of an Ethernet frame. Figure 4 shows the structure of the Address Resolution Protocol (ARP) which is used to convert a IP address into a physical address (in this case an Ethernet MAC address).

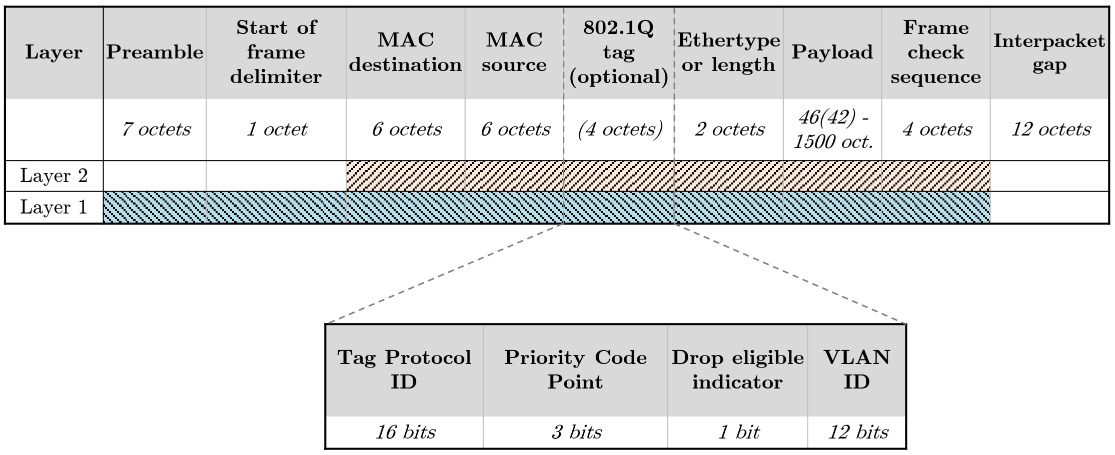

Figure 3: Ethernet frame and 802.1Q VLAN tag

MAC address
A MAC address identifies a network interface. It consist of six bytes of which the upper three bytes contain the Vendor Code and the lower three are unique for each interface. MAC addresses were once meant to be globally unique. However, virtualisation and the sheer number of interfaces changed this [45]. Keeping the semi-unique MAC addresses in traces would allow adversaries to track behaviour of a certain user if they separately obtain that users MAC address. Furthermore, it allows them to attack other parts of an anonymisation scheme (like the way the IP address is transformed) [37].

Several different primitives have been suggested and used for MAC addresses anonymisation:

- Random permutation of only the lower three bytes (vendor code preserving) [37, 5];
- Independent consistent permutation of the vendor code and the lower three bytes [37];
- Random permutation of the entire MAC address [37, 45, 5];
- Black marker [45, 32] (implemented in Anontool by Foukarakis et al.);
- Truncation [45, 14];
- Hashing [17].

One should consider when leaving the Vendor code in tact in an environment with a very non uniform Vendor distribution among Network Interface Controllers (NICs), this might lead to reidentification [37]. The anonymisation primitive one should choose therefore depends on the topology and size of the Link Layer Domain (LLD). When a tap is used in a very small LLD, the MAC addresses will reveal less valuable information when compared to a large and complex LDD.

A security analyst might want to know whether a NIC belongs to a network device (e.g. a router), a server NIC or a workstation NIC. Partitioning could be a safe and useful option for vendor code transformation. All known interfaces of a certain class are transformed to one specific vendor that represents the class. Unknown vendors are grouped to the unknown class (could be ⟨`00 00 00`⟩ for example). The lower part of the MAC address could be derived from the full original MAC address with help of a keyed hash function.

Frame Check Sequence
The FCS field contains the result of a Cyclic Redundancy Check (CRC) on the frame data. When

the CRC doesn't match the frame data, the Ethernet packet should be discarded. If data within the frame is anonymised, the CRC value in the FCS field has to be recalculated as well. However, when the frame had a bad CRC before the anonymisation, this fault should be propagated. [5]. Bad checksums might be interesting to the IDS as well.

VLAN ID
When VLAN (IEEE 802.1Q) tagging is used, the VLAN-ID might reveal structures of the internal network to an adversary. It is therefore recommended to either remove the VLAN tags completely (TraceWrangler and tcprewrite have this option), or transform the VLAN-ID (PktAnon has this option) in such a way that topology cannot be inferred from it.

Octet offset	**0**	**1**
0	Hardware type	
2	Protocol type	
4	Hardware address length	Protocol address length
6	Operation	
8	Sender hardware address	
10		
12		
14	Sender protocol address	
15		
18	Target hardware address	
20		
22		
24	Target protocol address	
26		

Figure 4: Internet Protocol (IPv4) over Ethernet ARP packet

Address Resolution Protocol
ARP is used for resolution of internet layer addresses into link layer addresses. In many cases this is the resolution of an IP address into a MAC address. When MAC and IP addresses are anonymised, the ARP protocol has to be taken into account as well (unless it is filtered out completely). The anonymisation scheme used on the hardware and protocol addresses of the ARP packet should preferably be the same as the one used outside the scope of the ARP packet. After all, inconsistent anonymisation might cause extra alerts at the IDS [37].

2.3.2 Internet Layer

The internet layer is designed to route packets across networks by means of IP addresses. Figure 5 shows the structure of the Internet Protocol version 4 (IPv4) header and figure 6 shows the structure of the newer IPv6 header. This section also covers the Internet Control Message Protocol (ICMP).

Offsets	**Octet**	**0**								**1**								**2**								**3**							
Octet	**Bit**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	**0**	Version				Internet Header Length				Differentiated Services Code Point						ECN		Total Length															
4	**32**	Identification																Flags			Fragment Offset												
8	**64**	Time To Live								Protocol								Header Checksum															
12	**96**	Source IP Address																															
16	**128**	Destination IP Address																															
20	**160**	Options (if IHL gt 5)																															

Figure 5: IP version 4 header

Differentiated Services Code Point
The Differentiated Services (DiffServ) and Explicit Congestion Notification (ECN) fields inside the Differentiated Services Code Point (DSCP) have replaced the older *Type of Service field* (ToS). These fields are used for *Quality of Service* (QoS) and Congestion Notification. Previous

work has shown that the ToS field can be used to identify types of routers and that user-defined fields can reveal user behaviour [53]. Furthermore this field can be used for covert channels. Since the Differentiated Services field is to some degree compatible with ToS, reidentification with it might still be possible. Replacing the 8-bit DSCP field with `0x00` would be a safe option for an IDS system to avoid reidentification.

Total Length
The total length field contains the length of the IP datagram including headers and data. This field is important for security analysis, because packet length in combination with other packet properties may indicate certain malware payload. Examples of these fixed size malware communications are the Slammer Worm (404 bytes) and the Nachi Worm (92 bytes) [53]. An IDS might also detect buffer overflows with help of this field. Yurcik et al. [53] state that anonymising this field with pure or keyed randomisation gives the least amount of false positives on a default Snort IDS install when the available primitives are randomisation, black marker and grouping.

Identification
The identification field is used to identify groups of IP datagrams that are fragmented but yet belong together. This is needed for proper IP reassembly. The IP identification sequence generating algorithm in many operating systems leaves a fingerprint. NMAP for example retrieves this identification field fingerprint with TCP SEQ probes, TCP probes send to closed ports and multiple ICMP responses [33]. Normally IP reassembly is performed at the IDS as well. Therefore it it is recommended to generate new random values for the IP IDs and assign these values to fragments that belong together. Some operating systems have truly random IP IDs, here the suggested transformation might be an overkill.

Flags
The IP flags (sometimes referred to as *fragmentation flags*) consist of three bits: “Must Be Zero (bit 0: MBZ), “Do not Fragment (bit 1: DF) and “More Fragments (bit 2: MF). If the DF flag is set, a router is requested not to fragment the sent datagram. If the router is not capable of doing this, the datagram is dropped (and optionally returns a “destination unreachable, fragmentation needed message). The DF flag can also be used to fingerprint a device or machine [33]. The MF flag is cleared for unfragmented datagrams and set for all fragments (except for the last one). An IDS might use these flags to recognise some old Denial of Service attacks like the Jolt attack, the Teardrop attack and some IGMP attacks.

Time To Live
The TTL field indicates how many hops from a node a packet is allowed to travel. At each node the value is decremented by one. Not all operating systems have the same default initial TTL value. Therefore, this field might reveal the used OS of a computer or network device. The TTL field can also be abused as a covert channel, which might be detected by an IDS. When privacy is more important then security, the TTL might be set to a fixed value (i.e. 10) or random values greater than 4. Very low TTL values can cause unwanted alerts at the IDS.

Header Checksum
The checksum field is used for error-checking of the IP header (like the Frame Check Sequence does with Ethernet). When a router receives a IP datagram, it calculates its checksum and verifies if the checksum is correct. When it is, the TTL is then decreased to denote a hop and the checksum with the new TTL is recalculated before the packet is sent to the next hop. If the checksum is incorrect at arrival, the packet is discarded. This field should not be anonymised, but rather corrected if any of the IP fields is anonymised under the condition that the checksum was correct in the first place.

Source and destination IP address
Privacy-wise, the IP address is the most important identifier for the internet layer. In contradiction to the MAC address at the layer below, this ID can travel much further (i.e. across the internet, depending on the address range). This is a privacy sensitive field, because IP addresses can be traced to machines which allows to form user trails. There are many tested and proposed methods to anonymise an IP address. However, when used with an IDS only anonymisation primitives that transform the original IP address into another one will be useful [29]. A good transformation scheme should not use invalid IP addresses.

These primitives have been proposed and/or used:

- Truncation (complete and partial)
- Black / White marker
- Random permutation (complete)
- Hashing (plain and keyed)
- Prefix (relation) preserving:
  - Prefix or Host permutation
  - Prefix preserving map (enumeration - TCPdpriv)
  - Crypto-PAn [13]
  - Fast Restorable Anonymization Algorithm [54]
  - Length prefix-preserving [31]
- Top-hash Subtree-replicated Anonymisation [39]
- (j, k)-obfuscation [40]

One might consider to anonymise only the IP addresses of the internal network, before forwarding the network traffic to the IDS. Optionally, the IP addresses of the external network can then be anonymised. This allows the use of IP blacklists at the IDS so that, for example, botnet activity can be detected.

Brekne et al. [6] mention that injection attacks can be successfully applied for reidentification of pseudonymised IP address in general. Since hash collisions are negligible[5], hashing can also be considered as semi-pseudonymisation. However, prefix preserving schemes are vulnerable to quicker reidentification of other hosts in a subnet, once one host's true identity is revealed.

Coull et al. [11] suggest to employ non-static pseudonyms for IP addresses, essentially breaking the one-to-one relation. This idea is worked out by Riboni et al. in [40]. They enforce a many-to-one mapping among IP addresses and pseudo-random group identification values. These groups are formed of nodes with comparable characteristics to avoid reidentification of nodes in the group by fingerprinting techniques.

IP Options

When the IP Header Length is greater than five, this indicates the option field is present and should be considered. This field can be used for OS fingerprinting and covert channels. Because of this, it might be suitable to use it in injection and probing attacks [45]. A safe anonymisation technique would be to fill the Option field with *No Operation* (NOP) values terminating the option field with a *End of Options List* (EOOL)[6], especially because this field can contain IP addresses as well (e.g.the record route option) [37].

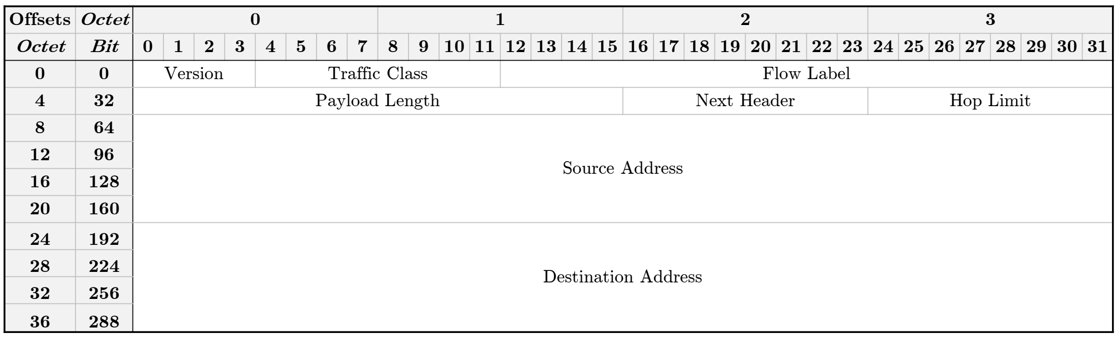

Figure 6: IP version 6 header

IPv6

Not much research has been performed on IPv6 packet anonymisation other than the IPv6 address fields. The address space of IPv6 is 2^{128} versus 2^{32} with IPv4. Patches for Crypto-PAn to

[5]Assuming recent cryptographic algorithms in the context of IP anonymisation
[6]For all IP Options see: http://www.iana.org/assignments/ip-parameters/ip-parameters.txt

support IPv6 have been written by Harvan & Schönwälder [20]. The Fast Restorable Anonymization Algorithm, proposed by Zhang et al. [54] also supports IPv6 address anonymisation by design.

The considerations for the IPv4's DSCP and ECN fields is the same for the IPv6 Traffic Class field. The Hop Limit can be considered the same as the Time-To-Live field in IPv4 packets.

The Flow label and Next Header are not yet covered by the current literature in the context of anonymisation.

Octet offset	**0**	**1**	**2**	**3**
0	Type	Code	Chechsum	
4	Rest of Header / Message body			

Figure 7: Internet Control Message Protocol

Internet Control Message Protocol
An ICMP message (also called control message) consists of a Type and Code field and in some cases a message body as can be seen in figure 7. There are Informational and Error messages defined in the protocol. However, the control messages of ICMPv4 and ICMPv6 are not compatible. Yurcik et al. [53] show that the timestamp in control messages can be used to detect clock skew of a device. Kohno et al. [26] and Lyon [33] describe OS fingerprinting methods with this protocol. Furthermore host names and IP addresses might be embedded in the payload (e.g. ICMPv6 Node Information Query/Response, and ICMPv4 Redirect message). ICMP can also be used as covert channel (like PingTunnel[7]). Botnets might also abuse ICMP for Distributed Denial of Service attacks (e.g. Stacheldraht).

There is no zero-sum tradeoff between security and privacy in this case. One might however verify the applied security policy within the organisation the anonymisation has to be applied for (i.e. whether ICMP traffic is filtered between subnets or not). If ICMP is allowed including pass-though of ICMP payload, the IP and host/domain name anonymisation scheme should consistently transform these in the ICMP packets too.

When other ICMP fields have been anonymised, the checksum should be recalculated if the checksum was correct in the first place (as described with IP checksums).

2.3.3 Transport Layer

The transport layer provides end-to-end communication services. The most common transport protocols are the User Datagram Protocol (UDP) and the Transmission Control Protocol (TCP). The first is a stateless communication protocol, the latter a more complex but reliable connection-oriented protocol. Figure 8 shows the structure of both protocol headers.

TCP/UDP port fields
Together, IP addresses and Transport layer ports form internet sockets. With both UDP and TCP, ports are used with communication between two hosts (source and destination). Lakkaraju et al. [29] empirically found that the utility of the anonymised network traces on an IDS, is more impacted by the fields than by the applied anonymisation primitive. Transformation of port numbers and IP addresses result in the highest loss of utility. This is due to the underlying port number assumption (briefly described in section 2.1). Snort also makes these assumptions, for example it assumes `FILE_DATA_PORTS`, `FTP_PORTS`, `HTTP_PORTS`, `SSH_PORTS`, etc. at specific port numbers. Most malicious traffic also follows the port number assumption according to [53]. Therefore the port numbers are one of the more important fields to security analysts.

When only a limited set of nodes run a specific service, this can lead to reidentification. In this situation, it might be better to apply filtering of that specific traffic.

TCP Sequence number
The TCP sequence number is used to correctly order segments that may be received in the wrong

[7]PingTunnel: http://www.cs.uit.no/~daniels/PingTunnel/

Offsets	*Octet*	0								1								2								3							
Octet	***Bit***	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	**0**	Source port																Destination port															
4	**32**	Length																Checksum															

Offsets	*Octet*	0								1								2								3							
Octet	***Bit***	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	**0**	Source port																Destination port															
4	**32**	Sequence number																															
8	**64**	Acknowledgement number (if ACK set)																															
12	**96**	Data offset				Reserved			NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
16	**128**	Checksum																Urgent pointer *(if URG set)*															
20 …	**160** …	Options *(if data offset gt 5. Padded at the end with ”0” bytes if necessary)*																															

Figure 8: UDP header (*top*) and TCP header (*bottom*)

order and to eliminate duplicates. Initial client connection requests need to be answered with a positive acknowledgement containing an initial server sequence number. This is part of the three way TCP handshake.

Initial TCP sequence numbers are sensitive to passive OS fingerprinting [45, 33] and may be guessed by an adversary (so called TCP sequence prediction attack). This field can also be abused for covert channel operations as described by SANS[8]. An IDS might have rules that contain certain specific sequence numbers, the current community rules of Snort currently count only one (INDICATOR-SCAN ipEye SYN scan). However, stream preprocessor (TCP reassembly) may also detect anomalies. Anonymisation of the sequence number should ideally follow the pseudonym consistency requirement and the transformation should disable passive OS fingerprinting.

TCP Acknowledgement number

The acknowledgement number is used (also in the three way handshake) to tell the receiving party a packet is received. Since this number can be derived from the sequence number, it holds many of its properties. It is therefore advised to anonymise this field together with the sequence number to get a concise anonymisation.

SANS described a method[8] to use this field as a covert channel.

TCP reserved bits

These three bits can be used for covert channels. No further references to anonymisation are found for this field.

TCP flags

The nine TCP flags (also referred to as control bits) are used for TCP connection dynamics. These flags play a role in security analysis, since they may indicate a Denial-Of-Service attack or malware backdoor (e.g. WinCrash and Hack-a-tack, as found in the Snort community rule-set). Due to different interpretations of the TCP protocol, the use of the TCP flags is different per OS. Therefore, the TCP flags can be used for OS fingerprinting [53].

TCP Window Size

The window size field specifies the upper boundary of the amount of bytes a sender can transmit before it receives an acknowledgement. Some operating systems have a limited number of values for the initial window size, therefore this field can be used for OS fingerprinting [33]. When no specific window size threat rules are applied, this field can safely be anonymised. The current Snort community ruleset does not contain any of these rules at this moment.

UDP and TCP checksums

When other UDP or TCP fields have been anonymised, the checksum should be recalculated if the checksum was correct in the first place. This way a bad checksum might still trigger an alarm at the IDS.

[8]http://www.sans.org/security-resources/idfaq/covert_chan.php

Urgent Pointer
If the urgent flag (URG) is set, the urgent pointer indicates the offset from the sequence number that contains the last urgent data. When an application needs to send urgent data, it gets priority over non-urgent data that needs to be sent. This field can be used as part of a covert channel [15].

TCP Options
The TCP options field is a variable length field that can contain many options. This field is moderately sensitive to OS fingerprinting. The author of the anonymisation tool *TraceWrangler* [5] Jasper Bongertz, referred in his blog[9] to the use of client IP addresses in the TCP options field. Even though the use of this is not authorised by the Internet Assigned Numbers Authority (IANA), it is known to be used. Because this field is variable in size, it offers options for covert channelling and could therefore be useful in data injection and probing attacks [44]. Some operating systems allow TCP timestamps in the option field, these could indicate clock skew, which could allow reidentification of a specific machine [26].

Pang et al. [37] plea for anonymisation primitives per TCP option, while others rather apply black marking or truncation. It might also be an option to overwrite specific TCP options with NOPs, since `0x00` (as used with black marking) officially means "End of Option List" and therefore creates artifacts.

2.3.4 Application Layer

The application layer caries application protocols like HTTP, FTP, IMAP, and so on. Since every application protocol has its own structure (headers and payload), this layer cannot easily be parsed as a whole. Either regular expression are used to anonymise data in a protocol intendant fashion, or parsing of specific protocols (by filtering) is performed. Streams introduce another intricacy. Data in the application protocol can be spread over multiple TCP packets. TCP reassembly has to be performed in order to parse a complete application level unit as a whole. In other words, if streams can be parsed, it is possible to use IDS rules that contain patterns that apply to multiple TCP fragments.

Seeberg et al. [43] proposed a novel classification scheme for anonymisation of HTTP traffic. They classified the various HTTP header fields:

"**must**"	indicates the greatest potential to identify a subject;
"**should**"	indicates a medium potential to identify a subject;
"**could**"	indicates a low potential to identify a subject;
"**no**"	indicates no identification of a subject can be performed.

A Snort ruleset was then analysed by Seeberg et al. to find rules that contained related HTTP header fields. When a rule for a field classified as "must" exists, this field is anonymised. Fields with the classification "should" and "could" as anonymised depending on the policy that is used. Seeberg defined the policies "strongest", "strong" and "weak". The IDS effectiveness after anonymisation with these policies is measured to be 42%, 80% and 100% respectively.

This method could be applied to other TCP/IP layers or application protocols as well.

3 Reidentification attacks

During the development of anonymisation tools and frameworks, several attacks have been developed in parallel to reidentify objects or patterns in anonymised data. King et al. created a taxonomy of attacks against anonymisation in [23]. Burkhart et al. followed and created the distinction between inspection attacks and injection attacks in [8]. Figure 9 illustrates the combined taxonomy of their work.

[9] https://blog.packet-foo.com/2013/07/trace-file-sanitization-for-network-analysts/

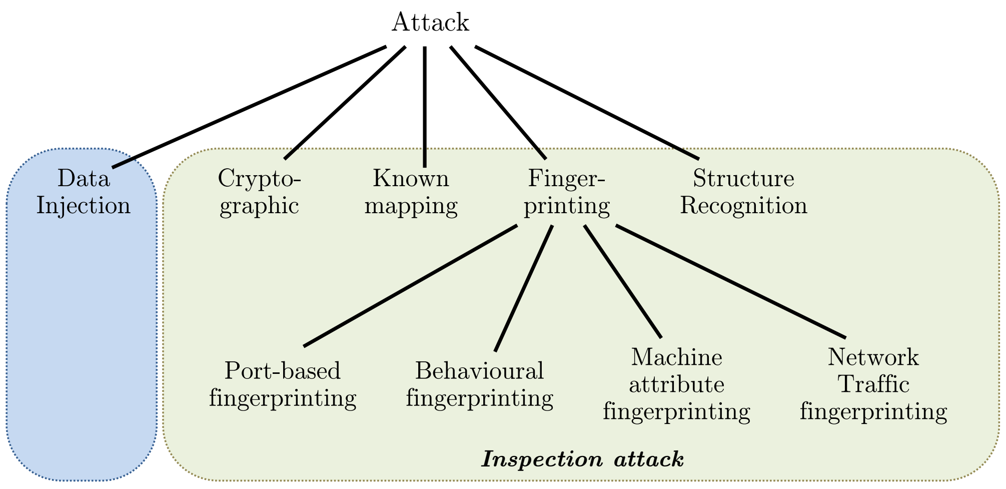
Attack
Data Injection
Crypto-graphic
Known mapping
Finger-printing
Structure Recognition
Port-based fingerprinting
Behavioural fingerprinting
Machine attribute fingerprinting
Network Traffic fingerprinting
Inspection attack

Figure 9: Combined attack taxonomy of King et al. and Burkhart et al.

Inspection attacks
Inspection attacks are those attacks that can be executed after the anonymisation process has been finished without prior knowledge of nor access to the infrastructure of the related dataset. However, these attacks do not nescessarily have to be passive; public information (like DNS and WHOIS records) can leverage an inspection attack. Also scanning and probing could be performed, once the anonymised data is published.

Matching of attributes of anonymised objects against attributes of objects that are already known, is called *fingerprinting*. When multiple mappings can be recognised in the anonymised dataset, this is called *Structure Recognition*. *Known mapping* tries to discover a mapping between un-anonymised and anonymised data within a dataset. When cryptographic anonymisation is used, a *Cryptographic attack* tries to infer the used cryptographic key or hash function that was used.

Data injection attacks
This is the act of injecting information to be logged, so that it may be recognised after the anonymisation process. This is comparable with a so called “chosen plain-text attack” against a crypto-system. Burkhart et al. [8] refer to this attack as a *privileged* one, because the adversary has more insight than an external observer. Their experiments show that it is easy for an adversary to make the injection attack more recognisable (by lengthening the injection pattern), but harder for the protector of the data to defend.

Perspective on PETs
Privacy Enhanced Technologies (PETs) enable users to hide content and addresses of visited websites for external observers by means of encryption. A well known PET is The Onion Router software[10]. Herrmann et al. [22] present a novel method that applies text mining and analysis of normalised frequency distribution of IP packets to infer visited sites from PET enabled communication.

Their “Multinomial Naïve Bayes” classifier approach shows that even packet-size frequencies could already leak some information.

3.1 Protection against data injection attacks

Because the data injection attack impose an active threat to the anonymisation scheme, some mitigations to this type of attack have been proposed. Brekne et al. suggest to use sampling in order to make attacks more costly [6]. If only parts of the original injected pattern are available in the anonymised dataset it will take longer for an adversary to discover and recover this pattern.

[10]TOR project: https://www.torproject.org/

Brekne et al. also suggest to prevent packet injection attempts. If the injected data can be filtered in advance, analysis on the anonymised dataset will never reveal the original pattern. Pang et al. [37] apply this for internal (benign) scanning devices. It is however, difficult to completely mitigate these injection attacks, because patterns may be embedded in what appears to be benign communication.

Finally Riboni et al. [40] propose an anonymisation method which uses non bijective mappings. Objects with a certain characterisation or fingerprint are grouped together. This way the pattern gets blurred into groups and one-to-one relations can therefore no longer be inferred by an adversary.

4 Tool development

Section 4.1 is an exposition of the most influential tools and frameworks regarding anonymisation of the past 20 years. Section 4.2 will briefly cover some of the less important anonymisation tools and in section 4.3 a comparison of the tools is made. The identified problematic areas of the tools and frameworks are covered in section 4.4. Due to the growing speeds of networks, section 4.5 will conclude with some of the options in performance scalability.

4.1 Timeline of anonymisation tools

TCPdpriv
TCPdpriv is written by Greg Minshall of Ipsilon Networks in the mid nineties and is one of the first publicly available anonymisation tools for network traffic.

It anonymises traces by rewriting packet header fields like IP addresses and port numbers. In addition it strips off TCP and UDP payload. Non TCP/UDP traffic IP payload is discarded completely [16] [MAN page].

This tool has no extension options and is limited in configuration. Moreover, it is not possible to map arbitrary data fields to anonymisation primitives [17].

The IP address anonymisation primitives vary from no changes (`-A99` option) to sequential numbering per unique IPv4 address (`-A0` option). In order to get a good balance between the usefulness and privacy of a trace, the tool has a prefix-preserving (`-A50`) option, which anonymises IP addresses but preserves the prefix nature of them [39].

The prefix-preserving anonymisation implementation of TCPdpriv is table based. It stores pairs of raw and anonymised addresses in memory to maintain consistency (one-to-one mapping) of the anonymisation. When a new address needs has to be anonymised, it will be compared to the raw addresses in the table for the longest prefix match. This method is very memory intensive and therefore not very suitable for real-time transformation. Since pseudonyms depend on the order of new IP addresses, the same IP addresses get different pseudonyms in different traces and therefore do not meet the pseudonym consistency requirement. This property makes it also infeasible for parallel processing of traces [50].

Ylonen analyses the security of the prefix-preserving anonymisation primitive used in TCPdpriv and proposes a method to break it by first identifying a host with a well-known traffic pattern [38, 51].

Crypto-PAn
Fan et al. addressed some of the problems with the prefix-preserving method of TCPdpriv. Their Crypto-PAn, for which the work was started in 2002, no longer uses table based transformation, but a stateless cryptographic algorithm [13]. This dramatically reduces the memory use. However, since 32 rounds of Rijndael encryption are needed for the anonymisation of an IPv4 address, this solution involves much more computational power. Apart from the lower memory footprint, Crypto-PAn meets the pseudonym consistency requirement and allows consistent parallel or distributed anonymisation (as long as the same cryptographic key is used).

Crypto-PAn is limited to IP address anonymisation and can not be used as a complete anonymisation framework [27]. In the original form it is not possible to anonymise IPv6 addresses, Harvan and Schönwälder made a patch to realise this [20]. Slagell et al. created a modified implementation of Crypto-PAn that contains a key generator, for easy administration. The other improvements came from David Stott of Lucent technologies, such as improved randomness, improved performance (using OpenSSL) and more levels of anonymisation[11].

Brekne et al. [7] and Coull et al. [11] address a flaw in Crypto-PAn. It anonymises IP addresses in such a way, that any bit in the anonymised address is depending on all previous bits of the un-anonymised address. Therefore if one address is de-anonymised, all other addresses that share the prefix of the un-anonymised IP address can easily be revealed.

Bro Anonymiser plug-in
In 2003 Pang et al. implemented an anonymisation plug-in for the Bro network security monitor [38] to take advantage of its application parsers and built-in language to support policy scripts. Their framework uses the filter-in principle and is able to do both on-line and off-line anonymisation. It is one of the first application level anonymisation tools available. There are several limitations to this approach. First, custom anonymisation primitives have to be written in the Bro language if the limited set of anonymisation primitives does not meet the requirements of the user. Second, Bro policy scripts have to be written for every application level protocol even if only the IP address has to be anonymised. Third, Bro is a security monitor and therefore not optimised for anonymisation in terms of speed [27]. The code of this anonymisation plug-in is no longer available, since it was included with Bro before version 0.9 (the newer `Anon.{cc,h}` are based on the ipsumdump / TCPdpriv code).

TCPmkpub
Pang et al. decided to create a general purpose multi-layer anonymisation framework, so that a wide range of privacy policies could be defined [37]. This tool called TCPmkpub was released early 2006. By default many fields of the TCP/IP layers can be anonymised, however flexibility is decreased by the fact that no generic anonymisation primitives can be applied [17]. Anonymisation methods are specific for certain field types.

TCPmkpub does not retain prefix-preserving relationship between internet and external addresses. Since external IP addresses have much less locality, they are anonymised using Crypo-PAn. The prefix and subnets of the internal network(s) have to be declared in a configuration file. For this part a modified anonymisation scheme of the TCPdpriv is used (mode `-A49`). The host part of internal IP addresses are anonymised via pseudo-random permutation (based on Luby and Rackoff). This anonymisation scheme is not as easy to break as the pure Crypto-PAn anonymisation scheme.

Some disadvantages of TCPmkpub are that it cannot be used for on-line anonymisation because it needs multiple passes over the original network trace [17], the application layer data is truncated and the policy is compiled into the binary (which makes it less flexible to use).

AAPI and Anontool
Koukis et al. [16] also worked on a flexible anonymisation framework in 2006. Their Anonymisation Application Programming Interface (AAPI) was also ment to be faster than the existing tools of that moment, while offering more anonymisation primitives [27]. Users can create their own policies using the C-like API up to the application layer [18]. The API has a large amount of anonymisation primitives varying from basic primitives to Crypto-PAn, Prefix-preserving-map, hashing and block cipher cryptography. It has support for regular expression matching and replacement and supports several input formats including Netflow and pcap [16]. However, a user cannot modify the anonymisation profile without recompiling the whole project [17].

Foukarakis et al. [16] created an open-source implementation of the AAPI, called Anontool. This implementation is more flexible because of the support for XML configuration files. Anontool supports input from Netflow / IPFix and pcap in stored format or as live feed. Output is limited to files only.

Foukarakis et al. describe shellcode detection in [16] by means of regular expressions. This kind of detection should be performed by an IDS or at least by a specialised system rather than an anonymisation framework. Lin et al. argue that the regular expression transforms of Anontool may miss sensitive data when no exact pattern can be found inside the data and therefore leak this [32]. Anontool does not meet the filter-in or defensive transformation principles.

[11]Notes from Stott: http://www.cc.gatech.edu/computing/Networking/projects/cryptopan/lucent.shtml

FLAIM
In 2006 Slagell et al. also worked on an anonymisation framework with similar objectives as AAPI/Anontool. They too created a suite of anonymisation algorithms that can be applied on sensitive fields in a network trace or log with help of XML configurations. FLAIM has been designed with modularity in mind so that it can anonymise several kind of capture or log formats [45].

FLAIM only allows certain primitives for the fields it can parse. It does not know how to handle arbitrary protocol encapsulations (e.g. IP-in-IP) and a primitive or function for checksum recalculation is missing [17]. Lastly modules for application layer protocols and real-time anonymisation output have not been developed yet.

PktAnon
In 2008 Gamer et al. developed an anonymisation framework called PktAnon. Their main requirements are: flexibility, extensibility and configurability. They built the protocol definitions into PktAnon in order to be able to parse the packets instead of relying on external parsing libraries. This way Gamer et al. implemented a clear object-oriented design. Loose coupling of network protocols (called protocol chain) is used in order to make arbitrary encapsulation like IP-in-IP possible. Defensive transformation is used in order to prevent leaking sensitive data from fields or packets that are not interpreted correctly. Finally this framework allows mapping of arbitrary anonymisation primitives to protocol attributes [17].

Together with the Anonymizer Bro plug-in of Pang et al. this is one of the few frameworks that support online anonymisation. This is realised through the use of pipes, rather than a more convenient Unix daemon or Windows service. Offline processing with PktAnon on a Intel Pentium 4 with a realistic anonymisation profile is measured to be near a 100 Mbps, which is too slow for practical online use. The authors suggest to include parallel processing or hardware-based cryptographic acceleration as future work.

PktAnon supports protocol parsing up to the transport layer and need special support to integrate application layer parsing. The authors plan to do this in the future [35].

PCAPAnon
Lin et al [32] have proposed PCAPLib. A framework consisting of an Active Trace Collector which classifies streams into benign and malicious traffic with help of commercial Devices Under Test (like Antivirus scanners, IDSs, Application firewalls, etc.) and an anonymisation module called PCAPAnon. PCAPAnon uses Wireshark dissectors to parse many of the protocol attributes and can therefore anonymise up to the application layer. It uses anonymisation primitives that preserve the semantics and length of the fields.

New in this concept is the multi domain threat detection, combined with anonymisation afterwards. Since threat detection is employed prior to the anonymisation, syntax correctness (needed with threat detection after anonymisation) is not considered important. This may however be something a security analyst may be concerned about. Packets and threats may have to be analysed, but if the dissectors get invalid field encodings, this may impede the analyst.

The PCAPLib code is available online, but the usability is still very low and the code is untidy. The code for example contains hard coded paths to developers' homedir, no clean integration with existing Wireshark libraries, missing documentation, no methods available to configure policies and so on.

4.2 Related tools

In this section some of the less known, less relevant or less innovatory tools are very briefly described.

Bit-Twist [21] and Tcprewrite (which is part of Tcpreplay [24]) are examples of tools that are suitable for anonymising and especially replaying packets. SCRUB-tcpdump [52] and TCPurify [4] have classic tcpdump-like features and syntax. NetDude [28] and TraceWrangler [5] are GUI-based utilities for packet manipulation. CANINE [44] and NFDump [19] can anonymise Netflow data. Tcpanon [47] is a python based application layer anonymisation tool. IP Summary Dump (ipsumdump) [25] can anonymise and dump packet data in human readable ASCII format.

Anonym [14] is an network data distribution analysis tool written in Matlab. CoralReef [36] is a monitoring and analysis framework that allows crypto-pan anonymisation as does TraceAnon from the LibTrace suite [2].

4.3 Comparison of tools

The comparison between the tools and frameworks of sections 4.1 and 4.2 is presented in a spread-sheet. This spread-sheet is split over two pages and can be found in Appendix A. The first page contains general information such as the name of the developers and organisations, related articles, development period and latest available version. The second page contains the actual comparison. Colours in the sheet indicate support. Green indicates full support, red indicates no support and amber indicates partial support or exceptions. When a tool supports anonymisation of a certain type, the primitives it supports are also given.

The compared software has been developed somewhere in the past 20 years and the quality of the code may differ from product to product. There may also be dependencies on external libraries. Therefore it is tested whether the source-code compiles without too much problems. The Debian 8.1 GNU/Linux distribution[12] is used as a solid testbed for this check. Results can be found under "code state" in the sheet.

As described in section 4.1, the IP address has been one of the fist candidates for anonymisation. Both IPv4 and IPv6 addresses should be anonymisable by the tool. Furthermore the obvious MAC address en UDP/TCP port numbers should be supported. Some of the less obvious fields that need anonymisation support are VLAN tags (at the link layer) and IP/TCP Options (at the internet and transport layer respectively).

Support for header checksum correction is highly recommended, since IDSes should still "see" valid un-anonymised packets as valid after anonymisation. Methods for anonymisation at the application layer should be supported, because this layer has the highest probability of containing privacy sensitive information (in case the traffic is not fully encrypted). Whether anonymisation of tunnels or arbitrary encapsulation is supported is also tested.

Because of high volumes and link speeds, real-time or online anonymisation is very important. Whether existing code can be reused in new projects or frameworks, depends on the license that applies to the software. The comparison indicates open-source type of licenses (GNU and BSD like) as green and proprietary ones as red, when there is no clear licencing information available this is indicated with amber.

Finally a score is given to every product. Fields with the colour green are given a value of 1, amber 0,5 and red or unknown 0. It must be noted that some tests or checks could not be performed because either the code was not available (AAPI, Bro anonymiser, CANINE, FlowScrub) or documentation was missing (Anonymizer). In that case the final score is grey coloured. The score is shown as percentage of total support coverage. When below one third the tool gets a red indication, between two third and one third it gets an amber indication and above two thrid it will be coloured green.

PktAnon (87,5%), Anontool (79,2%) and PCAPAnon (75,0%) have the best scores and are the only ones to be green.

4.4 Problematic areas

Of the tested tools PktAnon is the only anonymisation tool that does on-line anonymisation. AAPI and Bro with the anonymiser plug-in from Pang et al could also do this, but both are not available anymore (Anontool is, but is limited to off-line anonymisation). On-line or real-time anonymisation is therefore the biggest challenge.

The rest of the problems all have to deal with protocol interpretation / parsing. Not many tools properly support arbitrary encapsulation of protocols like IPv6 over IPv4. Application layer parsing is still problematic. Even if this is supported, this is often realised through regular

[12]Code name *Jessie* which is the current stable release

expression matching and replacing in a way that does not comply with the filter-in or defensive transformation principle. Finally IP Options, TCP Options, IPv6 and VLAN tags are poorly supported.

Implementation of anonymisation primitives is divided into two opposing camps. Gamer et al. advocate generic anonymisation primitives that can be applied to different protocol fields. And Pang et al. implemented specific anonymisation primitives for certain fields (in TCPmkpub). Generic primitives may introduce syntactical problems in transformations and too specific anonymisation primitives reduce the flexibility of the framework.

4.5 Scalability

In order to implement a scalable anonymisation system which is optimised for speed, we have to consider the use of the available algorithms. Tcpdpriv has a big impact on memory and does not allow parallel processing because of the sequential assignment of new pseudonym addresses. Crypto-PAn needs quite some computational power with its 32 rounds of AES. Top-hash Subtree-replicated Anonymisation (TSA) uses hashing for the most significant byte of an IPv4 address and a precalculated binary tree structure with subtree replication for the rest of the bytes. This binary tree is stored in memory, so lookups are used rather than calculations, which makes it faster for IPv4 anonymisation then Crypto-PAn. Since the address space of IPv6 is much bigger, TSA will not be efficient for anonymisation of these addresses. Fast Restorable Anonymization Algorithm (FRAA) does take IPv6 into account and also uses lookup techniques like TSA does. Unfortunately, the code of FRAA is not publicly available and the security and efficiency has not been reviewed by other researchers.

Multi-core processors allow parallel execution of software. AMD and Intel started to produce desktop CPU with such capabilities in 2005[13]. Nowadays many consumer devices ranging from phones to desktop computers, have multi-core CPUs. In order to take advantage of these cores, software has to be optimised to utilise them. The POSIX threading library (`pthreads`) for example, give programmers a very practical interface to do this. Snort's stream preprocessor is used to do TCP reassembly, which enables application level parsing of streams. Because the preprocessor uses pthreads, it is performing better on multi-core systems. TCP reassembly should also be considered in an anonymisation system, therefore threading is recommended.

4.5.1 Acceleration

This section describes some technologies which may accelerate on online anonymisation. Some examples of these technologies are given to illustrate their capabilities.

Specialised network capture cards
In order to solve the huge amount of interrupt handling needed at high network speeds, special purpose network capture cards can be used. Capture cards like the Emulex EndaceDAG[14] enable direct memory access to offload the CPU. A large circular buffer is used to hold arriving packets. This buffer is directly accessible from user-space by applications without making system calls. In addition they provide packet filtering, packet classification and high accuracy clock synchronisation. As of July 2015 Emulex produces capture cards with up to four 10Gb Ethernet ports.

Network Processing Unit
Network Processing Units (or Network processors) are microprocessors that are tailored to process network data. They are used in generic network devices like routers and switches, but can also be used for Intrusion Detection Systems and generic monitoring purposes. They are able to operate at wire speeds due to packet processing parallelism. In the 2000s the Intel IXP series NPUs were popular within the research community. De Bruijn et al. [12] proved it is possible to create an IDS on a IXP2400 based network card. This specific card has eight programmable micro-engines and a general purpose XScale processor running at 600MHz. Figure 10 shows the

[13]http://news.cnet.com/Dual-core-desktops-hit-the-market/2100-1042_3-5675050.html
[14]http://www.emulex.com

use of, and the relations to the micro-engines and XScale CPU. The implementation has the ability to do pattern matching comparable to that of Snort at Gigabit speed. Furthermore detection techniques at all levels of abstraction in communication are possible: packet, reassembled TCP streams, application layer protocol units, and flow aggregates.

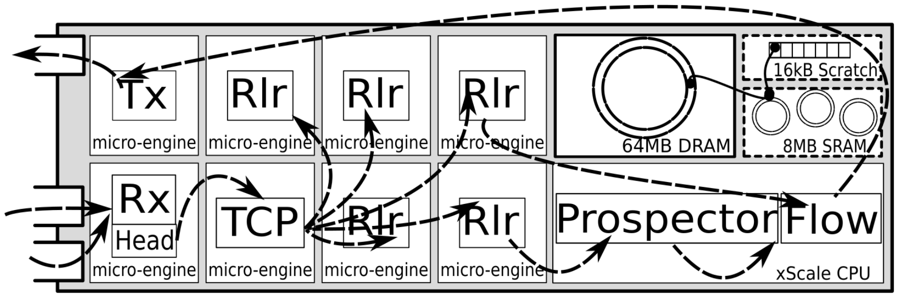

Figure 10: IDS design on an IXP2400 by De Bruijn et al. [12]

The IXP2400 was launched around 2002 and Intel sold the IXP line to Netronome in 2006. Their current state of the art are the NFP-6xxx series processors. These have an ARM11 core with 120 micro-engines and specialised hardware accelerators for DPI and cryptography. Configurations with 4 x 100GbE are available.

Field-Programmable Gate Array
A Field-Programmable Gate Array (FPGA) is a programmable integrated circuit. A so called Hardware Description Language (like VHDL or Verilog) is used to describe the functionality of the hardware. Modern high-end FPGAs have enough space and speed to contain for example multiple CPU instances (also called softcores).

Ubik et al. [46] developed a real-time anonymisation interface, based on a Xilinx Virtex-II FPGA. They implemented a prefix-preserving primitive for IP address anonymisation and support up to headers of some application layer protocols (e.g. HTTP). Their Transformation Unit (TU) is able to perform anonymisation with speeds above 1Gbps. However due to the dependancy on the SCAMPI[15] firmware these speeds could only be reached with a newer COMBO card design.

Scott [42] have developed a compact packet capture and classification module which can be used as part of a larger FPGA design. Their design was developed on a NetFPGA, but could be used on other interfaces as well. Matching is possible on arbitrary protocol headers and at the start of protocol payload. Packets of a certain class are marked with a new destination MAC address to indicate the classification. The whole design performs at wire-speed.

Majzoobi et al. [34] developed a FPGA-based True Random Number Generation using Circuit Metastability with adaptive feedback control. Their work is tested on a Virtex-5 FPGA.

Vu et al. [49] present a novel TCP reassembly technique for FPGAs that can hold 256K connections with 46K out-of-sequence connections using only 64MB DRAM. This is implemented on a Virtex-II Pro FPGA.

As of July 2015, the Xilinx Virtex-7 series of FPGAs are popular in the scientific community. This is probably due to the involvement of Stanford University and the University of Cambridge in the NetFPGA project[16]. The NetFPGA SUME is the latest NetFPGA interface which has four 10Gbps Ethernet ports. The COMBO 100G from Invea-Tech has a 100Gbps Ethernet port and is also based on the Virtex-7 FPGA.

General purpose CPUs
Modern general purpose CPUs have cryptographic acceleration capabilities. Examples of this are Intel's Advanced Encryption Standard - New Instructions[17] (AES-NI) and VIA's PadLock[18].

[15]SCAMPI project: http://www.ist-scampi.org/
[16]NetFPGA: http://netfpga.org
[17]www.intel.com/content/www/us/en/enterprise-security/enterprise-security-aes-ni-white-paper
[18]VIA PadLock: http://www.via.com.tw/en/initiatives/padlock/

Anonymisation primitives like Crypto-PAn could make use of these accelerations.

Remarks
The acceleration techniques described in the previous sections could be combined into one system. Clark et al. [9] for example combined both an IXP NPU with an FPGA to get higher throughput and more processing power. But other combinations can be made as well.

It must be noted that very specific knowledge is needed to do development on NPUs (assembly programming) and FPGAs (VHDL or Verilog programming).

5 Conclusion

In order to detect network threats, traffic inspection needs to be performed. Due to privacy concerns, results from these inspections can only be shared with semi-trusted parties after filtering and anonymisation has been applied. A lot of research is already conducted at the lower layers of the TCP/IP suite. Therefore there are quite some anonymisation primitives available. There are however different opinions on the implementation of these primitives. Some researchers advocate for general purpose primitives, while others prefer field specific primitives.

In order to anonymise network traffic such that it is still semantically and syntactically correct, many subtleties have to be taken into account. Specific IP addresses for example, can have an invalid meaning within a certain context. Apart from the anonymisation primitives the framework should also support functions for recalculating the checksums and the length field of packets. In order to prevent false positives from intrusion detection systems a layered primitive structure is recommended for maximum flexibility. This could be implemented as a small language to describe the transformations of specific parts of the traffic.

For application layer anonymisation, parsing of specific application protocols is recommended over regular expression transformations. Parsing allows the filter-in or defensive transformation principles to prevent leakage of sensitive data which was not recognised. The classification schema of Seeberg et al. can be generalised as generic anonymisation approach:

1. Identify the applications or protocols for threat detection;
2. Get statistics at the targeted network; [19]
3. Identify threats for the specific protocol/application;
4. Classify the field of the protocol/application in terms of privacy sensitiveness;
5. Build privacy policies and threat rules.

This approach enables a tight integration between the anonymisation system and intrusion detection system. Therefore a "Network native" architecture as show in figure 11 can be used. A two stage anonymisation scheme could be applied if the IDS supports this. When for example IP blacklists are used at the IDS, the primary anonymisation system only has to anonymise the internal IP addresses. After threat evaluation is competed, the external IP addresses may be anonymised as well.

If the network native anonymisation is too expensive (in computation or in development), a narrower architecture could be implemented. The "White fielding" architecture, as shown in figure 12, only transfers the fields of interest per application after the anonymisation. This could be realised by tagging the field with a classifier. Consequently, the threat matching engine has to be developed from scratch.

As of July 2015, no single anonymisation tool or framework is mature enough to perform deep packet anonymisation on all the TCP/IP layers at line rate speeds. Online anonymisation is the most problematic area, followed by the incompleteness of protocol parsing. The PktAnon framework from Gamer et al. is the most promising solution found.

[19]For example, the vendor distribution of NICs make it possible to choose an appropriate MAC anonymisation primitive

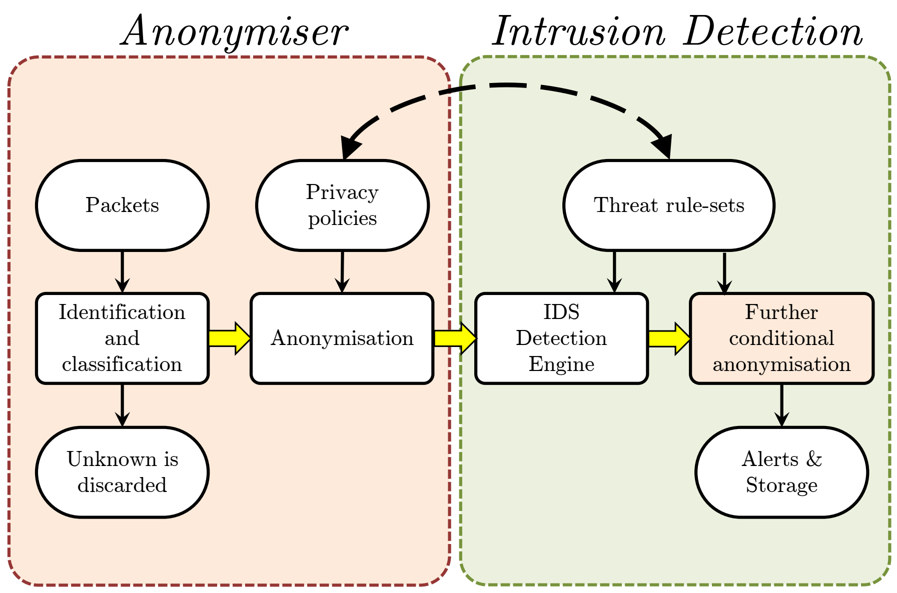

Figure 11: Relation between anonymisation and IDS in network native model

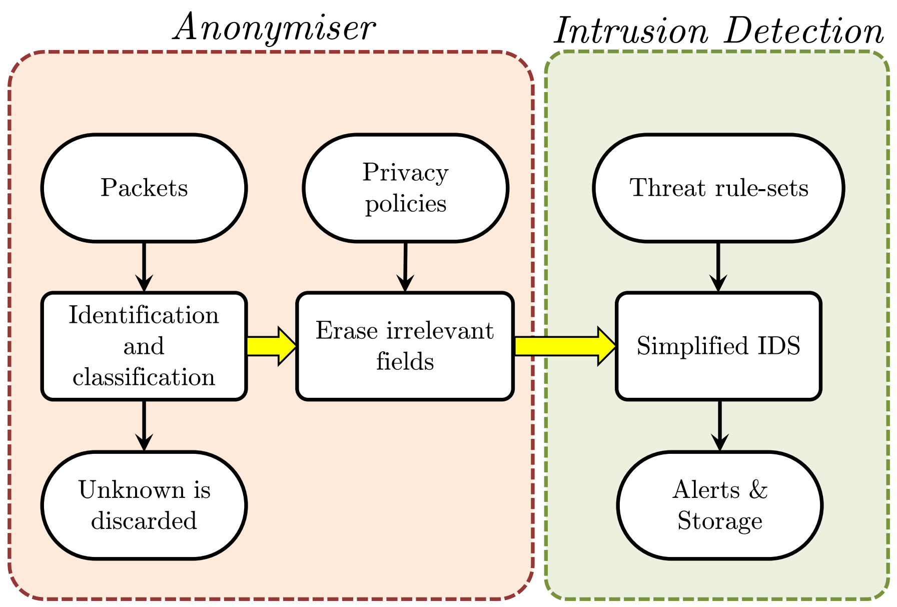

Figure 12: Relation between anonymisation and IDS in white fielding model

However, the following issues have been identified:

1. The processing speed is not high enough;
2. The flexibility of the anonymisation methods is very limited;
3. Application level anonymisation is currently not possible;
4. There is no support for TCP reassembly / disassembly.

Fortunately many hardware acceleration options are available. Since the current anonymisation software is not optimised for parallel execution the feasibility of a complete anonymisation system that supports deep packet anonymisation on all the TCP/IP layers at 10Gbps Ethernet is high.

De Bruijn et al. showed it is possible to make an IDS on a programmable Ethernet network card design from 2002 with 1Gbps, using only eight micro-engines and a general purpose CPU. This project incorporates similar techniques needed for the anonymisation system like packet inspection, reassembled TCP streams, application layer protocol units, and flow aggregates.

Modern PNUs provide many micro-engines, specialised cryptographic cores and line-rate speeds up to 100Gbps. The NetFPGA project has strong binding with the academic community and may also provide similar capabilities.

Even with a fast fully implemented anonymisation system with an expressive privacy policy definition language, the utility of the network data after anonymisation with regard to threat detection is variable. At this moment it is not possible to have network data completely anonymised while keeping the full utility for security. To achieve a good balance between privacy protection and security, cooperating parties still have to agree on the boundaries of these two areas.

Lastly, in an environment with semi-trusted parties, it is recommended not to share complete anonymised datasets. Currently the protection against anonymisation attacks is still inadequate. Though specific traces (i.e. that contain newly found threats) have less context and when anonymised with a proper privacy policy should be shareable.

5.1 PRISM

Near the end of this short research project a public deliverable of the PRIvacy-aware Secure Monitoring (PRISM) project[20] was found. PRISM is part of the Seventh Framework Programme [21] which is a funding programme for Research and Innovation in the European Union.

This PRISM document, titled "State of the art on data protection algorithms for monitoring systems" [41] was written in 2008 and has many similarities with this paper. It must be emphasised, that the results of this paper have been achieved independently. They discuss new developments with the latest insights.

6 Future work

No publications could be found of a system that performs deep packet anonymisation on all the TCP/IP layers at line rate speeds. Research on an embedded anonymisation system is therefore highly recommended.

7 Acknowledgements

I would like to thank the National Cyber Security Centre NL and particularly my supervisor dr. Jeroen van der Ham for guiding me with this short research project.

References

[1] T. AbuHmed, A. Mohaisen, and D. Nyang. A survey on deep packet inspection for intrusion detection systems. *CoRR*, abs/0803.0037, 2008.

[2] S. Alcock, P. Lorier, and R. Nelson. Libtrace: A packet capture and analysis library. *SIGCOMM Comput. Commun. Rev.*, 42(2):42–48, Mar. 2012.

[3] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In N. Koblitz, editor, *Advances in Cryptology CRYPTO 96*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin Heidelberg, 1996.

[20]PRISM: http://fp7-prism.eu

[21]FP7: http://ec.europa.eu/research/fp7/

[4] E. Blanton. TCPurify - a "sanatary" sniffer. http://web.archive.org/web/20110820174616/http://masaka.cs.ohiou.edu/~eblanton/tcpurify/ Accessed on: 2015-07-07.

[5] J. Bongertz. Trace file sanitization NG, 2013. https://sharkfest.wireshark.org/sharkfest.13/presentations/SEC-04_Trace-File-Sanitization-NG_Jasper-Bongertz.pdf Accessed on: 2015-07-07.

[6] T. Brekne and A. Årnes. Circumventing ip-address pseudonymization. In *Proceedings of the Third IASTED International Conference on Communications and Computer Networks, October 24-26, 2005, Marina del Rey, CA, USA*, pages 43–48, 2005.

[7] T. Brekne, A. Årnes, and A. Øslebø. Anonymization of IP traffic monitoring data: Attacks on two prefix-preserving anonymization schemes and some proposed remedies. In *Privacy Enhancing Technologies, 5th International Workshop, PET 2005, Cavtat, Croatia, May 30-June 1, 2005, Revised Selected Papers*, pages 179–196, 2005.

[8] M. Burkhart, D. Schatzmann, B. Trammell, E. Boschi, and B. Plattner. The role of network trace anonymization under attack. *Computer Communication Review*, 40(1):5–11, 2010.

[9] C. Clark, W. Lee, D. Schimmel, D. Contis, M. Koné, and A. Thomas. A hardware platform for network intrusion detection and prevention. In *In Proceedings of the 3rd Workshop on Network Processors and Applications (NP3), February 2004. 178*, 2003.

[10] J. Connolly, M. Davidson, and C. Schmidt. The trusted automated exchange of indicator information (taxii™). Technical report, MITRE Corporation, 2014.

[11] S. E. Coull, C. V. Wright, F. Monrose, M. P. Collins, and M. K. Reiter. Playing devils advocate: Inferring sensitive information from anonymized network traces. In *in Proceedings of the Network and Distributed System Security Symposium*, pages 35–47, 2007.

[12] W. de Bruijn, A. Slowinska, K. van Reeuwijk, T. Hruby, L. Xu, and H. Bos. Safecard: A gigabit ips on the network card. In D. Zamboni and C. Kruegel, editors, *Recent Advances in Intrusion Detection*, volume 4219 of *Lecture Notes in Computer Science*, pages 311–330. Springer Berlin Heidelberg, 2006.

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

[14] T. Farah and L. Trajkovic. Anonym: A tool for anonymization of the internet traffic. In *Cybernetics (CYBCONF), 2013 IEEE International Conference on*, pages 261–266, June 2013.

[15] G. Fisk, M. Fisk, C. Papadopoulos, and J. Neil. Eliminating steganography in internet traffic with active wardens. In *in IH 02: Revised Papers from the 5th International Workshop on Information Hiding*, pages 18–35. Springer, 2002.

[16] M. Foukarakis, D. Antoniades, and M. Polychronakis. Deep packet anonymization. In *Proceedings of the Second European Workshop on System Security*, EUROSEC '09, pages 16–21, New York, NY, USA, 2009. ACM.

[17] T. Gamer, C. P. Mayer, and M. Schller. Pktanon - a generic framework for profile-based traffic anonymization. *Praxis der Informationsverarbeitung und Kommunikation*, 31(2):76–81, 2008.

[18] S. Gattani. Reference models for network trace anonymization. Master's thesis, Iowa State University, 2008. Retrospective Theses and Dissertations. Paper 15304.

[19] P. Haag. Watch your flows with nfsen and nfdump. 50th RIPE Meeting; May 3, 2005 Stockholm.

[20] M. Harvan and J. Schönwälder. Prefix- and lexicographical-order-preserving ip address anonymization. In *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, pages 519–526, April 2006.

[21] A. Y. C. Heng. Bit-Twist: Libpcap-bassed ethernet packet generator. http://bittwist.sourceforge.net/ Accessed on: 2015-07-07.

[22] D. Herrmann, R. Wendolsky, and H. Federrath. Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, CCSW '09, pages 31–42, New York, NY, USA, 2009. ACM.

[23] J. King, K. Lakkaraju, and A. Slagell. A taxonomy and adversarial model for attacks against network log anonymization. In *Proceedings of the 2009 ACM Symposium on Applied Computing*, SAC '09, pages 1286–1293, New York, NY, USA, 2009. ACM.

[24] F. Klassen and AppNeta. Tcpreplay - pcap editing and replaying utilities. http://tcpreplay.appneta.com/ Accessed on: 2015-07-07.

[25] E. Kohler. Click for measurement. Technical report, UCLA Computer Science Department, 2006.

[26] T. Kohno, A. Broido, and K. C. Claffy. Remote physical device fingerprinting. *IEEE Trans. Dependable Secur. Comput.*, 2(2):93–108, Apr. 2005.

[27] D. Koukis, S. Antonatos, D. Antoniades, and E. Markatos. A generic anonymization framework for network traffic. In *Proceedings of IEEE International Conference on Communications, ICC 2006, Istanbul, Turkey, 11-15 June 2006*. IEEE, 2006.

[28] C. Kreibich. Design and implementation of netdude, a framework for packet trace manipulation. 2004.

[29] K. Lakkaraju and A. J. Slagell. Evaluating the utility of anonymized network traces for intrusion detection. In A. Levi, P. L. 0005, and R. Molva, editors, *SecureComm*, page 17. ACM, 2008.

[30] LIAM Group. Flaim core users guide, 2008.

[31] P.-C. Lin and Y.-W. Lin. Towards packet anonymization by automatically inferring sensitive application fields. In *Advanced Communication Technology (ICACT), 2012 14th International Conference on*, pages 87–92, Feb 2012.

[32] Y.-D. Lin, P.-C. Lin, S.-H. Wang, I.-W. Chen, and Y.-C. Lai. Pcaplib: A system of extracting, classifying, and anonymizing real packet traces. *Systems Journal, IEEE*, PP(99):1–12, 2014.

[33] G. Lyon. *Nmap Network Scanning*. Insecure Press, 2009. ISBN-10: 0-9799587-1-7.

[34] M. Majzoobi, F. Koushanfar, and S. Devadas. Fpga-based true random number generation using circuit metastability with adaptive feedback control. In B. Preneel and T. Takagi, editors, *Cryptographic Hardware and Embedded Systems CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 17–32. Springer Berlin Heidelberg, 2011.

[35] C. P. Mayer, T. Gamer, and Dr. M. Schöller. Pktanon manual, 2008.

[36] D. Moore, K. Keys, R. Koga, E. Lagache, and K. C. Claffy. The coralreef software suite as a tool for system and network administrators. In *Proceedings of the 15th USENIX Conference on System Administration*, LISA '01, pages 133–144, Berkeley, CA, USA, 2001. USENIX Association.

[37] R. Pang, M. Allman, V. Paxson, and J. Lee. The devil and packet trace anonymization. *SIGCOMM Comput. Commun. Rev.*, 36(1):29–38, Jan. 2006.

[38] R. Pang and V. Paxson. A high-level programming environment for packet trace anonymization and transformation. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '03, pages 339–351, New York, NY, USA, 2003. ACM.

[39] R. Ramaswamy, N. Weng, and T. Wolf. An ixa-based network measurement node. In *Proc. of Intel IXA University Summit, 2004.*, 2004.

[40] D. Riboni, A. Villani, D. Vitali, C. Bettini, and L. Mancini. Obfuscation of sensitive data for incremental release of network flows. *Networking, IEEE/ACM Transactions on*, 23(2):672–686, April 2015.

[41] C. Schmoll, S. Teofili, E. Delzeri, G. Bianchi, I. Gojmerac, E. Hyytia, B. Trammell, E. Boschi, G. Lioudakis, F. Gogoulos, A. Antonakopoulou, D. Kaklamani, and I. Venieris. *State of the art on data protection algorithms for monitoring systems.* PRISM, IST-2007-215350, Data Protection Algorithms, http://fp7-prism.eu/images/upload/Deliverables/fp7-prism-wp3.1-d3.1.1-final.pdf Accessed on: 2015-07-07.

[42] M. Scott. A wire-speed packet classification and capture module for netfpga. In *First European NetFPGA Developers' Workshop, Cambridge, UK*, 2010.

[43] V. Seeberg and S. Petrovic. A new classification scheme for anonymization of real data used in ids benchmarking. In *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, pages 385–390, April 2007.

[44] A. Slagell, Y. Li, and K. Luo. Sharing network logs for computer forensics: a new tool for the anonymization of netflow records. In *Security and Privacy for Emerging Areas in Communication Networks, 2005. Workshop of the 1st International Conference on*, pages 37–42, Sept 2005.

[45] A. J. Slagell, K. Lakkaraju, and K. Luo. FLAIM: A multi-level anonymization framework for computer and network logs. *CoRR*, abs/cs/0606063, 2006.

[46] S. Ubik, P. Zejdl, and J. Halak. Real-time anonymization in passive network monitoring. In *Networking and Services, 2007. ICNS. Third International Conference on*, pages 100–100, June 2007.

[47] Universita degli Studi di Brescia. tcpanon. http://www.ing.unibs.it/ntw/tools/tcpanon/ Accessed on: 2015-07-07.

[48] R. van Rijswijk-Deij. Ethical NREN data sharing. https://tnc15.terena.org/core/presentation/127 Accessed on: 2015-07-07. SURFnet at Terena Networking Conference 2015.

[49] T. H. Vu, N. Q. Tuan, T. N. Thinh, and N. T. H. Nguyen. Memory-efficient tcp reassembly using fpga. In *Proceedings of the Second Symposium on Information and Communication Technology*, SoICT '11, pages 238–243, New York, NY, USA, 2011. ACM.

[50] J. Xu, J. Fan, M. Ammar, and S. B. Moon. On the design and performance of prefix-preserving ip traffic trace anonymization. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, IMW '01, pages 263–266, New York, NY, USA, 2001. ACM.

[51] T. Ylonen. Thoughts on how to mount an attack on tcpdpriv's "-a50" option... http://ita.ee.lbl.gov/html/contrib/attack50/attack50.html Accessed on: 2015-07-07, 2001.

[52] W. Yurcik, C. Woolam, G. Hellings, L. Khan, and B. Thuraisingham. Scrub-tcpdump: A multi-level packet anonymizer demonstrating privacy/analysis tradeoffs. In *Security and Privacy in Communications Networks and the Workshops, 2007. SecureComm 2007. Third International Conference on*, pages 49–56, Sept 2007.

[53] W. Yurcik, C. Woolam, G. Hellings, L. Khan, and B. M. Thuraisingham. Toward trusted sharing of network packet traces using anonymization: Single-field privacy/analysis tradeoffs. *CoRR*, abs/0710.3979, 2007.

[54] P. Zhang, X. hong Huang, M. qi Luo, C. yu Ning, and Y. Ma. Fast restorable prefix-preserving {IP} address anonymization for ipv4/ipv6. *The Journal of China Universities of Posts and Telecommunications*, 17, Supplement 2(0):93 – 98, 2010.

Appendix A

Name	Tool / Library	Organisation:	Author:	Paper:	Document ID	Development period:	Latest version
AAPI	library	FORTH ICS / ENISA	D. Koukis et al.	"A Generic Anonymization Framework for Network Traffic"	10.1109/ICC.2006.255113	2006	n.a.
anontool	tool	FORTH ICS	M. Foukarakis	"Deep packet anonymization"	10.1145/1519144.1519147	sept 2006 - nov 2006	1.5
Anonym	tool	Simon Fraser University	T. Farah, L. Trajkovi	"Anonym: A tool for anonymization of the Internet traffic"	10.1109/CYBConf.2013.6617434	2013	0.2
Anonymizer	tool	University of Napoli Federico II	C. Mazzariello et al.	"Intrusion Detection Systems"	ISBN: 978-0-387-77265-3	jun 2005 - nov 2006	0.5
Bit-Twist	tool	Multimedia University	A. Yeow Chin Heng			jan 2006 - apr 2012	2.0
Bro anonymizer by Pang	tool + library	Princeton University	R. Pang et al.	"A high-level programming environment for packet trace anonymization and transformation"	10.1145/863955.863994	2003 - 2005	-
CANINE	tool	NCSA	K. Luo et al.	"Sharing network logs for computer forensics: a new tool for the anonymization of netflow records"	10.1109/SECCMW.2005.1588293	jun 2005 - sept 2005	1.2.1
CoralReef	tool	CAIDA	CoralReef dev. Team	The CoralReef Software Suite As a Tool for System and Network Administrators LISA '01 Proc. of the 15th USENIX conference		feb 1999 - okt 2014	3.9.4
Crypto-PAn	library	College of Computing at Georgia Tech.	Fan et al.	"Prefix-preserving IP address anonymization: measurement-based security evaluation and a new cryptography-based scheme"	10.1016/j.comnet.2004.03.033	aug 2002	1.0
FLAIM	tool	NCSA	LIAM Working group	in 20th LISA conference		jul 2006 - feb 2007	0.7.0
FlowScrub	tool		J. Oberheide			2014 - 2015	-
IP::Anonymous	library	Northwestern University	J. Kristoff			nov 2005	0.4
IPSummaryDump	tool	Computer Science Department - UCLA	E. Kohler	"Click for Measurement"		2001 - feb 2015	1.85
Lucent Crypto-Pan	library	Lucent Technologies	D. Stott			2005	-
NetDude	tool	ICSI Berkeley	C. Kreibich	"Design and implementation of netdude, a framework for packet trace manipulation"		dec 2002 - mar 2010	0.5.2
NFDUMP	tool	SWITCH	P. Haag	"Watch your Flows with NfSen and NFDUMP" - 50th RIPE Meeting		feb 2005 - jan 2012	1.3.6p1
PCAPAnon	tool	National Chiao Tung University	Ying-Dar Lin, et al.	"PCAPLib: A System of Extracting, Classifying, and Anonymizing Real Packet Traces"	10.1109/JSYST.2014.2301464	2012 - 2014	-
pktanon	tool	Karlsruhe Institute of Technology	Gamer et al.	"PktAnon – A Generic Framework for Profile-based Traffic Anonymization"	10.1515/piko.2008.0016	early 2008 - sep 2011	1.4.0
SCRUB-tcpdump	tool	colaboration	Yurcik et. al	"SCRUB-tcpdump: A multi-level packet anonymizer demonstrating privacy/analysis tradeoffs "	10.1109/SECCOM.2007.4550306	nov 2007	0.1
tcpanon	tool	Universita' di Brescia	E. Bonazzoli, F. Gringoli, L. Salgarelli			2007 - 2008	0.2
tcpdpriv	tool	Ipsilon Networks	Greg Minshall	Thoughts on How to Mount an Attack on tcpdpriv's ``-A50" Option (http://ita.ee.lbl.gov/html/contrib/attack50/attack50.html)		1996 - oct 2005	1.2
tcpmkpub	tool	colaboration	Pang et al.	"The devil and packet trace anonymization"	10.1145/1111322.1111330	jan 2006	0.1
tcprewrite	tool	AppNeta	A. Turner			may 1999 - dec 2014	4.1.0
TCPurify	tool	Purdue University	E. Blanton			apr 2000 - jan 2008	0.11.2
TraceAnon	tool	WAND - University of Waikato	P. Lorier et al.	"Libtrace: a packet capture and analysis library"	10.1145/2185376.2185382	oct 2005 - feb 2015	3.0.22
TraceWrangler	tool		J. Bongertz	"Trace File Sanitization NG" - Sharkfest '13		2013 - jan 2015	0.3.6

Name	Code state	MAC address	IPv4 address	UDP / TCP port nr.	IPv6	Header checksum correction	L5-7 substitution	IP / TCP options	VLAN tags	Tunneling	Real-time	License	Score
AAPI	code unavailable	unknown	prefix-preserving (Crypto-Pan) prefix-preserving-map (simpler)	unknown	unknown	unknown	yes programatically	unknown	unknown	unknown	unknown	unknown	20,8%
anontool	compiles	black marker	black marker prefix-preserving (crypto-pan) prefix-preserving-map	map zero	no support	yes	yes through policies	yes	aware	no	input only	GPL v2	**79,2%**
Anonym	depends on MATLAB	Truncation Reverse truncation Permutation Structured pseudonymization Black marker	Truncation Reverse truncation Permutation Prefix-preserving pseudonymization Black marker	Binning Permutation Black marker	Truncation Reverse truncation Permutation Prefix-preserving pseudonymization Black marker	no support	no support	no support	no support	no support	no support	No declared licenses	41,7%
Anonymizer	compile errors	unknown	prefix-preserving prefix-preserving-map (simpler)	substitution	unknown	unknown	unknown	unknown	unknown	unknown	unknown	GPL v2	25,0%
Bit-Twist	compiles	1-on-1 substitution	1-on-1 substitution	1-on-1 substitution	no support	yes	append own payload truncate	no support	no support	no support	no support	GPL v2	37,5%
Bro anonymizer by Pang	code unavailable	unchanged	prefix-preserving	yes	no support	yes	yes programatically	partially (fixed transformation)	unknown	unknown	yes	BSD	58,3%
CANINE	code unavailable	Not available in netflow v5 and v7	Prefix preserving (Crypto-Pan)	unknown	no support	unknown	no support	no support	no support	no support	no support		20,8%
CoralReef	compile errors	unchanged	Black marker Prefix preserving (Crypto-Pan)	unchanged	Black marker	yes	only IP-in-IP, rest is truncated	no support	no support	no support	input only	BSD	33,3%
Crypto-PAn	compiles	unchanged	prefix-preserving (crypto-pan)	unchanged	original implementation only has support for IPv4	no support	no support	no support	no support	no support	no support	BSD like	25,0%
FLAIM	compile errors	BinaryBlackMarker BytesTruncation BinaryTruncation Annihilation BinaryRandomPermutation Hash	BinaryPrefixPreserving (Crypto-Pan) BinaryBlackMarker Annihilation BinaryRandomPermutation NumericTruncation Hash	BinaryBlackMarker NumericTruncation Substitution Annihilation BinaryRandomPermutation Classify Hash	not on pcap level	Annihilation Hash	no support	yes	no support	no support	no support	BSD	50,0%
FlowScrub	code unavailable	yes	set black marker mask random permutation hashing, etc	Black marker	unknown	unknown	unknown	unknown	unknown	unknown	unknown	GPL v2	37,5%
IP::Anonymous	compiles	unchanged	prefix-preserving (crypto-pan)	unchanged	no support	no support	no support	no support	no support	no support	no support	GPL / Artistic	25,0%
IPSummaryDump	compiles	unchanged	Uses A50 anonymisation of tcpdpriv	unchanged	no support	no support	no support	no support	yes	no support	no support	BSD like	33,3%
Lucent Crypto-Pan	compiles	unchanged	prefix-preserving (crypto-pan)	unchanged	no support	no support	no support	no support	no support	no support	no support	BSD like	25,0%
NetDude	compile errors	manually	manually	manually	manually	manually	manually	manually	manually	manually	no support	BSD	45,8%
NFDUMP	compiles	unchanged	prefix-preserving (cryptopan)	unchanged	prefix-preserving (cryptopan)	no support	no support	no support	no support	no support	no support	BSD	33,3%
PCAPAnon	compile errors (messy code)	black marker	(length-)prefix-preserving	unchanged	(length-)prefix-preserving	yes	field transformation pattern substitution	yes	yes	yes	no support	GPL	**75,0%**
pktanon	compiles	keep bytewiseHashSha1 bytewiseHashHmacSha1	unchanged hashHmacSha1 bytewiseHashSha1 bytewiseHashHmacSha1 prefix-preserving (crypto-pan)	bytewiseHashSha1 bytewiseHashHmacSha1	unchanged hashHmacSha1 bytewiseHashSha1 bytewiseHashHmacSha1	yes	truncate in length Constant overwrite	Shorten	bytewiseHashSha1 bytewiseHashHmacSha1	yes	Through pipes	GPL v3	**87,5%**
SCRUB-tcpdump	compiles but segfaults!	unchanged	Black marker Pure random permutation Keyed random permutation Prefix-preserving pseudonymization Truncation	Black marker Pure random permutation Keyed random permutation Bilateral classification	no support	yes	Black marker Selective black marker	no support	no support	no support	no support	GPL v2	37,5%
tcpanon	compile errors	unchanged	unchanged	unchanged	unchanged	yes	yes	no support	no support	no support	no support	BSD	25,0%
tcpdpriv	compile errors	unchanged	Sequential numbering per unique IPv4 address Sequential numbering with upper/lower half of IPv4 addr. Sequential numbering for each IPv4 octet (A2) Semi prefix preserving option (A50 - see paper) No changes (A99)	A0: single 16 bit port to integer A1: split 8 bits to integers	no support	yes	truncated	replaced with 0x01 (nops)	no support	no support	input only	BSD	41,7%
tcpmkpub	compile errors	scrambling the entire MAC address scrambling the lower 3 bytes of addr, preserving Vendor-code scrambling the Vendor-code and lower 3 bytes independently	prefix-preserving (crypto-pan) for external addresses cryptographic	black marker	no support	yes	truncate	yes	no support	no support	no support	GPL v2	50,0%
tcprewrite	compiles	fixed endpoints	Deterministic pseudo-random with seed init Pseudo-NAT (lower part of CIDR unchanged) All IPs to 2 fixed endpoints	1-on-1 substitution	yes	yes	no support	no support	add remove no modification	no support	no support	BSD	58,3%
TCPurify	compiles	unchanged	randomised substitution within subnet	unchanged	no support	yes	truncated	no support	no support	no support	input only	GPL v2	33,3%
TraceAnon	compiles	unchanged	prefix substitution prefix preserving (crypto-Pan)	unchanged	no support	no support	no	no support	no support	no support	no support	GPL v2	25,0%
TraceWrangler	precompiled binary only	randomization replacement	random replacement	randomise option to leave < 1024	random replacement	yes	Only IP and MAC in DHCP	no support	yes	support for AYIYA	no support	closed source	58,3%