

Automated DDoS Attack Fingerprinting by Mimicking the Actions of a Network Operator

K.W. van Hove
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
k.w.vanhove@student.utwente.nl

ABSTRACT

A Distributed Denial of Service (DDoS) attack is an attempt to overload a service. The main data a network operator has access to during a DDoS attack is the network traffic. An experienced operator would easily identify the attack. To the best of our knowledge, no solutions are based on the knowledge of the network operator. We will propose a tool which we call the *dissector* that mimics the steps taken by a network operator to identify the key characteristics of an attack. These characteristics can be used for, but are not limited to, mitigation purposes. These key characteristics form a DDoS fingerprint. The results of our research show >90% of attack traffic being covered by the generated fingerprints, with next to no legitimate traffic being detected as malicious, whilst running in linear time.

Keywords

DDoS Attack, DDoS Mitigation, DDoS Attack Characterisation, DDoS Attack Fingerprinting

1. INTRODUCTION

A Distributed Denial of Service (DDoS) attack is an attempt to overload a service, thereby disrupting regular traffic and making the service unavailable for its intended use. Attacks range from a student wanting to sabotage or delay an online exam to organised criminals holding an online business ransom [3]. It has been estimated that the monetary loss for Dutch companies from DDoS attacks alone exceeded one billion euros in 2018 [3].

When a network operator takes notice of an attack, either by using monitoring services or complaints about a service interruption, they want to mitigate it as quickly as possible. In order to do that, they need a summary of the characteristics of the attack. A summary that tells them what the attack is and where the attack is coming from (for example, all DNS traffic over UDP, port 53 from IP address x , y and z), in order for them to create rules for their specific hardware or software that will block the specific attack.

The main data a network operator has access to is the network traffic – the packets coming in and out of the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

31st Twente Student Conference on IT July 5th, 2019, Enschede, The Netherlands.

Copyright 2019, University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science.

network. A network operator can already discern unusual patterns by looking at the network traffic. Every DDoS attack has an associated pattern, something that makes it stand out from ordinary traffic. However, with the current size of DDoS attacks, averaging 26.37 Gbps in the second quartile of 2018 [17], manual pattern recognition becomes infeasible in real time scenarios.

There have been several models for detecting traffic that stands out from normal traffic [5] [6] [21]. However, these methods are not suitable to deal with our scenario, as they require a protocol change or cannot be used with existing rule-based filtering hardware. While some papers describe a characterisation method [6] [15] [25], those methods are based on an active analysis of the current internet traffic, instead of a previously defined characterisation, making them unsuitable for a generic firewall rule.

The problem is thus that (1) none of the methods are generic, they cannot be used for purposes other than mitigation; (2) none of the solutions use the expertise of the network operator for automating the process. We do not directly propose a new detection or mitigation method, but rather a system that extracts key characteristics of a DDoS attack, which can further be used for mitigation, but also for other purposes such as attribution and correlation.

This paper proposes an automated method of fingerprinting DDoS attacks, by investigating the steps a network operator would take to mitigate a DDoS attack. We identify six requirements for the proposed fingerprint generation system. The generated fingerprint should allow the network operator to (1) use their existing hardware to mitigate the attack; (2) attribute the person responsible; (3) reproduce the attack in an academic setting; (4) correlate the attack with similar characteristics; (5) notify the owners of the abused machines so that they can take action and (6) give an insight in the global DDoS attack activity. To achieve these requirements, we define the following three main research questions (RQ):

1. What is the state of art of DDoS attack characterisation?
2. How can we generate a fingerprint that meets these six requirements?
3. What is the performance of our generated fingerprint in a real life scenario?

The further outline of this paper is as follows: Section 2 covers the current state of art when it comes to DDoS attack characterisation. Section 3 covers the assumed available hardware. Section 4 covers our proposal for an alternative way to fingerprint DDoS attacks. Section 5 verifies

whether the design from section 4 meets the six identified requirements. Section 6 evaluates the performance of the design from section 4.

2. STATE OF ART

In this section we present an overview of current DDoS characterisation and mitigation methods. In particular, in this section we answer RQ1: “What is the state of art of DDoS attack characterisation?”. The research can be distinguished into two main categories: statistical traffic analysis and anomaly detection. Both those categories can be subdivided into solutions that can be applied by a single institution, and solutions that require cooperation of a large part of the network. Most methods focus on the mitigation of DDoS attacks, where the characterisation is a means to mitigate the attack. As our goal is broader than just mitigation, we will focus on the characterisation part, and judge its (1) comprehensiveness, (2) distinguishability, (3) extraction time, and (4) general applicability.

Chonka et al. [6] describe a method based on chaos theory, which looks at the regular packet flow and trains a neural network to detect anomalies (mainly IP spoofing). Once an anomaly is detected, it determines based on the Lyapunov Equation whether this anomaly is probable to be caused by normal traffic fluctuations, or that the anomaly is likely to be caused by a DDoS attack. Saied et al. [19] describe a similar method. The problem here is that the result is not comprehensive or generally applicable. It may work very well for mitigation, but for other purposes it is not so well suited. We cannot point to specific features or patterns and say “that is what caused the DDoS attack”, making it unsuitable for, for example, correlation.

Li [15] describes a statistical model for identifying abnormal variations of long-range dependent time series, using statistical pattern recognition. Thapngam et al. [21] describe a statistical model for differentiating between a flash crowd – a sudden peak in network traffic – and a DDoS attack, by looking at packet features and their frequency, and determining the likelihood of the packet belonging to a genuine request. Yu et al. [24] describe a method based on analysing the properties of large peak traffic flow distributions, and thereby determining whether traffic is legitimate or belongs to a DDoS attack. The problem with these statistical methods is that most of them require a baseline measurement to function. This severely limits their general applicability. Additionally, they do tend to not function on regular firewalls and other rule-based network filtering systems.

Snoeren et al. [20] describe a design where the trail of an IP packet is delivered along with the packet, in order to allow the recipient to determine the origin of the packet. Ferguson et al. [10] employ a similar technique, though instead of letting the recipient trace the packet to the origin, they let every hop block traffic that was detected to spoof its source IP address, by detecting whether such traffic would pass through that node in the network. Yaar et al. [23] propose a system called SIFF, where the sender of the traffic is verified using a handshake, so that a host under attack can selectively drop unverified traffic, putting the burden of verification on the communicating parties instead of on the nodes connecting them. These methods are very comprehensive, distinguishable, are fast to extract and are generally applicable. This would create a great way to characterise DDoS attack if the protocols were generally implemented. The problem is that they require a change of how the internet currently works, as protocols need to be changed. Such a substantial change is

difficult to accomplish, as can be seen by the effort taken to get IPv6 supported at large scale, which after twenty years has still not risen above 50% coverage [2]. The benefit becomes a lot smaller if they are only implemented on a small scale, as both legitimate and non-legitimate traffic would decide not to implement it, which means we end up at the same problem, only with a slightly smaller dataset.

Choi et al. [5] describe a detection and mitigation method for HTTP traffic by defining normal user behaviour and frequency, and blocking traffic that exceeds a certain threshold, effectively applying rate limiting on all traffic. The characterisation is very comprehensive, as it contains all information of the HTTP traffic, is very distinguishable, as only the packets that show “odd” behaviour are targeted, and can be quickly extracted. The issue is that it requires extensive knowledge of the system, and only works for HTTP traffic. This makes it more difficult to apply generally. Ioannidis et al. [14] describe a more generalised variant; they describe mitigation method meant for routers, where, when the routers detect congestion, they will check aggregates (a filtering on certain properties, e.g. “packets to destination D ”, “TCP SYN packets”), then determine the aggregates causing the congestion, and drop them. They then push those filters upstream towards the source of the traffic. The idea of using aggregates is gives a comprehensive overview of the attack, the extraction time is low and it is generally applicable. The problem lies in finding the right aggregate. One way of finding such an aggregate would be using a method proposed by Yuan et al. [25]. They describe a method of characterising a DDoS attack using deep learning, where machine learning detects based on the features in the packet whether it belongs to a DDoS attack or not. It is comprehensive, it is distinguishable, and it is generally applicable. However, the extraction time is relatively slow, due to the constant checking each packet and consequently updating the model. It also raises the question: Why would we not harness the knowledge of the network operator, instead of letting machine learning decide what the right course of action is?

This paper will propose a generic approach, that requires no change in protocols or prior knowledge about the system, that extracts a summary of the characteristics of the attack that comprehensive, distinguishable, has a low extraction time, and is generally applicable, and that can be used for further purposes.

3. PRESUMED EXISTING HARDWARE

We have mentioned the ability to mitigate attacks using common, regular hardware as one of the requirements, but we have not yet defined what we consider to be common hardware. Firewalls are the most common blocking hardware, and Cisco, Fortinet, and Palo Alto are among the most common firewall vendors found in businesses [13]. The thing that the firewalls that these vendors provide have in common is that they are rule-based [7] [9] [1], meaning that they take a set of rules (e.g. “Accept incoming TCP traffic from 1.1.1.0/24 to port 25” and “Reject all”), and apply them top to bottom. If the incoming packet matches a rule, it will be accepted/rejected based on what the rule specifies. In this case, a TCP packet from 1.1.1.25 to port 25 would be accepted, because it matches the first rule, and a packet from 2.2.2.16 would be rejected. Generally, these systems also remember some for of state, so that a packet as a response to an accepted packet will also be accepted. A newer tool which is becoming increasingly more common in larger networks is BGP FlowSpec [12]. This system works by telling the upstream providers

to drop all traffic to a certain IP address based on certain filtering rules. This is once again rule-based.

Whilst there are specific hardware solutions for DDoS mitigation [11] [16], they generally function as a black box, making them unsuitable for other purposes than mitigation.

4. FINGERPRINT GENERATION ALGORITHM

This section will describe the generation part of RQ2, “How can we generate a fingerprint that meets these six requirements?”. We will outline the steps our tool, which we call the *dissector*¹, follows to determine and extract the characteristics of a DDoS attack. We assume that we have a packet capture of the network traffic at the time of the attack.

The first step a network operator takes is determining what the target of the attack is, after all, they need to know what is being attacked, before they can mitigate the attack. This is done by looking at the destination IP address of the traffic, where a large amount of packets per second for a certain IP address indicates the attack target (there are cases where this does not hold, for example in the case of specific application layer attacks [18] or UDP floods, but this will be able to detect the majority of volumetric attacks and protocol attacks).

The second step is determining what the majority of the traffic of that IP is, by taking the internet layer protocol. If it is IP, then we take the transport layer protocol (e.g. UDP or TCP). Most attacks abuse a specific application protocol, such as DNS or NTP, or generally flood the server with UDP or TCP traffic [8].

The third step is figuring out what the most likely type of the attack is. From the packets that are left, we take the distribution of source ports and destination ports. Most attacks have a source/destination port relation of one-to-many, many-to-one or one-to-one. For example, in the case of a DNS amplification attack, all traffic will come from source port 53 and will go to a random destination port. We determine the source or destination port that occurred the most, so source port 53 in the example above. We then filter based on that port we found.

The fourth step is to extract extra information from that attack. From the traffic that is left, we extract extra information for that specific protocol. For example, in the case of DNS amplification attack (UDP over port 53), we extract the DNS query that was sent. In the case of NTP flooding (UDP over port 123), we extract the request code. In the case of a port number of 0, we actually have to do with a UDP fragmentation attack, where the packets received cannot be reassembled into a larger packet. In this case we indicate that we have suffered from a fragmentation attack.

If this information is not available, for example in the case of a TCP SYN flood, and we consider identifying the attack as all TCP traffic to for example port 80 too crude, we can filter based on the most frequent Time-To-Live (TTL). In the case of a TCP SYN flood, the IP addresses are spoofed, so we cannot rely on their frequency, but the TTL for packets originating from the same server will most likely be equal. Using this metric, we can make an accurate filter for the spoofed traffic [22].

It is also possible for a server to be under an ICMP flood

attack, where the attacker attempts to overwhelm a targeted device with ICMP echo-request packets, causing the target to become inaccessible to normal traffic [8]. In this case, we can extract the ICMP type and code.

This list is by no means exhaustive – there are many more details that can be extracted from other attack vectors. We have chosen to focus on these examples due to their popularity, and we could not include all types of attack vectors due to time constraints, but many more packet attributes can help us narrow down the attack vector.

We have now determined the destination IP, transport protocol, source/destination ports and application protocol. We can thus now filter the traffic based on those attributes, and extract the source IP addresses that were involved in the attack. All this information combined (source IP addresses, destination ports, source ports, transport layer protocol, application layer protocol and specific application layer attributes) can be used to filter the traffic. We call this summary of information a *fingerprint*. However, DDoS attacks generally consist of multiple attack vectors, like a simultaneous DNS amplification and TCP flooding attack. We repeat the process for the remaining traffic, until there are no attack vectors left. This is the case when we can no longer see a peak in traffic from multiple IP addresses on one vector. We can link these attacks together using a unique key.

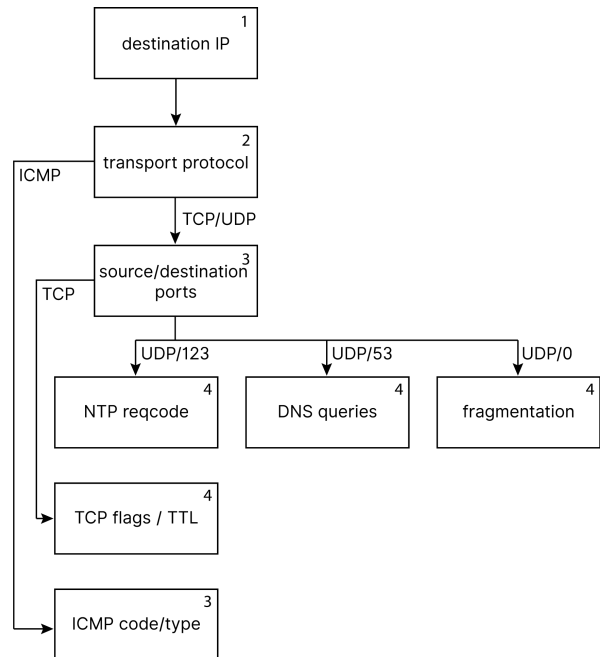


Figure 1. Overview of the steps involved in generating a fingerprint for an attack vector

4.1 Fine-graining the approach

We have previously mentioned that the list of protocol-specific characterisations shown in figure 1 is not exhaustive. One may wonder whether this approach is viable at all. Let us take a TCP SYN flood attack on port 80 of a web server as an example. The dissector determines the protocol (TCP) and the destination port (80), and the TCP flag (SYN). Whilst TCP packets to port 80 with a SYN flag set indeed belong to the attack, a lot of non-attack traffic also shares the same characteristics. We cannot rely on the source IP addresses, as they can be spoofed. In these moments, it is good to ask ourselves

¹https://github.com/ddos-clearing-house/ddos_dissector

what a network operator would do. In this case, a network operator would turn to other packet attributes, such as the TTL. If we notice that a lot of packets from different ASes share the same TTL, we can reasonably assume that something is wrong. Similarly, if we see a lot of traffic from the same AS, but with very different TTLs, we can also assume that something is wrong (with the exception of anycast addresses). These are things that stand out. Other aggregates as defined in Ioannidis et al. [14] are also possible. Additionally, the attacker may use packets of the same size. If we suddenly notice an unusual amount of packets with length x , then that is suspicious, and may be the basis of a characterisation for the attack. Similarly, if an attacker uses random data, with random lengths, but does not calculate the checksum properly, then the fact that the checksum is invalid also characterises the attack.

Additionally, one may wonder why we can claim that most attacks have a source-destination port relation of one-to-one, one-to-many or many-to-one. Let us take the list of most common non-application layer DDoS attacks provided by Cloudflare [8], and review them one by one:

- Memcached attack – This is an amplification attack, so all traffic will originate from one port (11211);
- NTP amplification attack – This is also an amplification attack, so all traffic will originate from one port (123);
- DNS amplification attack – This is again an amplification attack, so all traffic will originate from one port (53);
- SSDP attack – This is a reflection attack, so all traffic will originate from one port (1900);
- DNS flood – Here the attacker floods the DNS server, so all traffic will have one port as destination (53);
- HTTP flood – Similarly to a DNS flood, the attacker floods one destination port (80);
- SYN flood – This again floods one service with SYN packets with spoofed IP addresses, trying to exhaust the maximum number of connections. In order for this to be effective, all traffic will have one destination port;
- UDP flood – Here the attacker floods the server with UDP packets, and tries to overload the server trying to handle them. Here is a case of an attack that can be many-to-many, however, a firewall can be configured to simply drop UDP packets that have an unreachable destination port;
- ICMP flood – ICMP does not use TCP or UDP, and has no port numbers;

As one can see, in most cases our assumption holds.

This approach starts to become troublesome is when there are specific application layer attacks, for example by requesting a web page that takes a lot of processing power to render. In these cases, our peak analysis may not yield the correct result (as there may be relatively few packets associated to the attack), and an approach as proposed by Choi et al. [5] may yield better results.

5. FINGERPRINT APPLICATION

Now that we have designed an algorithm for generating DDoS fingerprints, we need to verify whether it can also satisfy the six conditions we set as part of RQ2.

The first condition, that the network operator is able to use their existing hardware to mitigate the attack, is clearly met, as from the fingerprint firewall rules can be derived.

The second condition, that we could attribute the attack to the person responsible, is partially met. For that law enforcement needs to be able to figure out where the attacker is located. This is not something we can fully solve technologically, but we can aid them by enriching the fingerprint. For example, for every source IP address, we can determine the AS number and country it belongs to. The caveat is that this information can change over time, hence we either need to collect that data at the moment of the attack, or we need the start time of the attack, so we can enrich it later with the right data from that time period. This is information that can be added to the fingerprint without any issue, which fulfils this requirement.

The third condition, that the attack can be reproduced in an academic setting, is partially met. To fully meet that requirement, we also need the packets per second, bits per second and the duration of the attack. Then the attack can be replicated using the filtering info from the fingerprint, by creating packets that match the fingerprint, and firing them at the same rate as the attack. These three statistics can be easily obtained from a sample of the attack. Alternatively, and preferably, the capture of the attack can be stored. Then it is possible to fully replay the attack in a locked down setting, which fulfils this requirement.

The fourth condition, that we can correlate the attack with other attacks with similar characteristics is met, because our fingerprint is generic, and does not depend on the network infrastructure or specifics of the application. We can hence check the similarity between two fingerprints.

The fifth condition, notifying the owners of abused machines, and the sixth condition, giving an insight of global DDoS attack activity, are not yet met, because they require a large dataset of DDoS fingerprints to function.

5.1 Fingerprint sharing and anonymisation

We have so far seen that the fifth and sixth requirement are not met due to a lack of data. For these requirements to be met, we need to incentivise sharing. However, the packet capture may contain privacy sensitive data, and as a result one may not be inclined to share their data.

For the packet capture of the attack, we can filter the data using the attack vectors found, and then remove the information we do not need. We do not need to keep the destination IP of an attack, nor do we need to keep the source and destination MAC addresses, so we can replace this values with 0-bytes. These details can give someone else insight in the internal infrastructure, and can give an attacker the upper hand. To alleviate these concerns, we remove this information. This results in an anonymised packet capture that can be shared without concerns. The fingerprint itself, as defined in figure 3, does not contain any data that could trace it back to the victim, so that can be shared without problems as well.

Now that we have addressed the privacy concerns, we assume that people are willing to share their anonymised packet captures and fingerprints. We thus assume that we have a large collection of fingerprints that we can use for analysis. Now let us get back to the original six conditions.

```
INPUT -m state --state NEW -m udp -p udp --
sport 53 -d 192.168.1.1 -j REJECT
```

Figure 2. An example of a firewall rule based on the fingerprint of figure 3 for Linux’ iptables

The fifth condition is now met. Our fingerprint also contains the source IP addresses. Using those, the owner of the machine can be located, and the local CERT/C-SIRT team can be informed that their machines are being abused. Alternatively, the CERT/CSIRT team could scan the fingerprints for their IP ranges.

The sixth condition is now met as well. If a lot of these attacks are collected in a central place, or shared among others, they can be used to give an insight in the trends of global DDoS attack activity, and they can be used to analyse trends.

This means that the final fingerprint, as shown in figure 3, meets all the six requirements we set, and is comprehensive, distinguishable, and generic.

One of the most common use cases for this generic fingerprint will be creating firewall rules based on it. We have already established the properties of the most common firewalls in section 3, and this section will describe how we can turn a fingerprint in a set of rules for a firewall. Let us take the fingerprint for figure 3 as an example. It is a DNS amplification attack originating from port 53. We can create a rule that drops all packets matching UDP, source port 53, where the state is new (as to not block replies to DNS request from inside the network) and the destination IP is the server being attacked. The same method can be applied to all attack types. It is good to notice that all filters we detected using the algorithm we described in section 4 can be implemented in a common firewall, as they only try to match certain fields. For example, a TCP reset attack directed at port 80 can be filtered by creating a rule that drops TCP traffic with a destination port of 80, the TCP reset flag set, and a specific TTL or AS.

6. PERFORMANCE EVALUATION

6.1 Detection performance

Now that we have a way to characterise DDoS attacks, and generate fingerprints based on those findings, we need to verify how well they fare in real situation. We test that using the following method:

- We take a packet capture where we know what packets belong to an attack, and what packets do not, by combining known DDoS traffic provided by the Masaryk University and a known good packet capture of a test machine running several services;
- We run the dissector on the packet capture;
- We then check whether it detected the right attack(s);
- We check how many attack packets are in the filtered packet capture and how many good packets are in the packet capture.

Our aim is to filter most bad (attack) traffic, while not filtering the good (non-attack) traffic. We do this for the three different types of attacks listed below. We define positive traffic as traffic that was part of the attack. The

```
{
  "src_ips": [
    {
      "cc": "US",
      "ip": "192.168.125.12",
      "as": 12345
    },
    ...
  ],
  "dst_ports": [
    1234,
    4567,
    ...
  ],
  "src_ports": [
    53
  ],
  "tl_protocol": "UDP",
  "app_protocol": "DNS",
  "key": "1d114fcbacdd54b5a60d353087f88b6",
  "avg_pps": 1000,
  "avg_bps": 1000000,
  "duration": 30.25,
  "start_timestamp": 1419262727
}
```

Figure 3. An example of a DDoS fingerprint for an attack vector

DDoS traffic dataset provided by Masaryk University consists of 272 attack traces from actual DDoS attacks, which were ordered from DDoS-as-a-Service providers. The traces recorded all network traffic, one at a time for at most 300 seconds per attack. The server was hosting a dummy webpage, and was not used for any other traffic [4].

We highlight the findings for four different attacks, and analyse them in detail.

(1) DNS amplification attack

Bad traffic: IPSTR-DNS-S-01_2014-12-22_15_38_41

Good traffic: GOOD-01_2019-05-24_19_42_21

This is a DNS amplification attack aimed at port 80. It uses open DNS resolvers to request DNS data, and lets the resolver reply to the victim. This way, a single request can cause a lot more data to be sent [8]. This sample has 191,416 packets, of which 185,576 belong to the attack. Positive here means that it was detected as part of the attack.

	Actual positive	Actual negative
Detected positive	98% (181,698)	0% (0)
Detected negative	2% (3,878)	100% (5,840)

We can see that for this example, the accuracy was rather good. > 95% of the attack was able to be detected, with 0% of legitimate traffic being blocked.

(2) TCP SYN flood attack

Bad traffic: RAGE-ESSYN-S-02_2015-01-03_23_20_03

Good traffic: GOOD-01_2019-05-24_19_42_21

This is an attack with a single vector. It floods the victim with TCP SYN packets, thereby trying to exhaust the server’s resources. The server will try to respond to the TCP SYN packets with a SYN ACK, but, because the IP addresses are spoofed, they will never be received [8]. This sample has 469,256 packets, of which 463,416 belong to the attack. Positive here means that it was detected as part of the attack.

	Actual positive	Actual negative
Detected positive	99% (461,010)	0% (0)
Detected negative	1% (2,406)	100% (5,840)

We can see that for this example, the accuracy was again rather good. > 95% of the attack was able to be detected, with 0% of legitimate traffic being blocked. In this case our TTL detection makes sure that only spoofed packets are blocked, and that all other traffic can still reach its destination.

(3) SSDP and DNS attack

Bad traffic: REST-RUDY-S-01_2014-12-29_16_08_33

Good traffic: GOOD-02_2019-05-25_20_11_01

This is an attack with two vectors. One of them is a DNS amplification attack, and the other is an SSDP attack. An SSDP attack is an attack where UPnP devices are abused to amplify UDP traffic [8]. In this case it asked devices to send requests to port 80. This sample has 10,015 packets, of which 6,371 belong to the attack. Positive here means that it was detected as part of the attack.

	Actual positive	Actual negative
Detected positive	55% (3,525)	0% (0)
Detected negative	45% (2,846)	100% (3,644)

In this case, only the SSDP attack was detected – the DNS attack was not discovered. Additionally, the filter that was proposed was to block all UDP traffic to port 80. In the test environment, this would not cause any issues, but in real life it may. This can be solved by adding more fine-grained detection methods for SSDP to the dissector. TTL analysis cannot help in this case, as the packets are coming from many amplifiers, and not from one attacker’s server. However, a large part of the attack was still mitigated.

(4) NTP amplification attack

Bad traffic: DEST-NTP-S-02_2014-12-29_10_38_51

Good traffic: GOOD-03_2019-05-26_14_53_12

This is an attack with a single vector. It sends packets with a spoofed IP address to a public NTP server, which then sends a large amount of data to the victim’s IP address [8]. This is similar to a DNS amplification attack. This sample has 561,623 packets, of which 547,892 belong to the attack. Positive here means that it was detected as part of the attack.

	Actual positive	Actual negative
Detected positive	100% (545,567)	25% (3,365)
Detected negative	0% (2,325)	75% (10,366)

We can see that in this case, nearly all malicious traffic was detected. However, we also notice that 25% of good traffic was filtered. The reason for this is that the good traffic set contains a lot of HTTPS traffic (TCP on port 443) to a set of servers with similar TTLs, which is then wrongly identified as an attack. This can be resolved by increasing the threshold for HTTPS traffic to be detected as part of the attack.

Other attacks

Here we give an overview of the performance of several tests with other attack sets, as shown in table 1. The good traffic that was used is *GOOD-03_2019-05-26_14_53_12*. This set contains a lot of HTTPS traffic as well, which is sometimes detected as a TCP attack, which explains the 25% legitimate traffic being detected as an attack (just like with the NTP attack mentioned above). This is something that has to be resolved in the future by fine-tuning the thresholds. Additionally, the filtering process sometimes fails to filter on all detected attack vectors with DNS attacks, causing negative values and values over 100% to appear.

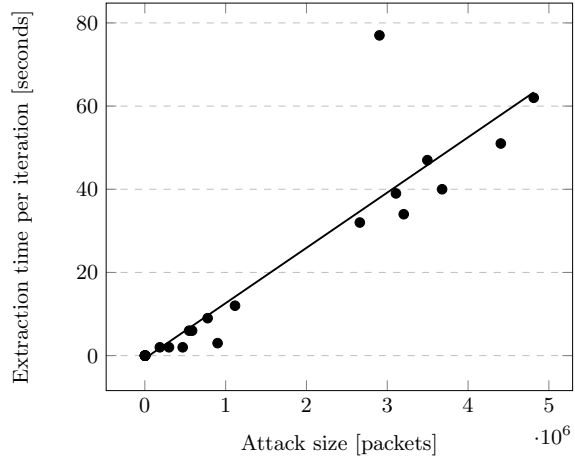


Figure 4. The size of the attack (in packets) plotted against the extraction time per iteration (in seconds)

However, apart from these inconsistencies, the overall results are rather promising, with a very low amount of false positives and a large amount of actual positives.

6.2 Speed performance evaluation

We have also mentioned that we wish our characterisation to be performant. We have taken a random sample of 26 attacks, and tested the extraction speed. Figure 4 shows the speed per iteration (in seconds) plotted against the size of the attack (in packets). Please note that if there are multiple attack vectors, that there will be multiple iteration, and hence the extraction time may be longer. This increase will be constant.

As figure 4 shows, the dissector runs in a near-linear fashion. During the measurement process, we observed that the greatest speed bottleneck at the moment is the speed of the hard drive. We noticed that our memory usage or CPU usage rarely got above 50%, whilst the hard drive would often be working at 100%. A relatively small amount of time was spent on the analysis part, and a significantly larger amount of time was spent on writing the result to the disk. We also noticed TCP attacks take significantly longer to dissect than UDP packets, due to the TCP attack both filtering on the top TCP flag and on the TTL, thereby having to walk through all packets more often (which shows in the large outlier visible in figure 4). We expect that significant performance improvements are possible by keeping track of a count for all packet data from the start, thereby only having to visit a packet once per iteration, instead of going over each packet once per attribute; this change would allow the dissector to run in true linear fashion.

7. CONCLUSION

This paper proposed an automated method of fingerprinting DDoS attacks. We have looked at (1) the state of art of DDoS attack characterisation, (2) the generation a fingerprint that meets the six requirements (to mitigate, attribute, reproduce and correlate the attack, notify the owners of the machines used for the attack, and to create an insight into global DDoS attack activity), and (3) the performance of our generated fingerprint. We have concluded that none of the current methods are (1) comprehensive, (2) distinguishable, (3) generally applicable, and (4) have a low extraction time. We have designed a tool which characterises a DDoS attack by mimicking the way

a network operator would analyse a DDoS attack. The tool allows outputs a summary that can be used to create firewall rules for hardware that is currently common. This characterisation also meets the six requirements described above, and is also comprehensive, distinguishable and generally applicable. The performance measurements show that it is also has a low extraction time and a high accuracy. The main limitation is having to adjust the dissector for every type of attack in order to filter on the proper fields. This paper focused on DNS, NTP, ICMP and TCP attacks, but we expect the same approach to work for other kinds of attacks as well. We believe that this contribution can significantly reduce the impact of DDoS attacks in the future, as our DDoS fingerprint can not only be used for mitigation, but also to aid other DDoS research.

Future research would have to look into expanding the data extraction for more application layer protocols, as well as looking into a way to automatically do that for previously unknown attack types. Additionally, research is required to make sure that the amount of non-malicious traffic being detected as malicious is reduced. Future research should also look into setting up the framework for the sharing of fingerprints for correlation and attribution.

References

- [1] Palo Alto. *PAN-OS® Administrator's Guide*. May 2019. URL: <https://docs.paloaltonetworks.com/documentation/80/pan-os/pan-os> (visited on 06/20/2019).
- [2] APNIC. "IPv6 Capable Rate by country". URL: <https://stats.labs.apnic.net/ipv6> (visited on 05/01/2019).
- [3] N. Boerman et al. *The impact of DDoS attacks on Dutch enterprises*. Nov. 2018. URL: <https://www.nbip.nl/wp-content/uploads/2018/11/NBIP-SIDN-DDoS-impact-report.pdf> (visited on 05/01/2019).
- [4] V. Bukač. "Small scale denial of service attacks [online]". Doctoral theses, Dissertations. Masaryk University, Faculty of Informatics, Brno, 2015 [cit. 2019-06-27].
- [5] J. Choi et al. "A method of DDoS attack detection using HTTP packet pattern and rule engine in cloud computing environment". In: *Soft Computing* 18.9 (2014), pp. 1697–1703.
- [6] A. Chonka, J. Singh, and W. Zhou. "Chaos theory based detection against network mimicking DDoS attacks". In: *IEEE Communications Letters* 13.9 (2009), pp. 717–719.
- [7] *Cisco ASA 5500 Series Configuration Guide using the CLI, 8.4 and 8.6 - Configuring Access Rules [Cisco ASA 5500-X Series Firewalls]*. Nov. 2018. URL: https://www.cisco.com/c/en/us/td/docs/security/asa/asa84/configuration/guide/asa_84_cli_config/access_rules.html (visited on 06/20/2019).
- [8] Cloudflare. *What Is a Distributed Denial-of-Service (DDoS) Attack?* URL: <https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/> (visited on 06/17/2019).
- [9] *Configuring a firewall policy*. URL: https://help.fortinet.com/fadc/4-8-0/olh/Content/FortiADC/handbook/firewall_policy.htm (visited on 06/20/2019).
- [10] P. Ferguson and D. Senie. *Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing*. United States, 2000.
- [11] Fortinet. *FortiSandbox*. URL: <https://www.fortinet.com/products/sandbox/fortisandbox.html> (visited on 06/20/2019).
- [12] N. Hinze et al. "On the Potential of BGP Flowspec for DDoS Mitigation at Two Sources". In: *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos - SIGCOMM 18* (Aug. 2018), pp. 57–59. DOI: 10.1145/3234200.3234209.
- [13] CTC Technologies Inc. *5 Top Firewall Providers for 2019*. Jan. 2019. URL: <https://www.ctctechnologies.com/5-top-firewall-providers-for-2019/> (visited on 06/20/2019).
- [14] J. Ioannidis and S.M. Bellovin. "Implementing push-back: Router-based defense against DDoS attacks". In: *Network and Distributed System Security Symposium*. 2002.
- [15] M. Li. "An approach to reliably identifying signs of DDOS flood attacks based on LRD traffic pattern recognition". In: *Computers & Security* 23.7 (2004), pp. 549–558. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2004.04.005>.
- [16] NETSCOUT. *Arbor Threat Mitigation System*. URL: <https://www.netscout.com/product/arbor-threat-mitigation-system> (visited on 06/20/2019).
- [17] Nexusguard. *DDoS Threats Report 2018 Q2*. URL: <https://www.nexusguard.com/threat-report-q2-2018> (visited on 05/01/2019).
- [18] S. Ranjan et al. "DDoS-Shield: DDoS-Resilient Scheduling to Counter Application Layer Attacks". In: *IEEE/ACM Transactions on Networking* 17.1 (Feb. 2009), pp. 26–39. ISSN: 1063-6692. DOI: 10.1109/TNET.2008.926503.
- [19] A. Saied, R. E. Overill, and T. Radzik. "Detection of known and unknown DDoS attacks using Artificial Neural Networks". In: *Neurocomputing* 172 (2016), pp. 385–393. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2015.04.101.
- [20] A.C. Snoeren et al. "Single-packet IP traceback". In: *IEEE/ACM Transactions on Networking (ToN)* 10.6 (2002), pp. 721–734.
- [21] T. Thapngam et al. *Discriminating DDoS attack traffic from flash crowd through packet arrival patterns*. Apr. 2011, pp. 952–957. DOI: 10.1109/INFCOMW.2011.5928950.
- [22] H. Wang, C. Jin, and K. G. Shin. "Defense Against Spoofed IP Traffic Using Hop-count Filtering". In: *IEEE/ACM Trans. Netw.* 15.1 (Feb. 2007), pp. 40–53. ISSN: 1063-6692. DOI: 10.1109/TNET.2006.890133. URL: <http://dx.doi.org/10.1109/TNET.2006.890133>.
- [23] A. Yaar, A. Perrig, and D. Song. "SIFF: A stateless Internet flow filter to mitigate DDoS flooding attacks". In: *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*. IEEE. 2004, pp. 130–143.
- [24] S. Yu, W. Zhou, and R. Doss. "Information theory based detection against network behavior mimicking DDoS attacks". In: *IEEE Communications Letters* 12.4 (Apr. 2008), pp. 318–321. ISSN: 1089-7798. DOI: 10.1109/LCOMM.2008.072049.
- [25] X. Yuan, C. Li, and X. Li. "DeepDefense: Identifying DDoS Attack via Deep Learning". In: *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*. May 2017, pp. 1–8. DOI: 10.1109/SMARTCOMP.2017.7946998.

Table 1. An overview of the results of the dissector for several attacks

AP: Actual Positive DP: Detected Positive AN: Actual Negative DN: Detected Negative

	AP/DP	AP/DN	AN/DN	AN/DP
ANON-SSYN-S-01_2014-12-20_20_04_11	71% (2768982)	29% (1132264)	75% (10366)	25% (3365)
ANON-SSYN-S-02_2014-12-20_19_41_22	71% (2649501)	29% (1072680)	75% (10366)	25% (3365)
BOOTIO-NTP-S-01_2015-03-26_15	99% (1919473)	1% (12241)	75% (10366)	25% (3365)
BOOTIO-SSYN-S-01_2015-03-26_16	78% (968858)	22% (272390)	75% (10366)	25% (3365)
BOOTIO-SSYN-S-02_2015-03-26_16	78% (2564576)	22% (730043)	75% (10366)	25% (3365)
CONS-SSYN-S-01_2014-12-20_20_17_05	71% (2585531)	29% (1038006)	75% (10366)	25% (3365)
CONS-SSYN-S-02_2014-12-22_15_29_38	71% (2485299)	29% (997426)	75% (10366)	25% (3365)
DEST-ESSYN-S-02_2014-12-29_15_05_10	52% (18835)	48% (17178)	100% (13725)	0% (6)
DEST-NTP-S-02_2014-12-29_10_38_51	100% (545567)	0% (2325)	75% (10366)	25% (3365)
HORNY-DNS-S-02_2015-04-07_20_31_25	145% (1738190)	-45% (-542114)	100% (13668)	0% (63)
HORNY-ESSYN-S-01_2015-04-07_18_16_10	90% (1130076)	10% (119136)	75% (10366)	25% (3365)
HORNY-ESSYN-S-02_2015-04-07_21_06_06	90% (1133480)	10% (121094)	75% (10366)	25% (3365)
HORNY-SYNACK-S-01_2015-04-07_18_27_15	90% (1153262)	10% (123293)	75% (10366)	25% (3365)
HORNY-SYNACK-S-02_2015-04-07_21_17_56	91% (3071319)	9% (314926)	75% (10366)	25% (3365)
HORNY-TCPFIN-S-01_2015-04-07_18_45_52	0% (0)	100% (30710)	100% (13731)	0% (0)
HORNY-TCPFIN-S-02_2015-04-07_21_29_07	83% (2418354)	17% (484980)	75% (10366)	25% (3365)
HORNY-TCPFUCK-S-01_2015-04-07_17_27_30	99% (182836)	1% (2699)	75% (10366)	25% (3365)
HORNY-TCPPSH-S-01_2015-04-07_18_51_59	0% (0)	100% (3158284)	100% (13731)	0% (0)
HORNY-TCPPSH-S-02_2015-04-07_21_34_57	83% (3215543)	17% (646757)	75% (10366)	25% (3365)
HORNY-TCPRST-S-01_2015-04-07_18_32_45	83% (535771)	17% (109328)	75% (10366)	25% (3365)
HORNY-TCPSEQ-S-01_2015-04-07_17_39_13	0% (0)	100% (44428)	100% (13731)	0% (0)
IPSTR-DNS-S-01_2014-12-22_15_38_41	97% (180522)	3% (5054)	75% (10366)	25% (3365)
IPSTR-DNS-S-02_2014-12-25_15_31_46	95% (283823)	5% (14879)	75% (10366)	25% (3365)
IPSTR-SYN-S-01_2014-12-21_19_38_21	0% (0)	100% (4976616)	100% (13731)	0% (0)
IPSTR-SYN-S-02_2014-12-21_20_25_15	0% (0)	100% (5052824)	100% (13731)	0% (0)
KRYPT-DNS-S-01_2014-12-20_20_44_56	196% (1228245)	-96% (-603016)	75% (10366)	25% (3365)
KRYPT-DNS-S-02_2014-12-21_21_12_12	100% (582485)	0% (2388)	75% (10366)	25% (3365)
KRYPT-ESSYN-S-01_2014-12-22_16_23_50	98% (2447100)	2% (58616)	75% (10366)	25% (3365)
KRYPT-ESSYN-S-02_2014-12-23_23_08_06	97% (871654)	3% (27041)	75% (10366)	25% (3365)
KRYPT-SSYN-S-01_2014-12-20_20_51_41	98% (2748668)	2% (61664)	75% (10366)	25% (3365)
KRYPT-SSYN-S-02_2014-12-21_21_04_11	85% (2298640)	15% (392280)	75% (10366)	25% (3365)
KRYPT-XSYN-S-01_2015-01-05_22_45_51	98% (2602289)	2% (58021)	75% (10366)	25% (3365)
KRYPT-XSYN-S-02_2015-01-06_23_51_17	98% (3033486)	2% (73789)	75% (10366)	25% (3365)
QUANT-SSYN-S-01	14% (122651)	86% (777644)	100% (13731)	0% (0)
QUANT-SSYN-S-02	28% (132545)	72% (335932)	100% (13731)	0% (0)
RAGE-ESSYN-S-02_2015-01-03_23_20_03	0% (0)	100% (463416)	100% (13731)	0% (0)
REST-SSYN-S-01_2014-12-22_15_57_50	71% (2336473)	29% (943109)	75% (10366)	25% (3365)
REST-SSYN-S-02_2014-12-23_23_21_20	71% (2659402)	29% (1104569)	75% (10366)	25% (3365)
REST-XSSYN-S-01_2014-12-22_16_09_59	72% (2510449)	28% (985731)	75% (10366)	25% (3365)
REST-XSSYN-S-02_2014-12-23_23_34_46	71% (2648000)	29% (1098449)	75% (10366)	25% (3365)
TITAN-ESSYN-S-01_2014-12-22_22_33_09	99% (429134)	1% (5551)	75% (10366)	25% (3365)
VDOS-ESSYN-S-01_2014-12-25_13_30_25	98% (5244277)	2% (87772)	75% (10366)	25% (3365)
VDOS-ESSYN-S-02_2014-12-25_17_30_40	99% (2822177)	1% (22766)	75% (10366)	25% (3365)
VDOS-TCPACK-S-01_2014-12-25_13_44_11	0% (0)	100% (2371551)	100% (13731)	0% (0)
VDOS-TCPACK-S-02_2014-12-25_17_46_09	0% (0)	100% (1962625)	100% (13731)	0% (0)
VDOS-TCPFIN-S-01_2014-12-25_13_58_30	100% (5254629)	0% (2487)	75% (10366)	25% (3365)
VDOS-TCPNO-S-01_2014-12-25_14_26_19	100% (2709138)	0% (6214)	75% (10366)	25% (3365)
VDOS-TCPRST-S-01_2014-12-25_14_44_50	100% (5413121)	0% (3027)	75% (10366)	25% (3365)
VDOS-TCPRST-S-02_2014-12-25_18_43_05	100% (5013179)	0% (2469)	75% (10366)	25% (3365)
VDOS-XSYN-S-01_2014-12-25_15_02_42	98% (4707311)	2% (105204)	75% (10366)	25% (3365)
VDOS-XSYN-S-02_2014-12-25_20_01_31	98% (1813114)	2% (31437)	75% (10366)	25% (3365)