UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics & Computer Science

Removing private sensitive data within DDoS attacks

T.J. van Geest

B.Sc Thesis

August 2018

Supervisor:
dr. J. J. C. Santanna

Design and Analysis of
Communication Systems Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Summary

Distributed Denial of Service (DDoS) attacks are attempts to make online services and systems unavailable. DDoS attacks growing in size, with a record of reaching a bandwidth 1.7Tbps in 2018. Sharing the data of these attacks is crucial for improving the detection and mitigation of DDoS attacks. However, victims of these attacks are reluctant to share data, due to the private sensitive information that might be leaked. In this thesis we study the sensitive information that is available in generic network traces, and the techniques and tools to anonymize this information. With the background of generic network traces, we consider the sensitive information in DDoS network traffic, to develop a new feature-set for anonymization tools. We apply this feature-set to the existing tools, and test the available tools for their performance. The best performing tool is tcprewrite, however there is no tool that meets all requirements for DDoS network traffic anonymization.

Contents

Chapter 1

Introduction

DDoS (Distributed Denial of Service) attacks are attempts to make online services and systems unavailable. Attackers use multiple traffic sources to overload the resources of their targets. The most DDoS attacks are aimed at the gaming industry, by gamers targeting the sites hoping to gain an advantage over competitors or simply out of frustration. Telecom and financial services follow at a distance as the second and third most used targets (McKeay, 2019). To increase the bandwidth of DDoS attacks, often protocols that provide information (e.g. DNS, NTP) are abused to execute an amplification attack. The attacker lets thousands of bots request certain information at multiple servers while pretending to be the victim. The servers will answer those requests, sending huge amounts of data to the victim. The answer that the server sends can be, dependent on the used protocol, several factors larger than the request.

On February 28th 2018 the largest DDoS attack measured up until that date took place. This attack made GitHub (intermittently) unavailable for a short period of time (Kottler, 2019). The attack peaked at a bandwidth of 1.35 Tbps, breaking the previous record of 1.2 Tbps. To put this in perspective, a HD Netflix stream is about 5 mbps (Netflix), resulting in 1.35 Tbps being about 270.000 simultaneous HD Netflix streams. A few days after this new record, NETSCOUT Arbor confirmed a 1.7 Tbps attack targeted at a U.S. provider (Morales, 2019). Both attacks were using a new type of amplification attack, abusing the memchached protocol. This new attack has an amplification rate of up to 51200 (Majkowski, 2019), leading to huge amounts of network traffic, hence resulting in the new records. While the peak size of attacks is increasing, the frequency of attacks has been hovering around 40 since 2016 (McKeay, 2019). This means that victims of DDoS attacks can (and should) expect to be targeted again in the future.

Using the network traffic captured during DDoS attacks, the protection against DDoS attacks can be improved. The network captures can also be helpful to the tracking of the perpetrator. This does not only help the victim that is sharing data, but other (possible) victims and further research as well. However, sharing this data is not as easy as it may seem. The data can contain a lot of information that is sensitive. This sensitive information can be extracted and abused by every party with access to the DDoS network traffic. For example, take an attack on a bank. The network capture of the bank may contain sensitive information about their clients such as their account numbers/balance, their expenses or even their home address. In case of data loss, these customers will lose their trust in the bank and move their accounts to a more trustworthy competitor. The network capture can also reveal (parts of) the structure of the banks internal network. This could lead to new attacks aimed at the specific weaknesses of the network structure. The risk of exposing sensitive information is often too high for victims to share their DDoS network traffic.

To overcome this problem, there are many tools to anonymize **generic** network traces. Schmoll et al. (2008) gave a state of the art in 2008 and van Dijkhuizen and van der Ham (2018) in 2018. However, van Dijkhuizen and van der Ham (2018) concluded that it is not recommended to share complete anonymized data-sets with parties that are not completely trusted. One of the remaining questions is: how should DDoS attack network traffic be anonymized to protect the victims privacy while still enabling usage of the data for improving the detection and mitigation purposes? For answering our main research question, we defined three sub-research questions.

RQ1: What are the generic practices on network traffic anonymization?

RQ2: How could the generic practices on network traffic anonymization be applied to DDoS attack network traffic?

RQ3: What is the best performing generic practice on DDoS attack network traffic anonymization?

In chapter 2 we answer RQ1. While answering RQ1 we identify the sensitive information in generic network traffic. The techniques that can be used to anonymize this data are discussed, followed by an overview of the tools that are available to be used for generic network traffic anonymization. In chapter 3 we focus on the DDoS attack network traffic. We discuss the sensitive information in the DDoS attack network traffic and compare how the tools discussed in chapter 2 can anonymize this information. In chapter 4 the tools as selected in chapter 3 are tested to compare their performance. In chapter 5 we use RQ1, RQ2 and RQ3 to answer our main research question and to discuss possible future work.

Chapter 2

Generic Practices on Network Traffic Anonymization

In this chapter we discuss the background of generic network traffic anonymization. Several papers have been written about the state and methods of network traffic anonymization tools. In (van Dijkhuizen and van der Ham, 2018), a survey of the literature on network traffic anonymization techniques over the period of 1998–2017 is presented. As we consider this a comprehensive and high quality piece of work, the remaining part of section 2 is a revision of this paper. In section 2.1 the sensitive information in the Data Link layer, Network layer and the Transport layer is described. In addition to this, several protocols of the Application layer are considered. Section 2.2 describes the anonymization techniques that can be used, followed by an overview of existing tools to anonymize network traces in section 2.3. It is important to keep in mind that this chapter is not focused on DDoS network traffic, but on **generic** network traffic. In chapter 3 we narrow this down to DDoS network traffic.

2.1 TCP/IP Layer Sensitivity

Communication between two systems in a network is an exchange of information (packets). When a packet is sent, it moves through different layers in the network. Every layer has its own protocols, which add or subtract their own header and then forward the packet to the next layer. The TCP/IP stack model defines five layers. Although we do not discuss the lowest layer (physical network), we do consider the data link layer, internet layer, transport layer and application layer. A typical network packet according to this model can be seen in Figure 2.1. The outer layer of this packet is the ethernet frame. This frame consists of the ethernet header, followed by the payload and the Frame Check Sequence (FCS). The ethernet payload consists of the IP header followed by its corresponding payload. The IP payload consists of

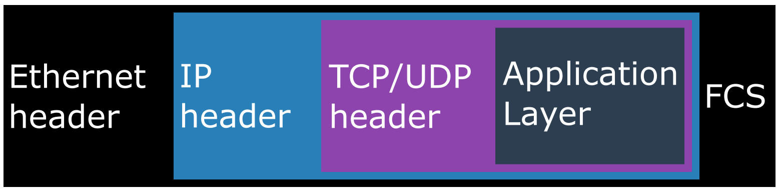

Figure 2.1: Typical network packet.

the TCP or UDP header, and the TCP/UDP payload which contains the application layer protocols. Each of the protocols (Ethernet, IP, TCP, etc.) has its own header. This header contains different pieces of information, which we call fields. In this subsection the sensitivity of the fields of different protocols per layer is discussed. We consider three levels of sensitivity: not sensitive, sensitive and very sensitive. Very sensitive information reveals the owner/identity of a system or network, sensitive information reveals general information about systems or the network, and all other information is considered not sensitive.

2.1.1 Data Link Layer

The data link layer has the purpose to transfer data between network entities. The most prevalent implementation of the data link layer is Ethernet, this is the only implementation considered in the remainder of this section. An ethernet frame contains five important fields, including the payload which simply has to be forwarded to the next layer. The remaining fields are two fields containing MAC addresses, one containing the Virtual LAN identifier (VLAN-ID), and the last field containing the Frame Check Sequence. All fields in the data link layer refer to the network or network structure of the receiving end of the communication. The fields listed are summarized in Table 2.1.

- **Source and Destination MAC Address**
  Each network interface has its own six byte MAC address. The first three bytes often indicate the manufacturer of the device. When a MAC address is known, the behavior of a specific device can be tracked, and the information may be used to attack the anonymization scheme. The vendor code can left intact, but if one vendor had a very small amount of network interfaces in the network trace, this may lead to deanonymization (Pang et al., 2006). MAC addresses therefore are very sensitive.

- **VLAN-ID**
  The VLAN-ID is an optional tag that can refer to a part of the local network. The

	MAC addr.	VLAN tags	FCS
Sensitivity	++	+	-

Table 2.1: Fields in the data link layer and their sensitivity. ++ means very sensitive, + is sensitive and - is not sensitive.

VLAN-ID may reveal structures of the internal network when exposed. Since this information is sensitive, it should be removed if it is present.

- **Frame Check Sequence**
  There is one field that does not contain sensitive data, but should be taken into account, the Frame Check Sequence (FCS). The FCS is the result of a cyclic redundancy check on the frame data, and should be recalculated if the data is changed to guarantee compatibility with tools used on anonymized data.

2.1.2 Internet Layer

The internet layer provides network between independent networks, and has two main protocols, Internet Protocol version 4 (IPv4) and Internet Protocol version 6 (IPv6). Most fields of the IPv6 header are comparable to those of the IPv4 header. Although the size and format of the IPv6 addresses are different, they have the same function as the IPv4 addresses. Current literature on anonymization does not cover the IPv6 fields 'Flow Label' and 'Next Header' yet. We discuss the IPv4 fields, but since this means the version is always 4, we can ignore the first field (which indicates the IP version). The other 11 fields are described below and given in Table 2.2.

- **DSCP & ECN**
  Differentiated Services Code Point (DSCP) and Explicit Congestion Notification (ECN) fields replace the older Type of Service (ToS) field. They are used to indicate the Quality of Service (QoS) and the congestion notification. The older ToS field could lead to type of router identification an reveal user behavior of the source of the packet, this makes the field very sensitive (Yurcik et al., 2007).

- **Total length**
  The total length indicates the packet length in bytes, both the header and the payload are included. Some security events have a fixed packet size, and therefore this field can be important to security analysis. However, this indication of the payload can also be used by attackers, resulting in the field being sensitive for both the receiving and sending parties. The anonymization of the total length is a trade-off between privacy and usefulness (Yurcik et al., 2007).

	DSCP & ECN	**Total length**	**Id.**	**Flags**	**Frag. offset**	**TTL**	**IP addr.**	**Header chksm**	**IHL**	**IP options**
Sensitivity	++	+	+	+	-	+	++	-	-	++

Table 2.2: Fields in the internet layer and their sensitivity. ++ means very sensitive, + is sensitive and - is not sensitive.

- **Identification**
  When an IP datagram is split up in different packets, the identification field indicates which packets belong together, which is necessary for correct re-assembly of the datagram. The generation of the identification number may lead to OS fingerprinting, making this a sensitive field for the device splitting up the packets.

- **Flags**
  The flags field consists of 3 bits. The first bit is reserved, and should be 0. The second bit is the Do not Fragment (DF) bit and the third bit is the More Fragments (MF) bit. If a router is not capable of sending a packet with DF set without fragmenting it, the router will drop the packet. The DF flag could indicate a feature of a device. The MF flag is set for all fragmented packets but the last in one datagram, and does not lead to identification. The MF flag may be useful for security analysis. This makes the flags field sensitive for the source of the packet.

- **Fragment Offset**
  The fragment offset is used to indicate where a packet belongs in a datagram, when this datagram is fragmented. If the total length of a packet is changed, the fragment offset should be recalculated to match this change. This field is not sensitive.

- **Time To Live**
  The Time To Live (TTL) is the maximum amount of hops a packet can make before it is discarded. Every time the packet is forwarded to a next node, the TTL is decreased by one. The sensitive TTL can be used to fingerprint the OS, detect route changes and estimate path hop lengths between source and destination (Yurcik et al., 2007).

- **Source and destination IP**
  The IP addresses are the most important and very sensitive identifiers in the Network layer. They can often be traced to unique machines or users. There are many different techniques used to anonymize IP addresses, resulting in different use for security analysis and level of privacy.

- **Header checksum**
  The checksum of the header is used to check if the header is correct. If the header is changed, the checksum should be recalculated. The checksum is not sensitive.

- **IHL & Options**
  The IP Header Length (IHL) indicates the length of the IP header in multiples of 32 bits. The IHL of an IP header without options is 5 (which means 20 bytes). There are several options that can be added to the IP header, which can contain different kinds of information, such as IP addresses and timestamps. The sensitivity can be very low but also very high, dependent on content. However, the options field is often not used. If there are IP options present, any changes should be reflected in the IHL number as well.

2.1.3 Transport Layer

The protocols of the transport layer provide host-to-host communication services. The most common protocols are the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). Both have their own header. The UDP header only contains 4 fields, the source port, destination port, length and checksum. All except the length can also be found in the TCP header. The TCP data offset and UDP length can be compared to their IPv4 equivalents. Following is a list of TCP header fields, and they are summarized in Table 2.3.

- **Source and destination port**
  Port numbers and IP addresses contribute most to the usefulness of network traces. The port number is used to assume what kind of data is in the payload of a packet. Network tools use port numbers to categorize packets. The port numbers cannot be used directly for identification but can be used to fingerprint services/applications that use specific ports, and therefore are sensitive.

- **Sequence and acknowledgement numbers**
  Sequence and acknowledgement numbers are used in the three-way handshake of TCP. They are used to guarantee the right packet order and to correct for lost and duplicate packets. The sequence number is prone to OS fingerprinting. The acknowledgement number is calculated based on the sequence number, and should be recalculated if the sequence number is modified. The sequence number is sensitive for the initiator of the TCP handshake, while the acknowledgement number is not directly sensitive.

	Port numbers	**Sq./Ack. Numbers**	**Flags**	**Window size**	**Checksum**	**Options**
Sensitivity	+	+	+	+	-	++

Table 2.3: Fields in the transport layer and their sensitivity. ++ means very sensitive, + is sensitive and - is not sensitive.

- **Flags**
  TCP flags are used to indicate a connection state or to provide additional information. DDoS attacks can abuse the TCP protocol, which means that the value of the TCP flags may be important. The flags can be used for OS fingerprinting of the source of the packet (Yurcik et al., 2007) and are therefore sensitive.

- **Window size**
  A sender can send an amount of packets up to the window size, before it waits for acknowledgements. The window size is set by the OS of the source, which means it can be used for OS fingerprinting, which is sensitive.

- **Checksum**
  As for all other checksums, the TCP checksum should also be recalculated if anything is changed within a TCP header or in the payload. The checksum is not sensitive.

- **Options**
  The TCP options field can contain multiple options and is of variable length. This field can be OS specific, and can even contain IP addresses. If time stamps are used in the options field, it can indicate clock skew, which can lead to a specific device (Kohno et al., 2005). The options field can vary between being not sensitive at all and being very sensitive.

2.1.4 Application Layer

There are many different protocols in the application layer. When the application layer is used for a DDoS attack, the packets often seem legitimate to Intrusion Detection Systems (IDS) and servers, and they will be handled like they are normal traffic (Beitollahi and Deconinck, 2012). We discuss 11 of the most used protocols for DDoS attacks as given by McKeay and Fakhreddine (2017).

- **DNS**
  The Domain Name System (DNS) is used to translate domain names to IP addresses. The domain name or IP address in the DNS request are very

sensitive. The extra fields contain information about the DNS queries. This information can be used for OS fingerprinting and tracking user behavior (Dooley and Rooney, 2017), and therefore DNS can be very sensitive.

- **CLDAP**
  Connection-less Lightweight X.500 Directory Access Protocol (CLDAP) is a protocol that is meant to be used to request information from a server. Besides the data that is requested, the CLDAP specifies the type of request and can also contain some user identifier. This can lead to identification and the tracking of user behavior (Young, 1995), making this protocol very sensitive.

- **NTP**
  The Network Time Protocol (NTP) is used to synchronize time between multiple devices. The NTP mainly contains version/type of the NTP packet, next to different times and offsets. These times and frequencies can be used for OS fingerprinting (Mills, 2006).

- **CHARGEN**
  The Character Generator Protocol (CHARGEN) is used to simply send random data without regard to the input. There is no sensitive information in this protocol (Postel, 1983).

- **SSDP**
  The Simple Service Discovery Protocol (SSDP) is used to discover plug & play devices over the network. SSDP messages can contain information of a specific device, such as UUID's and domain names. Dependent of the type of device, this can be very sensitive information (Presser et al., 2008).

- **RPC**
  Remote Procedure Call (RPC) is a protocol that is use to call a procedure to be executed on a specific network location. The RPC protocol contains an authentication that can indicate the type of procedure that is called. The body of RPC messages can indicate this type as well. The sensitivity therefore depends on the program/procedure that is using RPC (Thurlow, 2009).

- **SNMP**
  The Simple Network Management Protocol (SNMP) is a protocol used to manage a network. This means that the protocol is used for collecting and organizing information about the managed devices on the network. SNMP messages can reveal information about the network structure of a system (Case et al., 1990).

- **RIP**
  The Routing Information Protocol is a distance-vector routing protocol, mainly used in small networks due to poor scalability optimization. The routing table that can be send using RIP is of course very sensitive, containing information about network structure and IP or MAC addresses (Malkin, 1998).

- **MDNS**
  Multicast DNS (MDNS) is used instead of DNS in small networks where no DNS server is present. The same programming interfaces, packet formats and operating semantics as DNS are used. This means that MDNS can be very sensitive (Cheshire and Krochmal, 2013).

- **TFTP**
  The Trivial File Transfer Protocol is used to send files over a network. The only sensitive information TFTP can contain is the payload - the data being transferred. This payload is not encrypted and there is no authentication in the protocol, which means that the sensitivity is dependent on the data being transferred (Sollins, 1992).

- **GRE Protocol**
  Generic Routing Encapsulation is a tunneling protocol used to establish a direct connection between network nodes. The GRE header does contain the protocol type of the encapsulated data, which might indicate the type of content that is included. The content itself can be a wide range of protocols, including IP. The content of the GRE packets therefore is very sensitive (Farinacci et al., 2000).

Remarks

The Internet Control Message Protocol (ICMP) is part of the Internet Protocol Suite, but not part of a specific layer (although it is considered to be at the same level as TCP and UDP). ICMP is used to send error messages and operational information, for example feedback if a requested service is not available. ICMP requests and replies can be used for OS fingerprinting (Kohno et al., 2005), and IP addresses or host names can be present in the payload. Therefore ICMP messages are very sensitive.

2.2 Anonymization Techniques

In section 2.1 we discussed what sensitive information can be extracted out of network packets. To hide this information we need to anonymize the sensitive information. Anonymization is the removal of identifying details in a data set. For example, removing or replacing an IP address that could lead to the exact location of a device. There are three important goals of anonymization regarding network traces (Pang et al., 2006). The user of a trace file should not be able to: (1) identify specific hosts (devices owned by users) such that user behavior can be determined, (2) identify internal hosts (devices owned by the victim) such that services and internal hosts can be mapped, (3) learn details of a organizations network structure that would otherwise not be known. These goals ensure that there is no sensitive information leaked regarding external hosts, the victim itself or its systems.

Modifying network traces to remove private sensitive data can generally be achieved using anonymization or pseudonymization (Brekne and Årnes, 2005). Anonymization does not state in what way the identifying details are removed from the dataset, which means a lot of techniques can be used. Pseudonymization replaces the data of true identities with data of pseudonyms, alternate identities, to remove private sensitive information. Therefore, it can be seen as a subtype of anonymization. Pseudonymization has the advantage that data from different sources can be combined, but when the pseudonymization scheme is known, the process is reversible.

The effectiveness of anonymization is not only dependent on the anonymization-technique, but also on the input entropy of the data that is anonymized. If the entropy is too low, which means that there is a lack of variation of the network traces, the chance the data is deanonymized is high. The nine most important techniques to anonymize data (van Dijkhuizen and van der Ham, 2018) are the following:

- **Prefix preservation**
  The leading bytes of fields in network traffic often relate to a special case, such as multicast addresses for the IP fields, or the bytes referring to the manufacturer in a MAC address. These bytes might be relevant for use in research, and in the case they cannot lead to private sensitive information they should not be changed. Retaining the leading bytes is referred to as prefix-preservation.

- **Reordering**
  Reordering data rearranges the bytes of pieces of data. This shuffling is done randomly.

- **Data removal (filtering)**
  The most secure way to avoid sensitive data is to completely remove the data. This is often used when there is no alternative to completely anonymize the

data. Removing the data completely is called shortening or truncation. Another possible option to remove data is black marking, which is overwriting a value with fixed values, for example making every bit 0. Using this technique the information is completely removed, decreasing the use of data for research purposes.

- **Random substitution**
  Like data removal, randomizing data removes possible links between observations. The data is not shuffled, but randomly replaced by data from another source (e.g. random time shifting (Yurcik et al., 2007)).

- **Replacement**
  A simple way to anonymize data is to replace data one-to-one with a value of the same type. The effectiveness of the anonymization as well as the amount of information that is retained is completely dependent on the data that is used to replace the original data.

- **Generalization**
  Generalization replaces data with more general data. For example, values can be rounded, or divided in larger groups. The information the data contains is becoming less specific when using generalization, which means that the amount of information decreases.

- **Enumeration**
  Enumeration can be applied to ordered sets. The first value will be a random value, and each following field will get a greater value, such that the order of the original set is retained. The amount of information decreases, but since the order is the same, the information may still be useful. Enumeration is especially useful for timestamps, since they lose precision but retain their order.

- **Format-Preserving Encryption**
  Encryption is the process of encoding data using a secret key in its encryption algorithm. Encryption ensures discovering the original data is difficult for parties that do not know the key. The encrypted data often differs in size compared to the original data, to hide length and patterns from the attacker. Format-preserving encryption is meant to produce encrypted data that has the same format as the original data (Dworkin, 2016). The original format is often needed to be able to process the anonymized data using network tools (e.g. IP or MAC addresses), which is why format-preserving encryption is desirable.

- **Cryptographic permutation**
  Permutation functions can be used to map a set onto itself. Without knowl-

edge of how the permutation is applied, the process is not reversible. A cryptographic block cipher is a type of permutation that uses a key. This key, combined with the setting of the block cipher, can be used to undo or repeat the anonymization. Hashing is another method of permutation that can be used. However, hashing cannot guarantee unique values, which can influence the effective amount of information.

To make sure network tools can still replay anonymized traffic, it is important to correct TCP sequence/acknowledgement numbers, IP packet length fields and checksums after applying anonymization.

If there is any information left after anonymization, this information is available to both attackers and researches. This means there is always a compromise to be made. The victim has to choose to give some information away to keep the information more useful for research, such as format-preserving encryption, or to remove all information (e.g. black marking), leading to no loss of information but a decrease in usefulness. The tools that can be used to anonymize network traffic are described in section 2.3.

2.3 Anonymization Tools

In section 2.1 the sensitive information in generic network traffic is given, and in section 2.2 the techniques to anonymize network traffic is discussed. In this section we consider the tools that apply the anonymization techniques to the sensitive information in generic network traffic. There are several tools to anonymize network traces. Table 2.4 gives an overview of the most influential tools and the features they support, as given by van Dijkhuizen and van der Ham (2018). To our best knowledge, there should be no additions to this list since it was published. There is one tool worth mentioning, which is SafePcap (Omnipacket, 2019). SafePcap promises to support many protocols, and easy implementation of new protocols. However, only an online demo version of SafePcap is available, and we do not include this tool.

The features given in Table 2.4 contain several fields that are discussed in section 2.1. Besides these fields, the other columns cover the code state, tunneling, real time and license. The code state implies that the binary is executable or the source code compiles without too much problems. The support for tunneling or encapsulation is indicated by the column ’tunneling’, and ’real time’ shows if the tool supports real-time traffic, i.e. on the go anonymization. Whether the tool or framework can be reused depends on the license of the software. The coverage is calculated by simply assigning equal weights to each feature. In addition to this, a ’+’ has a weight of 1, a ’o’ has a weight of 0.5 and a ’-’ has a weight of 0. The tools, features and fields in the table will be reviewed and adjusted to match our specific requirements for DDoS network traffic in chapter 3.

Almost half of the tools have a positive code state, which means it should be possible to test them. The other tools are (partially) unavailable or not supported anymore. There is not one tool that supports all of the features, however there are a few promising tools. Pktanon is a highly configurable tool that supports most of the features, and can make use of different anonymization profiles. However, the application layer is not supported at all. Anontool does support anonymization of the application layer, but lacks IPv6 support and tunneling.

2.4 Concluding Remarks

The goal of this chapter is answering the research question: what are the generic practices on network traffic anonymization? In section 2.1 we look at which information should be anonymized. Network packets consist of different layers and protocols with their own headers. Each of these headers contains fields with information, and the sensitivity of this information can vary per field. The sensitivity of the fields in the data link layer can be found in Table 2.1, the internet layer in Table 2.2 and the transport layer in Table 2.3. The application layer contains lots of different protocols and discussing those protocols in detail is not in the scope of this thesis.

In section 2.2 we discuss the definition and techniques of anonymization. Anonymizing data means removing identifying details in the data set. To anonymize data different techniques can be used. Anonymization is a compromise between (possibly) leaking information and keeping the data useful for both research and attackers. To make sure the least amount of information leaks, one should use a technique such as black marking. To answer the research question we compare the tools that can be used to anonymize network traffic in section 2.3 using Table 2.4. The tools that are currently available all support different features, however there is not one tool that covers every important feature for network traffic anonymization. The tools that look the most promising are pktanon and anontool.

This chapter is focused on the anonymization of **generic** network traffic. In chapter 3 the tools and features they should support regarding DDoS traffic are selected to be able to answer the second research question.

Name	Code state	MAC addr.	IPv4 addr.	UDP TCP port	IPv6	Header chksum corr.	App. layer	IP TCP opts	VLAN tags	Tunnel -ing	Real time	License	feat. coverage
pktanon	+	+	+	+	+	+	-	o	+	+	+	+	87.50%
anontool	+	+	+	+	-	+	+	+	+	-	o	+	79.00%
PCAPAnon	-	+	+	-	+	+	+	+	+	+	-	+	75.00%
TraceWrangler	+	+	+	+	+	+	-	-	+	o	-	+	70.80%
tcprewrite	+	o	+	o	+	+	-	-	+	-	-	+	58.50%
Bro_anonymiser	?	-	+	+	-	+	+	o	?	?	+	+	54.00%
FLAIM	-	+	+	+	-	+	-	+	-	-	-	+	50.00%
tcpmkpub	-	+	+	+	-	+	-	+	-	-	-	+	50.00%
NetDude	-	o	o	o	o	o	o	o	o	o	-	+	46.00%
tcpdpriv	-	-	+	+	-	+	-	+	-	-	-	+	41.50%
Anonym	o	+	+	+	+	-	-	-	-	-	-	?	37.50%
Bit-Twist	+	o	o	o	-	+	-	-	-	-	-	+	37.50%
SCRUB-tcpdump	-	-	+	+	-	+	o	-	-	-	-	+	37.50%
CoralReef	-	-	+	-	+	+	-	-	-	-	-	+	33.50%
IPSummaryDump	+	-	+	-	-	-	-	-	+	-	-	+	33.50%
NFDump	+	-	+	-	+	-	-	-	-	-	-	+	33.50%
TCPurify	+	-	+	-	-	+	-	-	-	-	-	+	33.50%
AAPI	?	?	+	?	?	?	+	?	?	?	+	?	25.00%
Crypto-PAn	+	-	+	-	-	-	-	-	-	-	-	+	25.00%
IP::Anonymous	+	-	+	-	-	-	-	-	-	-	-	+	25.00%
Lucent_Crypto-Pan	+	-	+	-	-	-	-	-	-	-	-	+	25.00%
tcpanon	-	-	-	-	-	+	+	-	-	-	-	+	25.00%
Tcpdump_Anonymizer	-	?	+	+	?	?	?	?	?	?	?	+	25.00%
TraceAnon	+	-	+	-	-	-	-	-	-	-	-	+	25.00%
CANINE	?	-	+	?	-	?	-	-	-	-	-	+	16.50%

Table 2.4: Selection of tools and their feature support as given by van Dijkhuizen and van der Ham (2018). A '+'indicates good support, a '-' indicates no support, a 'o' indicates partial support, and '?' indicates exceptions.

Chapter 3

Anonymizing DDoS Attack Network Traffic

In chapter 2 we discussed the sensitive information in generic network traffic, the techniques to anonymize information and the tools that apply anonymization techniques to network traffic. While there are a few tools that cover a lot of the important features as selected by (van Dijkhuizen and van der Ham, 2018) in Table 2.4, this is all based on generic network traffic. In this chapter we look at the the part of network traffic that relates to DDoS attacks.

In section 3.1 we define what we consider to be DDoS network traffic. With all other traffic being removed, many of the fields discussed in section 2.1 are not relevant anymore. In section 3.2 we select the fields and features that tools should be able to anonymize and support to be relevant for DDoS attack data anonymization. The tools that are successfully tested are compared in section 3.3. Table 3.1 summarizes the findings of this chapter applied to Table 2.4 to show the tools and their features, aimed at the anonymization of DDoS network traffic.

3.1 Extracting DDoS Network Traffic

DDoS attacks overload the resources of the victim, by sending large amounts of network traffic to the victim. Network captures created during a DDoS attack contain all or part of the packets that are sent during the DDoS attack. This captured data can be considered as generic network traffic, since it does not only contain the DDoS attack. The traffic that we are interested in is a subgroup of generic network traffic: DDoS network traffic.

The differences between DDoS attack data and generic network traffic data can be used to develop a new, more specialized, feature-set that tools should support. By filtering data not relevant for our goal, the amount of fields that should be

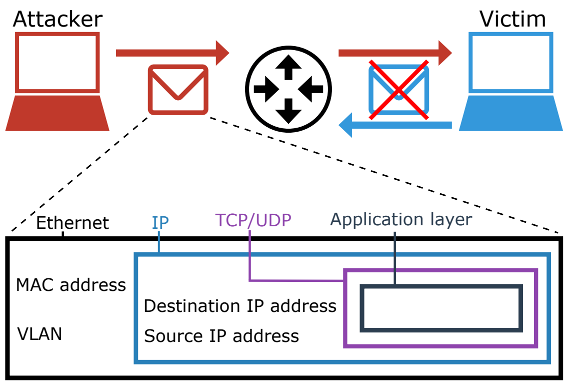

Figure 3.1: Packets in DDoS attacks.

anonymized can be reduced, leading to less requirements for the tools. The most important part of the data that should be removed is the *normal traffic*. By *normal traffic* we mean all traffic that is not part of the DDoS attack. This is user traffic and internal traffic that got captured in between the DDoS attack data. By removing this data a lot of sensitive information is removed, while not influencing the usefulness of the data. Due to the nature of DDoS attacks, all malicious traffic is one way traffic. All outgoing data (data sent by the victim) is not important, and therefore it should be left out of the data capture. Note that methods to filter this data are not in the scope of this thesis.

3.2 New Feature-set

In section 3.1 we discussed what DDoS network traffic is, and that we discard all other traffic. Figure 3.1 shows a packet in a DDoS attack. For all the remaining packets the sensitive data should be anonymized. In this section we will go through all layers in the TCP/IP model (as given in section 2.1) to find which fields should still be anonymized.

- **Data Link layer**
  Both the source and destination MAC addresses present in network captures belong to devices owned by the victim. The VLAN tags refer to the network of the victim, hence all three fields should be anonymized.

- **Internet layer**
  The destination IP address is one the most important fields and must be anonymized. The source IP address however could be useful in research and does not reveal anything about the victim. Therefore the IP address anonymization is split up in two separate columns in Table 3.1. The first column

is simply the support for anonymization of destination IP addresses. The second column is the possibility to anonymize the destination IP address without anonymizing the source IP address. The time to live and IP options could also reveal sensitive information, so for optimal security both fields are anonymized.

- **Transport layer**
  The UDP and TCP headers do not contain a lot of information, however the destination port and TCP options should be removed.
- **Application layer**
  The application layer contains many different protocols, and there is little support for all these protocols in the tools. The application layer in DDoS network traffic is created by the attacker. This means that any sensitive information that is in the application layer, is already known by the attacker. The application layer often defines the DDoS attack and therefore is important for research. To ensure no sensitive data is leaked, the victim should check the application layer for any sensitive information. Since there is no support for this in the tools yet, the column 'App. layer' indicates if there is support to simply remove the application layer (e.g. by truncating or black marking).

The header checksum correction is still relevant, since it may be necessary to correct checksums to be able to replay the anonymized .pcap files. The compatibility indicates if the tool can be used as a script, and especially for automated services this is important. The coverage indicates how much of the features are covered by a tool. It is important to note that there are no weights per feature, since this is really use-case dependent. In section 3.3 we discuss the tools that we test in chapter 4, and in Table 3.1 we compare the features of these tools with the new feature-set as determined in this section.

3.3 Selection of Tools

The tools that are reviewed in this thesis can be compiled/installed on the latest updates of Ubuntu and Windows as of June 2018. No code of the tools is edited, the tools take .pcap files as input and write to .pcap files as output, their dependencies are still available and easy to install, and there is enough documentation to operate the tools. Only six of the twelve tools that should still be available according to (van Dijkhuizen and van der Ham, 2018) meet these requirements.

- **tcprewrite**
  tcprewrite is part of the Tcpreplay suite. Tcprewrite does support more advanced functions than this thesis describes and tests, but these functions re-

Name	MAC	VLAN tags	IPv4	IPv4 dst.	IPv6	IPv6 dst.	TTL	IP options	TCP/UDP dst.	TCP options	App. layer	Chksm corr.	Compat-ibility	Coverage
tcprewrite	o	+	o	-	o	-	+	-	o	-	o	+	+	50%
TraceWrangler	+	+	+	-	+	-	-	-	+	-	-	+	-	46%
Bit-Twist	o	-	o	o	-	-	+	-	o	-	+	+	+	46%
TCPurify	-	-	+	-	-	-	-	-	-	-	o	-	+	19%
IPSummaryDump	-	-	o	-	-	-	-	-	-	-	-	-	+	12%
TraceAnon	-	-	o	-	-	-	-	-	-	-	-	+	-	12%

Table 3.1: Updated list of tools and their features. A ’+’indicates good support, a ’-’ indicates no support, and a ’o’ indicates partial support.

quire a custom cache file, something that is for specific network traces and not for a generic approach. Tcprewrite can change all source MAC addresses into one static address, and the same applies for destination MAC addresses. The VLAN tag can be removed or changed. The IP addresses can be anonymized in a deterministic manner, based on the seed value that the user provides.

- **TraceWrangler**
  Tracewrangler is the only Windows tool that is evaluated. It takes .pcap files as input, and gives .pcapng as output. The tool is very extensive, but is not scriptable. Therefore the tool is only useful for non-automated anonymization. The support for MAC and IP address anonymization is very complete, there is support for black marking, randomization, and replacement using a list. Prefix preservation is possible for both addresses, and both IPv4 and IPv6 are supported. The only but significant shortcoming here is the absence of a possibility to distinguish between source and destination IP. The application layer can be anonymized using black marking or randomization. VLAN tags can be deleted and there are some options for tunneling protocols. Tracewrangler can also truncate packets after a specific amount of bytes or (recognized) protocol.

- **Bit-Twist**
  Bit-Twist operates per layer in network packets. For every layer that is adjusted a new command has to be run. Bit-Twist can replace all source MAC addresses, or replace specific MAC addresses, with a static value. The same applies for destination MAC addresses, and source and destination IP addresses. Bit-Twist can remove all layers after the transport layer, and can append an arbitrary payload at the end of the packet.

- **TCPurify**
  TCPurify is a simple tool that has a limited feature set. There are three options for IP addresses: leaving them as they are, black marking or anonymizing using a randomization function that randomizes specific subnets. The last function is not considered since this is file specific. TCPurify removes everything after the IP header, however there is an option to disable this.

- **ipsumdump**
  Ipsumdump is a program meant to read packets and summarize them into a line-based ASCII file. However, there is an option to anonymize IP addresses, and give a .pcap file as output. The anonymization used preserves prefix and class.

- **traceanon**
  Traceanon is part of the libtrace library. Traceanon can encrypt the source

and/or destination IP addresses, and leave some specific prefixes. Traceanon can also anonymize IP addresses using Rijndael/AES encryption to encrypt each bit of a 32-bit IPv4 address separately (van Dijkhuizen and van der Ham, 2018).

3.4 Concluding Remarks

The goal of this chapter is answering the second research question: How could the generic practices on network traffic anonymization be applied to DDoS attack network traffic? We discussed the difference between captured network traffic and the relevant DDoS network traffic in section 3.1. All normal and outgoing traffic can be discarded since it not part of a DDoS attack. In section 3.2 the fields and features are discussed that tools should have to anonymize DDoS network traffic. Several fields are not sensitive with regard to the destination/victim, and do not have to be anonymized. The tools that are successfully tested and the new feature-set that they should support are given in Table 3.1. In section 3.3 we discuss the tools and their features. There is still not one tool that supports all features. TraceWrangler, tcprewrite and Bit-Twist support the most features, and seem the current best ways to anonymize DDoS network traffic. In chapter 4 we discuss the performance of the tools in Table 3.1.

Chapter 4

Performance of Network Traffic Anonymization Tools

In chapter 2 we discussed the tools for generic network traffic anonymization and the features they should support. In chapter 3 we narrowed this down to six tools that are currently available and selected the features these tools should support. The preliminary conclusion is that TraceWrangler should be the easiest tool to use, where as tcprewrite and Bit-Twist should offer roughly the same amount of features although requiring more effort to use.

In this chapter the tools are tested to compare their performance. The setup used for testing the tools is given in section 4.1. Since every tool has a different feature-set, we start with testing a feature they all support in section 4.2. Since this is the main (or only) feature of some tools, only three tools are tested in the additional tests in this chapter. These tools support both IP and MAC address anonymization. In subsection 4.3.1 and subsection 4.3.2 we compare performance using files that consist of different protocols (DNS, NTP, SSDP and UDP), where all files have the same size or the same amount of packets. The memory usage of the three tools is compared in subsection 4.3.3.

4.1 Setup

All tools are installed on the same hardware. Ubuntu is running in VirtualBox on Windows, with an external SSD (USB 3.1 Gen2) as hard disk. The system has an Intel core i7 7700HQ, 16GB RAM and a NVME SSD. Four cores and 8192MB RAM are assigned to the VirtualBox when it is running. Testing the tools on Windows is done without hyperthreading, to match the four cores of the VirtualBox. The pcap files are obtained at ddosdb.org, which means that they are already anonymized. These anonymized files can still be used to test the tools. Files with different sizes

Name	**DNS0**	**NTP1**	**SSDP1**	**UDP1**
Size (bytes)	2,000,000,364	400,000,055	400,000,169	400,000,416
Amount of packets	1732080	816281	1098808	781926

Name	**DNS12**	**NTP2**	**SSDP2**	**UDP2**
Size (bytes)	400,000,175	171,152,323	126,125,702	177,253,837
Amount of packets	346451	346450	346450	346450

Table 4.1: .pcap files used for testing.

are created by replaying the obtained .pcap files until the required amount of bytes or packets are recorded. To avoid misunderstandings about the file sizes, they are measured in bytes. To compare all tools, a .pcap file of two billion bytes is created.

For further testing only the three tools with IP and MAC address anonymization are used. These tools are tested with files using four different protocols as characteristic of the attack (DNS, NTP, SSDP and UDP). For all the protocols mentioned before there is a file of roughly 0.4 billion bytes, and exactly 346450 or 346451 packets. Since the protocols have different packet sizes, the same amount of packets often refers to a different file size. The different files with their amount of packets and file size can be seen in table 4.1. Note that the file DNS12 has both the file size of 0.4 billion bytes and 346451 packets.

The measured metrics are CPU usage, memory usage and the time it takes for a tool to complete a task. The CPU usage is the total CPU usage. While there are no programs running in the background, note that the CPU usage includes the CPU usage of the OS and other background processes. The memory usage is measured by taking the difference between the total memory used before the tool is started, and the maximum memory used while the tool is running. The time taken by TraceWrangler to complete a task is measured by hand since TraceWrangler is not scriptable. The time taken by the other tools is measured by writing timestamps to a log file before and after running a tool. The CPU and memory usage is measured on Windows using Open Hardware Monitor (Möller, 2016), and on Linux using nmon for Linux (Griffiths, 2018).

4.2 IP address anonymization

The first test run is used to compare all the evaluated tools. The only option active in this test is IP anonymization. If the tools support black marking, this option is used. To be able to visualize the performance the largest file (DNS0) is used. The results can be seen in Figure 4.1, Figure 4.2 and Figure 4.3.

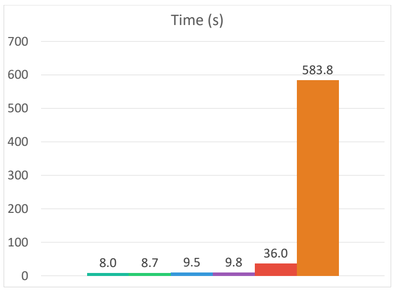

Figure 4.1: Time

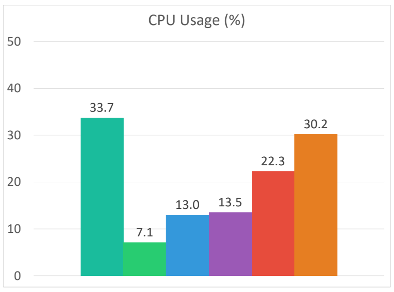

Figure 4.2: CPU

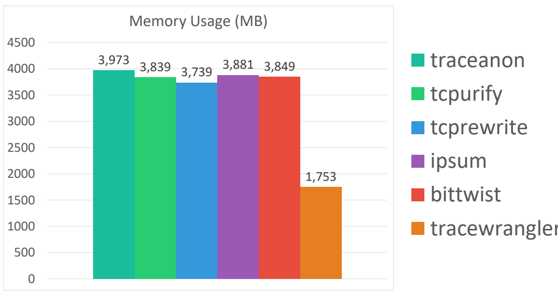

Figure 4.3: Memory

As can be seen in Figure 4.1, there is one tool that takes a considerable amount of time to anonymize the traces, TraceWrangler. Every option is turned off, except recalculation of the checksum. Since TraceWrangler is slow compared to the other tools in all tests, this could be due to an inefficient way of editing the packets. Bit-Twist is a lot faster, but still almost four times as slow as the other Linux tools. The CPU usage of traceanon and tracewrangler are relatively high (Figure 4.2). In Figure 4.3 can be seen that all tools use roughly 3800MB, while TraceWrangler only uses 1753MB. The higher CPU usage of TraceWrangler compared to all other tools except traceanon could be compensation for the smaller amount of memory used. Since traceanon, TCPurify, tcprewrite and ipsumdump use roughly the same amount of time and memory, the differences in CPU usage could indicate the optimization of the anonymization process, with TCPurify being the most optimized and traceanon being the least optimized. It should be noted that bittwist is using one core at 100% almost the whole time, meaning that the tool is relying heavily on one core. Support for multithreading could improve performance significantly.

4.3 IP and MAC address anonymization

The following tests are more specific and aimed at the tools capable of anonymizing MAC and IP addresses. Tracewrangler randomizes both IP and MAC addresses (instead of black marking, to keep the difference between source and destination). Tcprewrite anonymizes the IP addresses and uses black marking for the MAC addresses. However, since the value that is used to replace the MAC addresses can be set as input, the last bit of the source MAC address is set to 1 to still be able to distinguish between source and destination. Bittwist can only set fixed values for the IP and MAC addresses. The same is applied as with the MAC addresses in tcprewrite: the last bit of each source address is set to 1. This ensures the difference between source and destination is not removed. We compare the performance of the tools using files with the same file size but different protocols in subsection 4.3.1, and files with the same amount of packets in subsection 4.3.2.

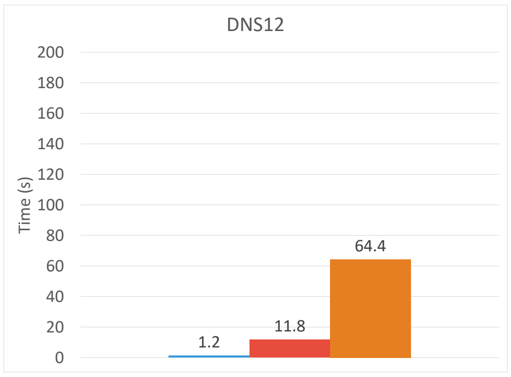

Figure 4.4: DNS

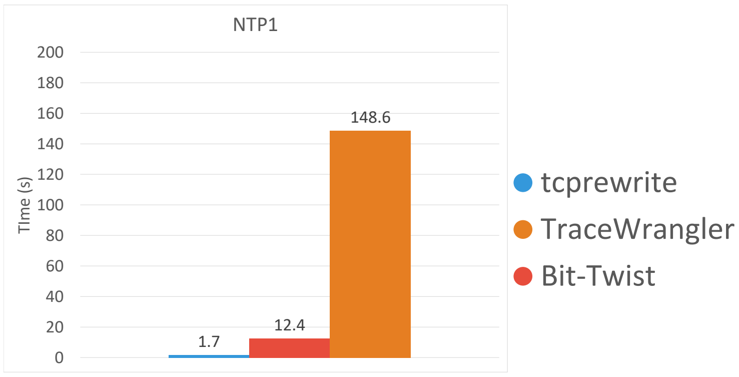

Figure 4.5: NTP

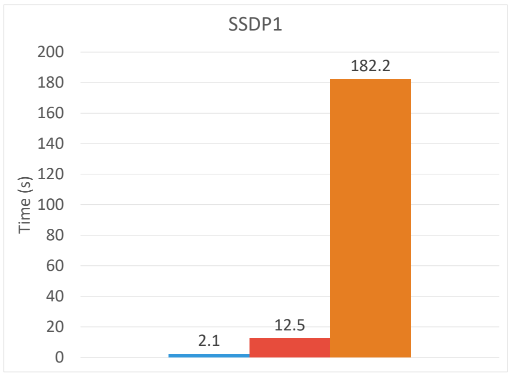

Figure 4.6: SSDP

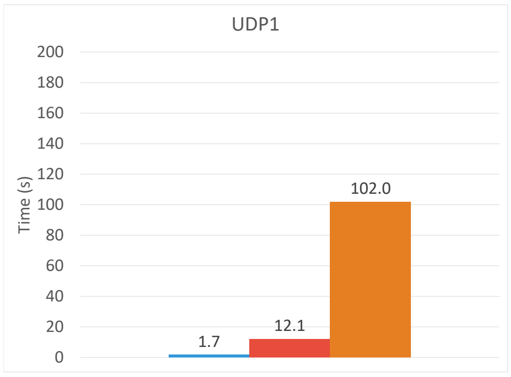

Figure 4.7: UDP

4.3.1 Protocols

The results of the tools anonymizing files with the same size but different protocols can be found in Figure 4.4, Figure 4.5, Figure 4.6 and Figure 4.7. The CPU usage has an absolute variation of 3%, and the memory usage is within 5% of the average. This indicates that the protocol defining the attack does not matter for CPU and memory usage. The same applies for the time for tcprewrite and Bit-Twist. However, TraceWrangler shows huge differences in the time it takes to process the files. The amount of time TraceWrangler takes to process the files is in order of the amount of packets the file contains, where the least amount of packets is processed quickest.

4.3.2 Packets

The results of the tools anonymizing IP and MAC addresses of files with different sizes but the same amount of packets can be found in Figure 4.8, Figure 4.9, Figure 4.10 and Figure 4.11. It can be seen that the time TraceWrangler takes is more dependent on the amount of packets than the file size. Tcprewrite takes a very small amount of time, and although it looks like it scales based on the file size, this can

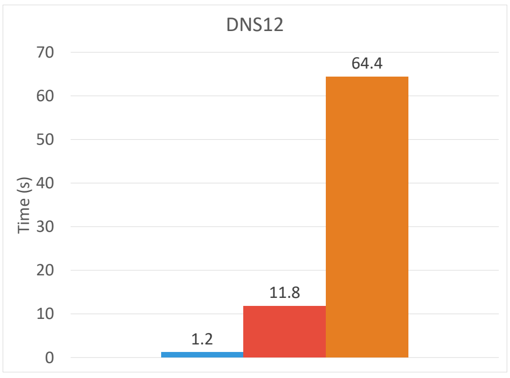

Figure 4.8: DNS

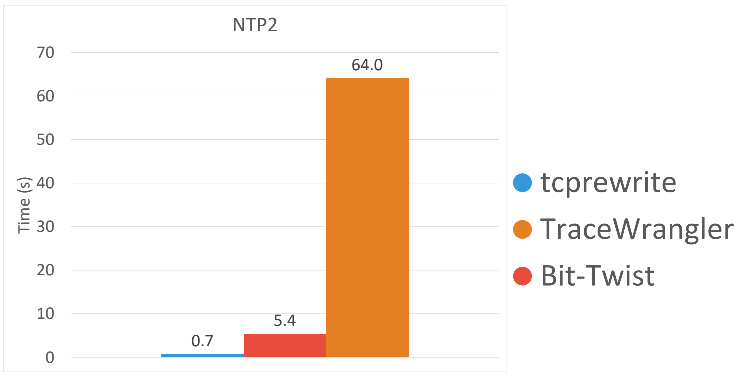

Figure 4.9: NTP

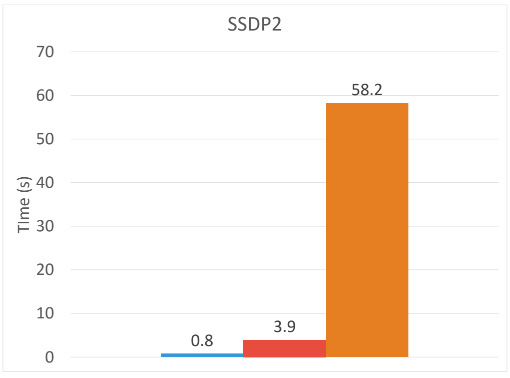

Figure 4.10: SSDP

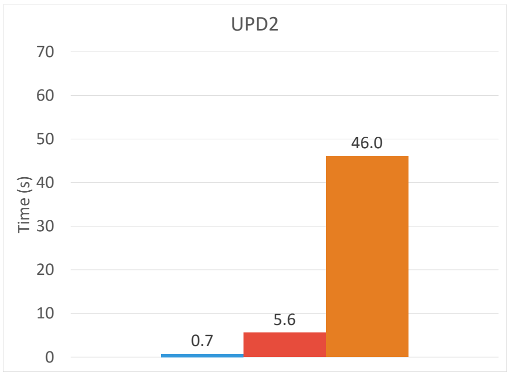

Figure 4.11: UDP

not be concluded. The time Bit-Twist takes however does scale based on the file size. The CPU usage again is within margin of error for all files. The memory usage scaling is discussed in subsection 4.3.3.

4.3.3 Memory usage

Although the memory usage per file per tool can differ a lot, in all tests there is consistency in the amount of memory used. The amount of memory used per tool is dependent on the size of the input file. This can be seen in Figure 4.12. If we give all tools the same amount of memory, TraceWrangler can process the largest files, followed by tcprewrite. TraceWrangler has some more variation than the other tools, but this is within 12% of the average value. Bit-Twist has one value that is not in line with the other values, with DNS0. All Linux tools used the same amount of memory while processing DNS0, and since Bit-Twist uses more memory than other tools in all other tests, a memory limit could be the cause of this inconsistency. Bit-Twist processes the least bytes with the same amount of available memory as the other tools, while TraceWrangler processes more than double the amount of bytes with the same amount of available memory.

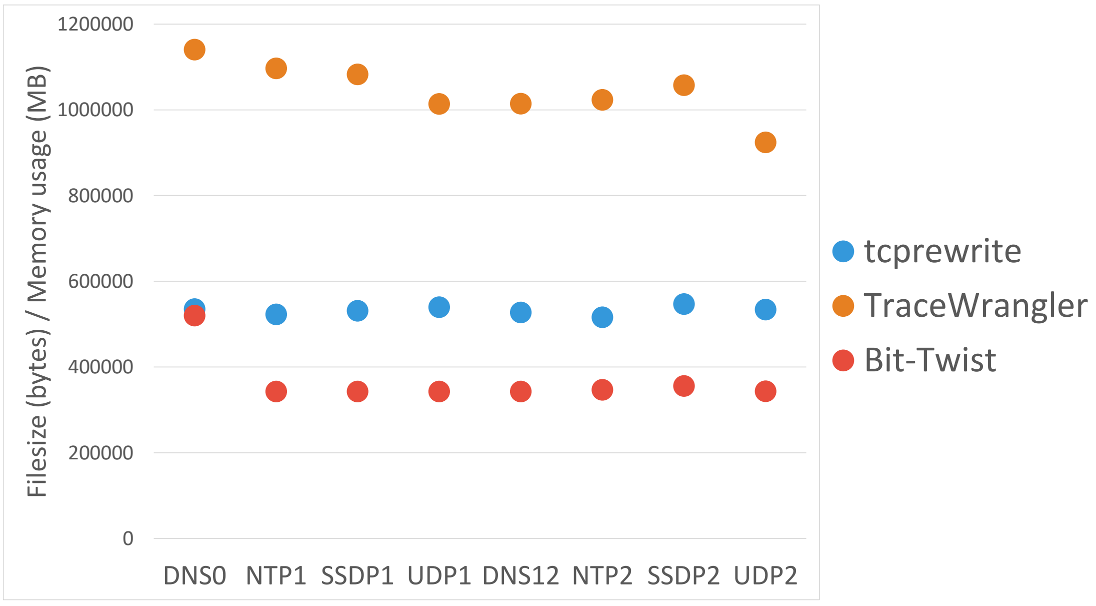

Figure 4.12: File size vs. memory usage. All tools use the same amount of RAM per byte file size for every file.

4.3.4 Remarks

Due to the different feature sets and techniques to anonymize specific protocols and fields, developing tests that each tool can execute is very difficult (e.g. all tools can anonymize IP addresses, but just one tool can use black marking for only the destination IP address). TraceWrangler crashes and exits while processing specific files. The cause of these crashes could not be found. Size and protocols did not seem to have any influence on this matter. While processing some files, TraceWrangler reported an error and stopped anonymizing, without closing the program. As before, a clear reason could not be found. A restart of the computer seemed to fix this second issue. Besides these issues, the maximum file size that can be taken as input for TraceWrangler is 2GB. Some of the tools processed files within or close to one second. Since the CPU and memory usage are measured every second, this could have led to inaccurate results.

4.4 Concluding remarks

In this chapter we answer the third research question: what is the best performing generic practice on DDoS network traffic anonymization? There were six tools to be tested. First, in section 4.2, all tools were tested on the same file using only IP address anonymization. All Linux tools used roughly the same amount of memory, while TraceWrangler used about half of that. TCPurify performs best in terms of time and CPU usage, where as TraceWrangler has the second highest CPU usage and is sixteen times slower than the second slowest tool. In subsection 4.3.1 tcprewrite, TraceWrangler and Bit-Twist were used to anonymize files with the same file size but different protocols. Bit-Twist and tcprewrite used the same amount of time for each of the files, however the time TraceWrangler takes is different per the protocol. In subsection 4.3.2 the cause for this is found, TraceWrangler performs anonymization on files with the same amount of packets in about the same amount of time, despite the differences in file size. Bit-Twist cleary scales based on the file size, and tcprewrite seems to do the same. In subsection 4.3.3 we noticed that TraceWrangler uses the least amount of memory when all tools anonymize the same file. TraceWrangler performs the worst in almost every test regarding time taken, and also has a high CPU usage. tcprewrite takes the least time in every single test, has the lowest CPU usage and performs good in the memory usage as well, and is the overall best tool of all tools tested.

Chapter 5

Conclusion and Future Work

The goal of this thesis is answering the research question: how should DDoS attack network traffic be anonymized to protect the victims privacy while still enabling usage of the data for improving the detection and mitigation purposes? To answer the research question we defined three sub-research questions.

In chapter 2 we answer RQ1: what are the generic practices on network traffic anonymization? We identify the sensitive information in the data link layer, internet layer and transport layer. The sensitive information is found in the headers of protocols, and given in Table 2.1 (data link layer), Table 2.2 (internet layer) and Table 2.3 (transport layer). The most important protocols in the application layer are discussed as well. Anonymization of data is reviewed in section 2.2. Anonymizing data is removing identifying information. There are several techniques that can anonymize data, however anonymization is always finding middle ground between keeping data useful for research (and attackers) and leaking information. Black marking can be used to leak the least amount of data. The last section, section 2.3, compares the features of the tools that can be used for anonymization in Table 2.4. There does not exist a tool that covers all features. There are tools that seem promising, such as anontool and pktanon.

We narrow the generic network traffic down to DDoS attack network traffic by answering RQ2: how could the generic practices on network anonymization be applied to DDoS network traffic? in chapter 3. In section 3.1 we specify which part of a network capture consists of DDoS network traffic. Since all other traffic is not related to the DDoS attack, it can be discarded. The fields and features that we identified in section 2.1 are reevaluated to be aimed at DDoS network traffic only in section 3.2. In section 3.3 we consider the tools that meet the requirements to be tested, and the new feature-set applied to these tools is given in Table 3.1. The preliminary conclusion is that the best tools are tcprewrite, TraceWrangler and Bit-Twist in terms of supported features. tcprewrite and Bit-Twist are the best tools for automated DDoS network traffic anonymization, where as TraceWrangler is the most easy to operate at the cost of not being scriptable.

In chapter 4 we answer RQ3: What is the best performing practice on DDoS attack network traffic anonymization? We tested all six tools for IP anonymization only in section 4.2. TCPurify uses the least time and has the lowest CPU usage, where as TraceWrangler has the second highest CPU usage and is the slowest tool. TraceWrangler however, uses the least amount of memory, all other tools use about twice the amount of memory TraceWrangler uses. In subsection 4.3.2 and subsection 4.3.1 tcprewrite, TraceWrangler and Bit-Twist are tested using both MAC and IP address anonymization. The CPU usage for each tool stays the same. The time tcprewrite and Bit-Twist take to complete anonymization of a file is dependent on the file size, where as the time TraceWrangler takes is dependent on the amount of packets in a capture file. In subsection 4.3.3 we noticed that the memory usage of tcprewrite, Bit-Twist and TraceWrangler scales based on the file size of the input file. TraceWrangler is the most efficient in memory usage, followed by tcprewrite. tcprewrite performs best in CPU usage and the time the tool takes to anonymize a file and performs well in memory usage, making tcprewrite the overall best tool of the tools tested.

The best practice to anonymize DDoS attack network traffic to protect the victims privacy while still enabling usage of the data for improving and detection and mitigation purposes is anonymizing the data using tcprewrite. However, tcprewrite does not cover all features as considered in chapter 3. This should be taken into account when using tcprewrite.

5.1 Future Work

While answering the main research question we noted that not all theoretical features for anonymization are covered yet. Out of all tools given by van Dijkhuizen and van der Ham (2018), only six tools were easy to install and use. Many tools are not supported anymore. Developing new tools and new features for existing tools is an important part of improving the practices to anonymize DDoS network traffic.

Considering the TCP/IP stack model, there is sufficient information available on the most used protocols such as Ethernet, IP and TCP/UDP. The protocols and situations that are used less often lack this information. This includes IPv6, ICMP and tunneling. The most obvious lack of support however is in the application layer. The application layer supports a lot of different protocols and each of these protocols has its own sensitive information. For DDoS network traffic, the information in application layer protocols is created by the attacker. Further research is needed to target and anonymize this information.

Bibliography

M. McKeay. DDOS BY THE NUMBERS, 2019. URL https://blogs.akamai.com/2018/06/summer-soti---ddos-by-the-numbers.html.

S. Kottler. February 28th DDoS Incident techreport, 2019. URL https://github.blog/2018-03-01-ddos-incident-report/.

Netflix. Internet Connection Speed Recommendations. URL https://help.netflix.com/en/node/306. Retrieved July 18, 2019.

C. Morales. NETSCOUT Arbor Confirms 1.7 Tbps DDoS Attack; The Terabit Attack Era Is Upon Us, 2019. URL https://www.netscout.com/blog/asert/netscout-arbor-confirms-17-tbps-ddos-attack-terabit-attack-era.

M. Majkowski. Memcrashed - Major amplification attacks from UDP port 11211, 2019. URL https://blog.cloudflare.com/memcrashed-major-amplification-attacks-from-port-11211/.

C. Schmoll, S. Teofili, E. Delzeri, G. Bianchi, I. Gojmerac, E. Hyytia, ..., and I. Venieris. State of the art on data protection algorithms for monitoring systems. Specific targeted research project, PRISM Consortium, 2008. URL http://fp7-prism.eu/images/upload/Deliverables/fp7-prism-wp3.1-d3.1.1-final.pdf. test.

N. van Dijkhuizen and J. van der Ham. A Survey of Network Traffic Anonymisation Techniques and Implementations. *ACM Comput. Surv.*, 51(3):52:1–52:27, May 2018. ISSN 0360-0300. doi: 10.1145/3182660. URL http://doi.acm.org/10.1145/3182660.

R. Pang, M. Allman, V. Paxson, and J. Lee. The Devil and Packet Trace Anonymization. *SIGCOMM Comput. Commun. Rev.*, 36(1):29–38, January 2006. ISSN 0146-4833. doi: 10.1145/1111322.1111330. URL http://doi.acm.org/10.1145/1111322.1111330.

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. URL http://arxiv.org/abs/0710.3979.

T. Kohno, A. Broido, and K. C. Claffy. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2(2):93–108, April 2005. ISSN 1545-5971. doi: 10.1109/TDSC.2005.26.

H. Beitollahi and G. Deconinck. Tackling Application-layer DDoS Attacks. *Procedia Computer Science*, 10:432 – 441, 2012. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2012.06.056. URL http://www.sciencedirect.com/science/article/pii/S1877050912004139. ANT 2012 and MobiWIS 2012.

M. McKeay and A. Fakhreddine. State of the Internet / Security. type, Akamai, 2017.

M. Dooley and T. Rooney. *DNS Security Management*. Wiley-IEEE Press, 2017. ISBN 9781119328292.

A. Young. Connection-less Lightweight X.500 Directory Access Protocol. RFC 1798, IETF, June 1995. URL https://www.rfc-editor.org/rfc/rfc1798.txt.

D.L. Mills. *Computer network time synchronization: the Network Time Protocol*. CRC Press, 2006. ISBN 9780849358050.

J. Postel. Character Generator Protocol. RFC 864, IETF, 1983. URL https://www.rfc-editor.org/rfc/rfc864.txt.

A. Presser, L. Farell, D. Kemp, W. Lupton, S. Tsuruyama, S. Albright, ..., and J. Fuller. UPnP Device Architecture 1.1, 2008. URL http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf.

R. Thurlow. RPC: Remote Procedure Call Protocol Specification Version 2. RFC 5531, IETF, 2009. URL https://www.rfc-editor.org/rfc/rfc5531.txt.

J. Case, M. Fedor, M. Schoffstall, and J. Davin. A Simple Network Management Protocol. RFC 1157, IETF, 1990. URL https://www.rfc-editor.org/rfc/rfc1157.txt.

G. Malkin. RIP Version 2. RFC 2453, IETF, 1998. URL https://www.rfc-editor.org/rfc/rfc2453.txt.

S. Cheshire and M. Krochmal. Multicast DNS. RFC 6762, IETF, 2013. URL https://www.rfc-editor.org/rfc/rfc6762.txt.

K. Sollins. THE TFTP PROTOCOL (REVISION 2). RFC 1350, IETF, 1992. URL `https://www.rfc-editor.org/rfc/rfc1350.txt`.

D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. Generic Routing Encapsulation (GRE). RFC 2784, IETF, 2000. URL `https://www.rfc-editor.org/rfc/rfc2784.txt`.

T. Brekne and A. Årnes. Circumventing IP-address pseudonymization. In *Communications and Computer Networks*, 2005.

M. Dworkin. Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption. Technical report, March 2016.

Omnipacket. SafePcap, 2019. URL `https://omnipacket.com/safepcap`.

M. Möller. Open Hardware Monitor, 2016. URL `https://openhardwaremonitor.org/`.

N. Griffiths. nmon for Linux, 2018. URL `http://nmon.sourceforge.net/pmwiki.php?n=Main.HomePage`.