LATENCY AND THROUGHPUT COMPARISON BETWEEN IPTABLES AND NFTABLES AT DIFFERENT FRAME AND RULE-SET SIZES

Bachelor Degree Project in Information Technology
G2E, 22.5hp
Spring term 2018

Tomas Jonsson, c15tomjo@student.his.se

Supervisor: Dennis Modig
Examiner: Jianguo Ding
Abstract

Firewalls are one of the most common security tools used in computer networks. Its purpose is to prevent unwanted traffic from coming in to or out of a computer network. In Linux, one of the most common server operating system kernels available, iptables has been the go-to firewall for nearly two decades but a proposed successor, nftables, is available. This project compared latency and throughput performance of both firewalls with ten different rule-set sizes and seven different frame sizes using both linear look-ups and indexed data structures. Latency was measured through the round-trip time of ICMP packets while throughput was measured by generating UDP traffic using iPerf3. The results showed that, when using linear look-ups, nftables performs worse than iptables when using small frame sizes and when using large rule-sets. If the frame size was fairly large and rule-set fairly small, nftables was often performed slightly better both in terms of latency and in terms of throughput. When using indexed data structures, performance of both firewalls was very similar regardless of frame size or rule-set size. Minor, but statistically significant, differences were found both in favour of and against nftables, depending on the exact parameters used.

Keywords: Firewall, Iptables, Nftables, Latency, Throughput, Performance

Abstrakt


Nyckelord: Brandvägg, Iptables, Nftables, Latens, Genomströmning, Prestanda
7.2.7. 1518 Bytes Frames
8. Conclusions
9. Discussion
10. Future Work

Appendix A. Validity Threats
Appendix B. iptables configuration
Appendix C. IPset configuration
Appendix D. nftables configuration
Appendix E. nftables set configuration
Appendix F. Test script
1. Introduction

Firewalls are one of the most common security devices used on computer networks (Richardson 2008). Its primary purpose is to filter incoming and/or outgoing traffic according to a set of rules defined in its configuration. Firewalls of different levels of complexity can filter traffic on different layers of the network stack.

One commonly used firewall is iptables which has been included in the Linux kernel for almost two decades. Iptables has been continuously optimized and extended over the years but has inherent problems that cannot be patched. A new firewall framework called nftables, aimed to solve the problems of iptables, has been developed and is considered stable enough to be used in production environments.

This experiment tested and compared the latency and throughput of both firewalls with ten different rule-set sizes and seven different frame sizes using both traditional linear look-ups and using indexed data structures. It was a quantitative experiment that aimed to provide an answer to whether there is a statistically significant performance difference between the two firewalls. The experiment largely followed guidelines provided by the Benchmarking Methodology Working Group (BMWG) of the Internet Engineering Task Force (IETF).

Latency was measured through the round-trip time of ICMP packets while throughput was measured and generated by iPerf3.

The results are presented in both graph and table formats along with p-values to determine the reliability of the results. Details on methodology and the implementation of said methodology can be found in sections 5 and 6.

A minor discussion about the results and other observations can be found at the end of the report along with suggestions for future work regarding iptables and nftables.
2. Background

The term firewall in relation to Information Technology (IT) was possibly first used in 1992 when Marcus J. Ranum published a paper named “A Network Firewall” (Pfleeger, Pfleeger & Margulies 2015). An official definition of the term does not exist, but the Merriam-Webster dictionary defines it as “computer hardware or software that prevents unauthorized access to private data (as on a company's local area network or intranet) by outside computer users (as of the Internet)” (Merriam-Webster Dictionary, 2018).

Firewalls are often the first line of defence against attacks and are considered a vital component for maintaining a secure environment (Vacca 2009). Firewalls are one of the most common security technologies adopted by organisations. Almost all organisation deploys firewalls according to the 2008 CSI Computer Crime & Security Survey (Richardson 2008). The number of firewalls in an organisation is heavily correlated with the number of employees. A report by Skybox Security in 2016 showed that 93% of organisations with 500-999 employees have 49 or fewer firewalls while 36% of larger organisations with more than 10000 employees have more than 500 firewalls. An additional 39% have 50-499 firewalls (Skybox Security 2016).

A firewall can be seen as an ordered list of rules with an associated action. An incoming packet will be compared to the list of rules in sequential order. If there is a match the associated action will be taken, and the packet will be processed accordingly. If the packet doesn’t match the rule, it will be compared to the next rule in the list. If the end of the list is reached without a match the default policy is applied. The default policy is usually to reject the packet but can be configured to accept packets (Vacca 2009).

Firewalls can be categorized into one of two broad categories, network or host-based. Host-based firewalls are software firewalls protecting individual computers regardless of the network they are connected to. Network firewalls watches traffic flowing from one network to another, for example from a Wide Area Network (WAN) like the Internet, to an internal corporate network. Many organizations use both types to meet security requirements (Northrup, n.d.). Figure 1 shows a typical use case for a network firewall.

Firewalls are sometimes referred to as either hardware or software firewalls. Hardware firewalls are devices with the specific task of acting as a firewall. These will run some kind of firewall software, but usually no other unnecessary software. Complex systems tend to be more vulnerable as the likelihood of bugs increases and tools present on the system can be used by an attacker to launch additional attacks from the firewall system. Therefore, firewall systems generally do not include software such as compilers, programming libraries or other tools (Pfleeger, Pfleeger & Margulies 2015).
There are several different types of firewalls of varying complexity and functionality. The oldest and simplest type of firewall is stateless packet filtering. It relies solely on IP (Internet Protocol) address and port numbers to determine whether to permit or deny a packet. This makes it very fast but also very inflexible. As the data portion of the packets aren't read no particularly advanced configurations can be made with stateless packet filtering. Rule sets can quickly become large which increases the risk for misconfiguration which is a security concern (Pfleeger, Pfleeger & Margulies 2015).

A stateful firewall tracks different states of packets. This allows firewall rules that allow network sessions which is very common when communicating over a network. Traffic usually goes both from client to server and from server to client. With stateless packet filtering this would require separate rules for each port used by the client (both for sending packets and receiving packets) which can be very difficult to know in advance. Stateful firewalls solves this problem by tracking the state of incoming packets and being able to determine whether packets belong to an ongoing session (Vacca 2009).

Even more advanced firewalls exist such as application proxies. An application proxy acts as an intermediate on the network and analyses all packets before resending or discarding them. The proxy analyses the content of the packet all the way to layer 7 (the application layer) of the OSI (Open Systems Interconnection) model. The proxy must be written for each supported application which means that the proxy may need to be updated when applications are updated. Other disadvantages include additional processing which means performance is decreased. The advantage is that security can be very good indeed as the exact needs for a specific application can be met (Henmi, Lucas, Singh & Cantrell 2006).

It is quite common for firewalls to perform other tasks such as Network Address Translation (NAT). NAT is a process in which the source or destination IP address is rewritten. This is usually performed because the number of public IPv4 addresses are limited and as such private IP ranges specified in RFC (Request For Comments) 1918 (Rekhter, Moskowitz, Karrenberg,
de Groot & Lear 1996) are used on internal networks. NAT is usually not seen as a security measure but can be used to hide the IP addresses of internal hosts from an attacker (Pfleeger, Pfleeger & Margulies 2015).

Other features sometimes provided by firewalls include service differentiation enabling Quality of Service (QoS) to prioritize certain types of traffic. Services such as Voice over IP are very dependent on low latency to operate properly (Sun, Mkwawa, Jammeh & Ifeachor 2013).

2.1. Iptables and nftables
Packet filtering by the Linux kernel has been possible since 1994 when ipfw was ported from BSD. In 1998 ipchains was released along with a heavily reworked Linux kernel version 2.2. A year later Linux kernel version 2.4 was released and along with it the user space tool “iptables”. Iptables has been the de facto standard tool ever since (netfilter.org n.d.).

Iptables is technically only a user space tool that is used to configure the kernel’s packet filtering table, but the name is often used to refer to several components of the firewall, including the kernel packet filter (Kenton 2015).

Iptables has been continuously developed over the years by adding modules for various features. Some of them are built-in, such as “conntrack”, “limit” and “iprange”. Others need to be installed separately.

Related tools include ip6tables (iptables for IPv6), arptables (packet filter rules for Address Resolution Protocol) and ebtables (filtering tool for a Linux-based bridging firewall).

Some frameworks can be used in conjunction with iptables to improve performance and capabilities. One such framework is IPset, which is configured with the “ipset” user space tool. IPset can use bitmaps or hashes to store IP addresses, ports, MAC addresses (or combinations of them) for fast look-ups. Unlike regular iptables rules, IPset sets are not traversed linearly and instead uses indexed data structures which can make look-ups very fast, even in very large rule-sets (Van Styn 2012).

Nftables is the proposed successor to iptables and was included in version 3.13 of the Linux Kernel which was released in January 2014. It is meant to replace iptables, ip6tables, arptables and ebtables with a new packet classification framework based on a network-specific virtual machine (VM) and a single user space tool named “nft”. The tool compiles the rule-set into bytecode and pushes it into the kernel and does the opposite when retrieving the rule-set. (netfilter.org n.d.).

Both iptables and nftables are structured in similar ways. They use tables as a container for a set of chains. Chains are containers for a set of rules which are traversed sequentially. Iptables comes preconfigured with five tables (raw, filter, nat, mangle and security) that are hard coded and cannot be removed. The filter and nat tables are the most commonly used tables and they each come preconfigured with three different chains. The filter table is the default table and consists of the INPUT, OUTPUT and FORWARD chains. Packets addressed to the local system is sent through the INPUT chain, packets originating from the local system is sent through the OUTPUT chain and incoming packets not addressed to the local system is sent through the FORWARD chain.

Nftables does not come preconfigured with any chains with specific semantics which is the case for iptables. Chains are created as needed and as such no unnecessary chains are registered. For example, it is unlikely that a host-based firewall needs a forwarding chain, but such a chain is always registered with iptables (Nftables 2016).
Iptables always records the number of times a certain rule has been matched. Nftables leaves these counters completely optional (Nftables 2016).

Nftables can specify several actions in one single rule, whereas iptables has been limited to just one. The workaround in iptables is to create custom chains to jump to but this makes the rule-set more complex (Nftables 2016).

When a change in the rule-set is made with iptables, the entire configuration is flushed and rewritten. Nftables solved this problem by representing the rule-set as a linked list enabling adding new rules without touching the existing ones (Nftables 2016).

Post Linux kernel version 4.1, nftables can concatenate multiple keys and combine them with maps and dictionaries. Nftables natively supports sets, which is an indexed data structure from which the contents can be searched very quickly (Nftables 2016).

With nftables, much of the intelligence and complexity is moved out of the kernel codebase and moved to the user space tool and the network-specific VM. This should make it easier to deliver new features without the need to update the kernel and only updating the user space tools (Nftables 2016).

2.2. Terminology
The OSI model is a conceptual model defined in ISO/IEC 8348:2002 (International Organization for Standardization 2002). The model defines seven layers and is designed to provide interoperability between diverse communications systems. Each layer encapsulates the layers above it, enabling a flexible, modular, model.

Ethernet is a family of technologies usually used to connect devices at the physical and data link layers of the OSI model. It has almost entirely replaced other technologies such as Token Ring in Local Area Networks. Common standards in the family include 10BASE-T, 100BASE-TX, and 1000BASE-T for speeds of 10Mbit/s, 100Mbit/s and 1Gbit/s respectively. Currently version 2 of Ethernet is used virtually everywhere. Unless otherwise specified, the use of the term Ethernet is assumed to refer to version 2.

A Network Interface Controller, sometimes referred to as a Network Interface Card, is a hardware component that allows connection to a computer network. They implement the necessary electronic circuitry required for communication on the data-link layer of the OSI model using a specific physical layer of the OSI model.

Media Access Control (MAC) addresses, sometimes referred to as hardware addresses, are unique identifiers given to all NICs and are, in most IEEE (Institute of Electrical and Electronics Engineers) network technologies, used as network addresses for communication on the data link layer.

IP is a communications protocol used extensively in networks around the world. It delivers packets from a source to a destination based on IP addresses. There are two versions of IP currently in use, version 4 and version 6. IPv4 was specified in RFC 791 and uses fixed length addresses of 32 bits in length (Postel 1981a). The total addressable space is thus \(2^{32}\) addresses. This has been shown to be rather limited as the number of devices connected to the Internet has grown and IPv6 was specified in RFC 2400: Internet Protocol, Version 6 (IPv6) Specification in 1998. IPv6 expands the address space to \(2^{128}\) addresses by using addresses of 128 bits in length (Deering & Hinden 1998). IPv4 and IPv6 are not compatible with each other. This project only studies IPv4 in all tests. All mentions of IP should be considered IPv4 unless otherwise specified.
User Datagram Protocol (UDP) is a simple, connectionless, protocol at the transport layer (layer 4) of the OSI model used for communication with a minimum of protocol mechanism. It assumes IP as the underlying protocol and allows application programs to send messages to one another over a packet-switched network. It does not guarantee delivery or duplicate protection, making it a best-effort protocol. UDP is defined in RFC 768 (Postel 1980).

Transmission Control Protocol (TCP) is a connection-oriented end-to-end reliable protocol at the transport layer of the OSI model designed to provide reliable inter-process communication between pairs of processes on distinct but interconnected hosts on a network. It uses sequence numbers and acknowledgements as means of ensuring a reliable transfer of information by retransmitting missing, or otherwise unusable, packets. TCP is defined in RFC 793 (Postel 1981c).

Internet Control Message Protocol (ICMP) is a support protocol for IP and its messages usually contain operational information about sent datagrams. For example, if a packet’s time to live (TTL) field reaches zero, a message is sent to the originating address with information regarding the packet’s fate. The ICMP Echo Request and ICMP Echo Reply messages are commonly used by user applications to verify connectivity between hosts. ICMP is defined in RFC 792 (Postel 1981b).

Address Resolution Protocol (ARP) is a protocol to discover a link-layer address, for example a MAC address, associated with a given network layer address, for example an IPv4 address. On Ethernet networks ARP usually uses both broadcast packets and unicasts to associate a network address, usually an IP address, with a MAC address. ARP is defined in RFC 826 (Plummer 1982).

Throughput is defined in RFC 1242 (Bradner 1991) as “The maximum rate at which none of the offered frames are dropped by the device”. Throughput is usually measured in bits per second.

Latency in a store and forward device is defined by Bradner (1991) as “The time interval starting when the last bit of the input frame reaches the input port and ending when the first bit of the output frame is seen on the output port”. This is however very difficult to measure without specialized equipment without affecting the results significantly. Therefore, the definition used in this report is time elapsed from sending a packet until a response is received, such as an ICMP echo request and echo reply. As per RFC 3511 (Hickman et al. 2003), latency can only be measured at throughput levels, that is, at the highest offered load with zero packet loss. At loads less than throughput levels, it is called delay instead of latency.

Store and forward is a technique where incoming packets to an intermediate device are verified before being forwarded. This ensures that no runt frames are forwarded unto the network or CRC (Cyclic Redundancy Check) doesn’t fail (D-Link n.d.).

Runt frames are frames which do not meet the minimum frame size for the given protocol and medium. For IP over Ethernet, the minimum length of the data field of an Ethernet frame is specified by Hornig (1984) in RFC 894 to 46 bytes. Adding the data link header of 18 bytes brings the minimum frame size up to 64 bytes.

Frame size is the total size of the link layer frame, including header, FCS (Frame Check Sequence) and other information such as IEEE 802.1Q header. The required information depends on the media in use as mentioned by Bradner & McQuaid (1999) in RFC 2544. The contents of the Protocol Data Unit (PDU) of the frame is often referred to as a packet.
Maximum Transmission Unit (MTU) is the size of the largest PDU that can be transferred in a single network layer transaction. The standard MTU for Ethernet networks is defined in RFC 791 (Postel 1981a) and is set to 1500 bytes. If a large PDU cannot be sent over a network link it will, unless otherwise specified, be split in several distinct parts in a process called fragmentation. ICMP messages can be used to find the MTU of a path to the intended destination. This process is called Path MTU Discovery and is defined in RFC 1191 (Mogul & Deering 1990) for IPv4. Path MTU Discovery is performed when a client wants to perform it and involves sending increasingly large packets with the “Don’t Fragment” bit set. This is defined in RFC 1191:

When a router is unable to forward a datagram because it exceeds the MTU of the next-hop network and its Don't Fragment bit is set, the router is required to return an ICMP Destination Unreachable message to the source of the datagram, with the Code indicating "fragmentation needed and DF set”.

(Mogul & Deering 1990)

Firewalls come in several different architectures. The one used in this project is the “dual-homed firewall” architecture, which is a design where the firewall has two Network Interface Cards (NIC). One NIC is connected to an untrusted network, such as the Internet, and the other to a trusted network, such as a private Local Area Network (LAN) (Zwicky, Cooper & Chapman 2000).

Wide Area Networks (WAN) are communications networks that span a large geographical area, such as cities, countries or the entire world. They can be private networks connecting branch offices or they can be made public connecting many smaller networks, for example LANs, together. The largest WAN in the world is the Internet (Lifewire 2017a).

Local Area Networks (LAN) are networks spanning a small geographic area, for example a single building and are usually controlled by a single entity. LANs allow sharing of resources to other computers on the network, or other networks through, for example, WANs. Many offices, schools and homes have a LAN these days (Lifewire 2017b).

Bytecode are instructions designed to be executed by a virtual machine. The bytecode is interpreted by software and turned into machine code that the physical machine running the virtual machine can execute (Rouse 2005).
3. Research Aim

The aim of the experiment was to compare the latency and throughput, as defined in the background section, of iptables and nftables as the size of the rule-set increased, using both linear look-ups and indexed data structures using several different frame sizes.

3.1. Motivation

There has been very little, if any, work done regarding nftables, making it an interesting area to study. Despite being considered stable for a number of years, adoption rate of nftables has been very slow. It is unknown why the adoption rate has been slow, but experiments regarding performance metrics may shed light on the reasons why or provide an incentive for organizations to consider switching to nftables.

Latency was chosen as the target metric of the experiment as it is an immediately noticeable characteristic of network communications, regardless of application, and a significant factor in providing a positive user experience in a large number of areas.

Throughput was, in part, chosen because the test parameters suggested by RFC 3511 (Hickman et al. 2003) are identical to the ones suggested for latency in the same document. Therefore, they could be performed at the same time which was desirable when time was a constraint. The other part is that Internet traffic has grown, and is expected to grow, significantly in the last few years. Cisco expects that by 2021, the annual global Internet traffic will reach 3.3 ZB (Zettabytes) which is a threefold increase over 2016 numbers, Consumer Video-on-Demand (VoD) traffic will double, Internet gaming traffic will grow nearly tenfold, and IP Video will account for 82% of all Internet traffic (Cisco 2017).

An experiment performed by Google in 2009 showed that an additional delay of 100 ms resulted in -0.20% daily searches per user. A 400 ms delay resulted in a -0.59% impact on daily searches per user (Brutlag 2009).

Another experiment by Google showed that latency is more important than people themselves realise. When asked, people would rather have 30 search results than 10, but results showed that people searched 20% less overall when showing 30 results. The reason, it was concluded, was that 10 results were displayed faster than 30 results. Similarly, a 30% reduction in page size on Google Maps resulted in about 30% more map requests (Shankland 2008).

Cedexis, a software-defined application delivery platform, released a whitepaper in 2015 (Cedexis 2015) where they state:

- A 50 ms increase in latency results in gamers being twice as likely to abandon the game in question.
- A one-second delay in website performance can mean 7% fewer customers and the associated revenue.
- 81% of users will abandon a video if it does not start immediately.

Akamai, one of the world’s largest Content Delivery Networks (CDN), states that poor website performance has a significant impact on the revenue generated. 51% of shoppers surveyed stated that site performance had an influence on their buying behaviour. 28% of users do not return to a web site if it doesn’t perform sufficiently well, and a further 6% do not go to the affiliated retail store (Akamai 2002).

In High-Frequency Trading (HFT) computers use complex algorithms to analyse markets and perform transactions at very high speeds, often making decisions in milliseconds (Investopedia...
n.d.). In order to perform transactions as fast as possible HFT firms usually locate their servers at the same premises where an exchange’s servers are, as distance is typically a large factor in latency. To ensure all HFT firms at a given location compete on the same conditions, cable lengths are identical for all co-located clients within the exchange’s premises (Picardo 2018). Any advantage in latencies therefore must be achieved through other means.

3.2. Research Questions
The following two questions are addressed in this project:

1) Are there statistically significant differences in latency and throughput performance between nftables and iptables at differently sized rule-sets?
2) Are there statistically significant differences in latency and throughput performance between nftables and iptables at different frame sizes?

3.3. Objectives
The research objectives to be performed in order to conclude the experiment were:

1. Designing a general experiment layout.
2. Implementing the experiment design using available, and appropriate, hardware and software.
3. Performing pilot tests to determine exact experiment parameters and software behaviour.
4. Performing the experiment.
5. Presenting the results of the experiment.
6. Analysing and drawing conclusions based on the results.

3.4. Hypothesis
The hypothesis was that nftables would perform better than iptables as it is significantly newer than iptables and netfilter, the group responsible for both iptables and nftables, states performance as a reason to switch to nftables (Nftables 2017).

The associated null hypothesis aimed to be rejected was that nftables would not perform better than iptables and any such differences were completely coincidental.
4. Related Work

Hoffman, Prabhakar & Strooper (2003) analysed iptables performance and focused on the performance and correctness of the rules (whether they are working as intended). The test was performed at 10, 100 and 1000 Mb/s links, measuring throughput and delay at different rule set and frame sizes. The experiment was performed using two computers with multiple NICs and configuring the routing table to use the appropriate interface with regards to the IP address. The performance tests were performed using specialized hardware in the form of a SmartBits 6000B load generator. The test showed no problems using 10 Mb/s links. On 100 Mb/s links the performance started suffering when the size of the rule set exceeded 100 rules. On 1000 Mb/s the delay was surprisingly large regardless of frame size, but throughput was reasonable except at small frame sizes.

Junjie & Wenhui (2013) compared the throughput, maximum number of concurrent connections and latency of iptables and a Cisco ASA 5505 hardware firewall using recommendations found in RFC 2544 (Bradner & McQuaid 1999). The results show that the Cisco ASA 5505 performed better as it pertained to throughput and latency, but iptables could handle a larger number of concurrent connections and handled burst traffic, possibly attributed to the fact that the iptables machine had superior hardware.

Šimon, Huraj & Čerňanský (2015) studied iptables and ip6tables performance under Distributed Denial of Service (DDoS) attacks using both IPv4 and IPv6. The experiment studied system load, memory usage, sent and received bytes and end-to-end latency. The results show that iptables and ip6tables are very effective at filtering unwanted traffic while simultaneously forwarding legitimate traffic.

Niemann, Pfingst & Göbel (2015) studied the effects on the throughput performance of iptables at varying number of concurrent connections, frame sizes and number of rules for several different transport layer protocols using iptables. They found that throughput decreased roughly linearly with the number of rules for both IPv4 and IPv6, though not at the same rate (0.05% and 0.03% per rule respectively).

Riordan & Storkenmaier (2012) analysed the effects of reducing latency in the Xetra (“Exchange Electronic Trading”) stock trading system from 50 ms to 10 ms. The definition of latency was the time it took for an investor to submit and receive feedback about an order. The results indicate that a reduction in latency makes it easier to manage adverse selection risk and an increase in liquidity, mostly in small- and medium-sized stocks.

A study by Dick, Wellnitz & Wold (2005) analysed the effects of network delay and jitter on player performance in popular competitive online multiplayer games. The study compared the players’ perception of their skill and the actual outcome at different latency and jitter levels. The results showed that the game in question and the algorithms used to manage the network game data had a significant effect on the outcome. In at least one game, Unreal Tournament 2004, latency was as important as player skill for the outcome of the match.
5. Experimental Design

The experiment was a controlled fixed design experiment, also known as a quantitative experiment (Wohlin et al. 2012). The experiment largely followed recommendations set by the BMWG of the IETF as specified in RFC 3511 (Hickman, Newman, Tadjudin & Martin 2003). As the firewall also forwarded data frames based on information in the network layer, it also fulfilled the definition of a router according to RFC 1242 (Bradner 1991). Therefore, recommendations from RFC 2544 (Bradner & McQuaid 1999) has also been taken into consideration as this RFC’s terminology is based on RFC 1242 (Bradner 1991).

Conversations with Dialect, a general-purpose hosting provider in Skövde, and an administrator at the University of Skövde helped in determining some of the rule-set sizes.

The general design was as follows:

- Measure latency, as defined in the background section of this document, by sending ICMP Echo Requests from one network to another and recording the elapsed time as the ICMP Echo Reply is returned.
- Measure throughput, as defined in RFC 1242 (Bradner 1991) by sending UDP/IP packets from one network to another and recording the rate of transferred data.
- Traffic flowed from a simulated unprotected network to a simulated protected network.
- A dual-homed firewall controlled access by packet inspection between the two networks.
- Default input and forwarding policies were to deny any packets.
- Non-experiment related traffic was to be avoided wherever possible.
- Rule-set sizes of 20, 40, 60, 80, 100, 1000, 2500, 5000, 7500 and 10000 rules, with the rules associated with the actual test traffic at the bottom of the list as required by RFC 3511 (Hickman et al. 2003).
- Frame and packet sizes, as recommended in RFC 2544 (Bradner & McQuaid 1999), can be found in table 1.
- Per requirement in RFC 3511 (Hickman et al. 2003), tests were to be performed at throughput levels, that is, at the highest offered load with zero packet loss, which was to be achieved by sending a unidirectional stream of UDP/IP packets from a client on the unprotected network to a recipient on the protected network.
- Every test was to last at least 120 seconds as required by RFC 2544 by Bradner & McQuaid (1999).
- Every test was to be repeated at least 20 times as required by RFC 2544 by Bradner & McQuaid (1999).

5.1. Experiment Variables

Wohlin et al. (2012) defines independent variables as “All variables in a process that are manipulated and controlled are called independent variables”. The independent variables in this experiment design were the size of the rule-set and the frame size.

Table 1. Frame and packet sizes used in the experiment.

<table>
<thead>
<tr>
<th>Frame Size</th>
<th>64 B</th>
<th>128 B</th>
<th>256 B</th>
<th>512 B</th>
<th>1024 B</th>
<th>1280 B</th>
<th>1518 B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet Size</td>
<td>46 B</td>
<td>110 B</td>
<td>238 B</td>
<td>494 B</td>
<td>1006 B</td>
<td>1262 B</td>
<td>1500 B</td>
</tr>
</tbody>
</table>
Wohlin et al. (2012) defines dependent variables as “Those variables that we want to study to see the effect of the changes in the independent variables are called dependent variables”. The dependent variables in this experiment were the latency, as defined in the background section, measured recording the round-trip time of ICMP packets and throughput of UDP traffic measured by iPerf3.

5.2. Limitations
There are many factors that govern firewall performance and there was not enough time to account for them all in this experiment. There are many limitations to this experiment, some of which are discussed below.

- The experiment was only performed using 1 Gbit/s NICs. There were no other options available at the time of testing, but a high performance corporate network is likely to use higher speed networks.
- Only iptables and nftables were tested. There are many other options available, for example “PF Tables” for BSD-based operating systems or hardware firewalls from Cisco, Juniper or other network equipment vendors.
- The experiment was only performed using four parallel threads (see the Pilot Test section).
- As only one firewall rule can be the last rule in a rule-set, throughput was measured at one rule less than the specified rule-set size in order to save time.
- The experiment was only performed using UDP, as specified by RFC 3511 (Hickman et al. 2003), and ICMP packets. No testing was done to assess the performance using TCP or any other transport layer protocol.
- No testing was performed to assess the performance of any application protocol behaviour. For example, RFC 3511 (Hickman et al. 2003) recommends testing maximum HTTP (Hypertext Transfer Protocol) transaction rate.
- Only latency, not delay, as specified in the background section, was tested.
- In addition to latency and throughput, RFC 3511 (Hickman et al. 2003) specifies several metrics relevant to firewall performance. These metrics such as concurrent connections, maximum TCP connection establishment rate and others were not tested.
- Relevant hardware metrics, such as CPU (Central Processing Unit) load, were not measured as it could have affected the results of the latency and throughput tests.
- Only the MTU of 1500 bytes specified in RFC 894 (Hornig 1984) was tested. Although it is rare to use higher MTU values over WANs, it is sometimes used on LANs.
- NAT was not used in the experiment although it is commonly implemented on private networks.
- Configurations of the firewalls were kept as similar as possible which meant features only available to one of the firewalls, such as nftables ability to specify several actions in a single rule, were not tested.
5.3. Threats to Validity

Wohlin et al. (2012) discusses four areas of threats to the validity of an experiment. The four areas can be mapped to different stages of an experiment, from theory to observation. It is very important to consider the various threats that exist in each area in order to not draw erroneous conclusions. The four areas are:

1. **Conclusion validity.** This validity is sometimes referred to as statistical conclusion validity and concerns issues that may lead to erroneous conclusions about relations between the treatment (i.e. changing the value of a variable) and the outcome. Sample size and type of statistical test are examples of issues that need to be taken into consideration.

2. **Internal Validity.** This validity concerns the relationship between cause and effect. An outcome has to be caused by the intended treatment and not be influenced by some other factor that is not controlled in the experiment.

3. **Construct Validity.** This validity concerns the extent to which the experiment setting reflects the area under study. The relationship between theory and observation is very important in order to draw valid conclusions.

4. **External Validity.** This validity concerns whether the causal effect can be generalized outside the scope of the study.

The importance of each individual area of threat depends on the experiment in question. In a quantitative experiment such as this one, conclusion validity is of high priority. A table of threats to validity can be found in Appendix A.

5.3.1. Ethical Considerations

These experiments were conducted entirely on computer hardware without human interaction of any kind. No ethical issues were identified and therefore no ethical issues were taken into consideration.
6. Experimental Process
The experimental setup consisted of five physical computers, one of which acted as a dual-homed firewall controlling traffic between a simulated WAN to a LAN. Two of the computers were used to generate and receive a unidirectional UDP/IP stream at throughput levels, while one computer sent ICMP Echo requests to a recipient. The physical topology can be found in figure 2.

In order to minimize unwanted and unnecessary traffic on the network, the hardware was physically separated from any other networks. DHCP was not used on any machine, nor was there a DHCP server on the network. ARP requests were avoided by manually setting static ARP mappings for all computers using the “arp” tool in ubuntu:

```
sudo arp -s <ip> <mac address>
```

MTU was left at the default value of 1500 bytes as this is the maximum MTU allowed for IP packets over Ethernet as specified in RFC 894 (Hornig 1984) and it is also the de facto standard MTU used on most Ethernet networks.

Private network layer addresses, as specified in RFC 1918 (Rekhter et al. 1996), were used on both sides of the dual-homed firewall even though one of the sides is meant to simulate a WAN. The logical topology can be seen in figure 3.

All computers were restarted after each test in order to provide, to the extent possible, equal testing conditions.

As iptables cannot disable counters, nftables was also configured to use counters to provide equal benchmarking conditions to the greatest extent possible.

![Figure 2. Physical topology used in the experiment.](image-url)
6.1. Hardware
The physical computers used in the experiment were all identically equipped with the hardware specified in table 2.

Only the dual-homed firewall used the Intel 82574L Gigabit Ethernet Controller.

All computers were connected to Netgear ProSafe GS108E switches using category 6 Unshielded Twisted Pair (UTP) cables terminated with 8P8C modular connectors. The Netgear ProSafe GS108E switches support a total bandwidth of 16 Gbit/s over its 8 ports (Netgear 2017). The specification for category 6 Ethernet cables can be found in TIA/EIA-568-B.2 Addendum 1 (Electrical Contractor 2003) and allows for speeds up to 10 Gbit/s Ethernet over cables up to 100 meters in length (Barnett, Groth & McBee 2004). The computers on each side of the firewall used cables of 50 cm in length, while the firewall used cables of 2 meters in length.

6.2. Software
The operating system used on all computers in the experiment was Ubuntu 17.10. Ubuntu is a very popular Linux distribution that has been used extensively by, amongst others, Google (ZDNet 2012). Ubuntu releases are published every six months (Ubuntu 2018) which means the kernel is kept up-to-date. Debian, the distribution from which Ubuntu was originally forked, recommends using Linux kernel 4.10 or greater for nftables (Debian 2018) which contributed to the choice of Ubuntu, as Ubuntu 17.10 uses Linux kernel 4.13.0-36. The operating system was installed with minimal additional software. The only additional software installed was “openssh-server” (used for administration) and “iptables-persistent” with the “ipset-persistent” extension to enable the saving of rule-sets between reboots.

Table 2. Hardware of the computers used in the experiment

<table>
<thead>
<tr>
<th>CPU</th>
<th>Intel Xeon E3-1226 v3 @ 3.3 GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM</td>
<td>20 GB 1600 MHz DDR3 ECC</td>
</tr>
<tr>
<td>Network Interface Cards</td>
<td>Intel Ethernet Controller I217-LM (rev 4)</td>
</tr>
<tr>
<td></td>
<td>Intel 82574L Gigabit Ethernet Controller</td>
</tr>
</tbody>
</table>
The unidirectional stream of UDP/IP traffic used to measure throughput was generated using iPerf 3.13. iPerf3 is a powerful cross-platform tool capable of generating TCP, SCTP or UDP packets over both IPv4 and IPv6. It is deployed using a client-server model where one computer only sends traffic and one only receives traffic. Older versions of iPerf can be deployed in a bidirectional mode where traffic is sent both ways, but as RFC 3511 (Hickman et al. 2003) specifies a unidirectional stream, the latest version was used. The software can be configured to use features such as bandwidth limitation, specifying packet size, maximum segment size, parallel threads and other parameters which made it an appropriate choice for this experiment.

The latency measurement tool of choice was ping. Ping is an immensely popular cross-platform utility that implements the ICMP Echo Request and ICMP Echo Reply specification and reports statistics regarding round-trip time, packet loss and other errors. It can, amongst other things, be configured to send ICMP Echo Requests for a certain amount of time, at a specific interval and using a specific payload size. The “-q” flag was used to prevent unnecessary output and the “-w” parameter was used to specify the two-minute duration of the tests.

Analysis of data during pilot tests was done using the Wireshark software. Wireshark is one of the most widely-used network protocol analysers in the world (Wireshark 2018). It can do both live capture and offline analysis of hundreds of different protocols in a graphical user interface (GUI).

The firewalls used were, as discussed in the background section, iptables and nftables. Both firewalls were tested using linear look-ups, and by using indexed data structures which were available natively in the case of nftables, while iptables used the IPset framework. Iptables version 1.6.1 was included with Ubuntu 17.10 while nftables had to be installed from the official Ubuntu repository. The latest version available at the time of the experiment was nftables 0.7. IPset version 6.30 was installed and configured using a “hash:ip,port” type. Configurations of the firewalls can be found in Appendix B through E.

A script was used to set the static ARP mappings, start iPerf3 on both machines, start the measurement and record the results to a file. The script can be found in Appendix F.

6.3. Pilot Tests
To verify that the networks stacks, switches, and cables are working as intended, some simple tests using iPerf3 were performed.

The software was set up to send network traffic as quickly as possible from a computer on the unprotected network to a computer on the protected network. The tests were performed using both TCP and UDP without bandwidth limitations. Default settings were used in terms of packet sizes which means very large packets of 128 kB and 8 kB were used for TCP and UDP respectively. Tests were performed using a varying number of parallel threads. The default setting in iPerf3 is to use one thread, but it can be changed using the “-P” parameter. Using more than 4 parallel threads had a tendency of producing packet loss which is not allowed in latency testing as specified in RFC 3511 (Hickman et al. 2003). Therefore, four parallel threads were used in the experiment.

iperf3 -t 60 -P 4 -c 172.16.2.3
iperf3 -u -b 0 -t 60 -P 4 -i 0 -f k -c 172.16.2.3
Ten tests, each lasting one minute, were performed. Figure 4 and table 3 shows the result of the test. The results confirm that the hardware and network stacks are working as intended.

To verify the implementation of the ping tool provided with Ubuntu 17.10 some analysis in Wireshark was performed. The ping manual page for Linux (Die.net n.d.) states that the “-s” parameter specifies the ICMP payload size excluding the 8 bytes of ICMP header data. Analysis in Wireshark shows this is correct and that no other headers, like Ethernet and IP, are included. Payload size is consequently calculated by subtracting Ethernet header size, IP header size and ICMP header size from the intended frame size.

Using the ping tool and specifying payload size, the Maximum Transmission Unit (MTU) of the network path to the destination host can be determined. This can be performed by sending ICMP Echo Requests with increasingly large payload with the “Don’t Fragment” bit set. Using the ping tool, setting the “Don’t Fragment” bit is achieved by adding the “-M do” parameter to the command. A as previously discussed, the standard MTU for an Ethernet network is 1500 bytes, and the “-s” parameter does not include the size of header data from neither the ICMP (8 bytes) nor IP (20 bytes in this experiment) headers. Thus, the payload size for an ICMP Echo Request can be no larger than 1472 bytes in order to not exceed a total of 1500 bytes excluding Ethernet header and trailer.

```
root@HostA:~# ping -s 1472 -M do 172.16.2.2
PING 172.16.2.2 (172.16.2.2) 1472(1500) bytes of data.
1480 bytes from 172.16.2.2: icmp_seq=1 ttl=64 time=0.550 ms
```

<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
<th>Std. dev.</th>
<th>Media utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>935 Mbit/s</td>
<td>935.9 Mbit/s</td>
<td>936 Mbit/s</td>
<td>0.3 Mbit/s</td>
<td>100%</td>
</tr>
<tr>
<td>UDP</td>
<td>949 Mbit/s</td>
<td>949.7 Mbit/s</td>
<td>951 Mbit/s</td>
<td>0.8 Mbit/s</td>
<td>100%</td>
</tr>
</tbody>
</table>
root@HostA:~# ping -s 1473 -M do 172.16.2.2
PING 172.16.2.2 (172.16.2.2) 1473(1501) bytes of data.
ping: local error: Message too long, mtu=1500

This verified the implementation of ping as described in the ping manual page for Linux (Die.net n.d.) and the MTU to be 1500 bytes on the network path to the destination.

IPerf3 traffic was analysed in Wireshark to verify the implementation of the “-l” parameter. The documentation described the parameter as “The length of buffers to read or write. iPerf works by writing an array of len bytes a number of times” but it did not state whether that included any header information of any kind. Analysis showed that the implementation, like in the case with ping, did not account for any information in any headers. As the UDP header is 8 bytes long, just like an ICMP header, the length specified in iPerf3 was the same as the one specified with the ping command; by subtracting the Ethernet header size, IP header size and UDP header size from the intended frame size.

RFC 2544 (Bradner & McQuaid 1999) requires benchmarking of interconnect devices to be tested for at least 2 minutes and to be repeated at least 20 times. A pilot test to determine whether 2 minutes is appropriate was conducted. Five tests were performed for each duration with the default packet length (102 bytes frame size). No firewall was used.

iperf3 -u -b 0 -t 0 -f 4 -l 56 -f k -i 0 -c 172.16.2.3
ping -q -w <duration in seconds> 172.16.2.2

The results are shown in figure 5 and table 4.

Table 4. Pilot test latency results at different test lengths.

<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
<th>Std. dev.</th>
<th>Media Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 min</td>
<td>0.399 ms</td>
<td>0.473 ms</td>
<td>0.546 ms</td>
<td>0.033 ms</td>
<td>100%</td>
</tr>
<tr>
<td>3 min</td>
<td>0.394 ms</td>
<td>0.471 ms</td>
<td>0.552 ms</td>
<td>0.031 ms</td>
<td>100%</td>
</tr>
<tr>
<td>4 min</td>
<td>0.386 ms</td>
<td>0.472 ms</td>
<td>0.555 ms</td>
<td>0.039 ms</td>
<td>100%</td>
</tr>
<tr>
<td>5 min</td>
<td>0.390 ms</td>
<td>0.474 ms</td>
<td>0.566 ms</td>
<td>0.035 ms</td>
<td>100%</td>
</tr>
</tbody>
</table>
No differences of significance were found in the duration test. As time is a limiting factor in this experiment it was decided that performing more than 20 tests was not feasible given the number of combinations of rule-set sizes and frame sizes used in the experiment.

The finalized experiment parameters were as follows:

- 20 tests per combination of frame size, rule-set size and firewall.
- Each test had a duration of 120 seconds.
- The following iPerf3 command was used on the client:
  
  ```
  iperf3 -u -b 0 -t 0 -l <payload size> -f -k -P 4 -i 0 -c <ip>
  ```

- The following ping command was used:
  
  ```
  ping -q -w 120 -s <payload size> <ip>
  ```
7. Results

The results are presented in graph in table formats. The graphs only show the mean values of each while the tables contain exact values and the p-value calculated by performing a two-tailed independent-samples t-Test. A t-Test is a method to calculate whether the difference between two treatments is statistically reliable if the samples are assumed to be normally distributed. The p-value describes the likelihood that the results of the experiment does not represent the real-world differences. A lower p-value means it is less likely that the differences in the results are due to coincidences. In most cases, a p-value of 0.05 or lower is considered good enough to reject the null hypothesis.

The formula for the t-Test reads as follows:

$$t = \frac{(\bar{x}_1 - \bar{x}_2)}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}}$$

\(\bar{x}_1\) is the mean of the first set of values.

\(\bar{x}_2\) is the mean of the second set of values.

\(S_1\) is the standard deviation of the first set of values.

\(S_2\) is the standard deviation of the second set of values.

\(n_1\) is the total number of values in the first set.

\(n_2\) is the total number of values in the second set.

The resulting t-value has a corresponding p-value which can be found in look-up tables, or by using software to automate the process. The latter was used in this project.

7.1. Latency

The latency results were gathered using the “ping” tool and represent the round-trip time of ICMP packets. The results are shown in milliseconds with a precision of 3 decimal places in both graphs and tables.

To differentiate nftables using linear look-ups and nftables using indexed data structures they are referred to as “nftables” and “nftset” respectively.
7.1.1. 64 Bytes Frames

Figure 6 shows the results from the experiments using all four treatments. Note that ipset and nftset are overlapping in figure 6. Table 5 shows the results using linear look-ups at different rule-set sizes. Nftables performed significantly worse than iptables at all rule-set sizes using 64 bytes frame sizes. The p-values are all extremely small, ranging from 0.001 to 4.0E-58. The null hypothesis is not rejected at any rule-set size.

Table 6 shows the results using indexed data structures. Results are much closer, but IPset performed equally well or better at all rule-set sizes with very low p-values in all but two cases. The null hypothesis is not rejected at any rule-set size.

Table 5. Latency results in ms from 64 bytes frame size experiments using linear look-ups.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>iptables</th>
<th>nftables</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.455</td>
<td>0.694</td>
<td>0.001</td>
</tr>
<tr>
<td>40</td>
<td>0.457</td>
<td>1.341</td>
<td>5.1E-57</td>
</tr>
<tr>
<td>60</td>
<td>0.459</td>
<td>1.473</td>
<td>1.1E-57</td>
</tr>
<tr>
<td>80</td>
<td>0.564</td>
<td>1.623</td>
<td>4.6E-26</td>
</tr>
<tr>
<td>100</td>
<td>1.118</td>
<td>1.785</td>
<td>5.3E-15</td>
</tr>
<tr>
<td>1000</td>
<td>4.362</td>
<td>9.879</td>
<td>3.4E-51</td>
</tr>
<tr>
<td>2500</td>
<td>10.842</td>
<td>25.270</td>
<td>1.3E-52</td>
</tr>
<tr>
<td>5000</td>
<td>20.137</td>
<td>48.757</td>
<td>1.0E-51</td>
</tr>
<tr>
<td>7500</td>
<td>29.210</td>
<td>71.838</td>
<td>6.3E-50</td>
</tr>
<tr>
<td>10000</td>
<td>37.857</td>
<td>95.562</td>
<td>4.0E-58</td>
</tr>
</tbody>
</table>

Table 6. Latency results in ms from 64 bytes frame size experiments using indexed data structures.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>IPset</th>
<th>nftset</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.454</td>
<td>0.459</td>
<td>0.016</td>
</tr>
<tr>
<td>40</td>
<td>0.456</td>
<td>0.458</td>
<td>0.757</td>
</tr>
<tr>
<td>60</td>
<td>0.456</td>
<td>0.459</td>
<td>0.129</td>
</tr>
<tr>
<td>80</td>
<td>0.456</td>
<td>0.459</td>
<td>0.022</td>
</tr>
<tr>
<td>100</td>
<td>0.456</td>
<td>0.460</td>
<td>0.004</td>
</tr>
<tr>
<td>1000</td>
<td>0.456</td>
<td>0.460</td>
<td>1.0E-5</td>
</tr>
<tr>
<td>2500</td>
<td>0.455</td>
<td>0.458</td>
<td>0.007</td>
</tr>
<tr>
<td>5000</td>
<td>0.455</td>
<td>0.458</td>
<td>0.002</td>
</tr>
<tr>
<td>7500</td>
<td>0.455</td>
<td>0.460</td>
<td>0.003</td>
</tr>
<tr>
<td>10000</td>
<td>0.455</td>
<td>0.460</td>
<td>2.1E-5</td>
</tr>
</tbody>
</table>
7.1.2. 128 Bytes Frames

![Diagram of Latency 128 Bytes Frame Size]

Figure 7. Latency results from 128 bytes frame size experiments.

Figure 7 shows the results from the experiments using all four treatments. Note that ipset and nftset are overlapping in figure 7. Table 7 shows the results using linear look-ups at different rule-set sizes. Nftables performed significantly worse than iptables at all rule-set sizes using 128 bytes frame sizes. The p-values are extremely small in all cases. The null hypothesis is not rejected at any rule-set size.

Table 8 shows the results using indexed data structures. Nftset performed better than IPset at all rule-set sizes with low, or very low, p-values in all but one case. The null hypothesis is rejected at all rule-set sizes except 80 rules.

Table 7. Latency results in ms from 128 bytes frame size experiments using linear look-ups.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>iptables</td>
<td>0.481</td>
<td>0.471</td>
<td>0.470</td>
<td>0.694</td>
<td>1.081</td>
<td>3.490</td>
<td>8.655</td>
<td>15.844</td>
<td>22.941</td>
<td>30.307</td>
</tr>
<tr>
<td>nftables</td>
<td>0.857</td>
<td>1.171</td>
<td>1.278</td>
<td>1.426</td>
<td>1.541</td>
<td>7.915</td>
<td>20.274</td>
<td>38.538</td>
<td>56.705</td>
<td>75.347</td>
</tr>
<tr>
<td>p-value</td>
<td>3.0E-8</td>
<td>1.9E-51</td>
<td>1.1E-48</td>
<td>4.6E-14</td>
<td>6.0E-23</td>
<td>1.3E-44</td>
<td>1.4E-51</td>
<td>1.7E-53</td>
<td>5.9E-56</td>
<td>5.4E-47</td>
</tr>
</tbody>
</table>

Table 8. Latency results in ms from 128 bytes frame size experiments using indexed data structures.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPset</td>
<td>0.479</td>
<td>0.490</td>
<td>0.481</td>
<td>0.480</td>
<td>0.481</td>
<td>0.480</td>
<td>0.480</td>
<td>0.480</td>
<td>0.478</td>
<td>0.483</td>
</tr>
<tr>
<td>nftset</td>
<td>0.466</td>
<td>0.468</td>
<td>0.466</td>
<td>0.477</td>
<td>0.468</td>
<td>0.465</td>
<td>0.465</td>
<td>0.469</td>
<td>0.466</td>
<td>0.469</td>
</tr>
<tr>
<td>p-value</td>
<td>9.9E-9</td>
<td>0.022</td>
<td>2.4E-14</td>
<td>0.752</td>
<td>2.5E-10</td>
<td>5.4E-10</td>
<td>8.1E-8</td>
<td>2.5E-8</td>
<td>3.2E-7</td>
<td>6.2E-13</td>
</tr>
</tbody>
</table>
7.1.3. 256 Bytes Frames

Figure 8. Latency results from 256 bytes frame size experiments.

Figure 8 shows the results from the experiments using all four treatments. Note that ipset and nftset are overlapping in figure 8. Table 9 shows the results using linear look-ups at different rule-set sizes. Nftables performed better than iptables at rule-set sizes of 20 and 40 rules, but in all other cases iptables performed better. The p-values at both 20 and 40 rules are low enough to reject the null hypothesis. In all other cases the null hypothesis is not rejected.

Table 10 shows the results using indexed data structures. Nftables performed equally well or worse than iptables at all rule-set sizes except at 40 rules, though in all cases but one (20-rule rule-set) the p-values are so high that no conclusions can be drawn. The null hypothesis is not rejected at any rule-set size.

Table 9. Latency results in ms from 256 bytes frame size experiments using linear look-ups.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>iptables</td>
<td>0.480</td>
<td>0.481</td>
<td>0.477</td>
<td>0.477</td>
<td>0.478</td>
<td>3.116</td>
<td>7.456</td>
<td>13.801</td>
<td>19.983</td>
<td>25.796</td>
</tr>
<tr>
<td>nftables</td>
<td>0.476</td>
<td>0.477</td>
<td>0.481</td>
<td>0.478</td>
<td>1.212</td>
<td>6.870</td>
<td>17.275</td>
<td>32.723</td>
<td>48.789</td>
<td>64.115</td>
</tr>
<tr>
<td>p-value</td>
<td>0.001</td>
<td>5.6E-5</td>
<td>0.116</td>
<td>0.908</td>
<td>5.0E-12</td>
<td>2.8E-54</td>
<td>7.7E-32</td>
<td>2.2E-50</td>
<td>1.9E-50</td>
<td>2.5E-51</td>
</tr>
</tbody>
</table>

Table 10. Latency results in ms from 256 bytes frame size experiments using indexed data structures.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPset</td>
<td>0.481</td>
<td>0.484</td>
<td>0.481</td>
<td>0.480</td>
<td>0.481</td>
<td>0.480</td>
<td>0.483</td>
<td>0.481</td>
<td>0.481</td>
<td>0.479</td>
</tr>
<tr>
<td>nftset</td>
<td>0.483</td>
<td>0.481</td>
<td>0.481</td>
<td>0.480</td>
<td>0.481</td>
<td>0.480</td>
<td>0.483</td>
<td>0.481</td>
<td>0.481</td>
<td>0.481</td>
</tr>
<tr>
<td>p-value</td>
<td>0.018</td>
<td>0.064</td>
<td>0.622</td>
<td>0.239</td>
<td>1.000</td>
<td>0.820</td>
<td>0.299</td>
<td>0.806</td>
<td>0.479</td>
<td>0.286</td>
</tr>
</tbody>
</table>
7.1.4. 512 Bytes Frames

Figure 9. Latency results from 512 bytes frame size experiments.

Figure 9 shows the results from the experiments using all four treatments. Note that ipset and nftset are overlapping in figure 9. Table 11 shows the results using linear look-ups at different rule-set sizes. Nftables performed better using 20- and 40 rule rule-sets but worse in all other cases. The p-values are very low in all cases. The null hypothesis is rejected at 20- and 40-rule rule-sets but not rejected at any other rule-set size.

Table 12 shows the results using indexed data structures. Nftset performed better than IPset in some cases and worse in some, but in nearly all cases the p-values are high or very high. The exceptions are 2500- and 5000- and 7500-rule rule-sets where nfset performed better than IPset with a p-value low enough to conclude it is not a coincidence. The null hypothesis is rejected in the case of 2500- and 5000-rule rule-sets but not rejected in any other case.

Table 11. Latency results in ms from 512 bytes frame size experiments using linear look-ups.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>iptables</th>
<th>nftables</th>
<th>ipset</th>
<th>nftset</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.491</td>
<td>0.488</td>
<td>0.487</td>
<td>0.484</td>
<td>2.489</td>
</tr>
<tr>
<td>40</td>
<td>0.492</td>
<td>0.502</td>
<td>0.501</td>
<td>0.501</td>
<td>5.347</td>
</tr>
<tr>
<td>60</td>
<td>0.487</td>
<td>0.502</td>
<td></td>
<td>13.507</td>
<td>26.031</td>
</tr>
<tr>
<td>80</td>
<td>0.484</td>
<td>0.501</td>
<td></td>
<td></td>
<td>38.456</td>
</tr>
<tr>
<td>100</td>
<td>2.489</td>
<td>5.347</td>
<td></td>
<td></td>
<td>50.639</td>
</tr>
<tr>
<td>1000</td>
<td>5.870</td>
<td>13.507</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2500</td>
<td>10.933</td>
<td>26.031</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5000</td>
<td>15.496</td>
<td>38.456</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7500</td>
<td>20.460</td>
<td>50.639</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 12. Latency results in ms from 512 bytes frame size experiments using indexed data structures.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>IPset</th>
<th>nftset</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.492</td>
<td>0.490</td>
<td>0.080</td>
</tr>
<tr>
<td>40</td>
<td>0.491</td>
<td>0.491</td>
<td>0.713</td>
</tr>
<tr>
<td>60</td>
<td>0.492</td>
<td>0.492</td>
<td>0.139</td>
</tr>
<tr>
<td>80</td>
<td>0.491</td>
<td>0.492</td>
<td>0.701</td>
</tr>
<tr>
<td>100</td>
<td>0.491</td>
<td>0.490</td>
<td>0.398</td>
</tr>
<tr>
<td>1000</td>
<td>0.493</td>
<td>0.491</td>
<td>0.599</td>
</tr>
<tr>
<td>2500</td>
<td>0.492</td>
<td>0.489</td>
<td>0.050</td>
</tr>
<tr>
<td>5000</td>
<td>0.492</td>
<td>0.489</td>
<td>0.005</td>
</tr>
<tr>
<td>7500</td>
<td>0.492</td>
<td>0.489</td>
<td>0.026</td>
</tr>
<tr>
<td>10000</td>
<td>0.492</td>
<td>0.492</td>
<td>0.959</td>
</tr>
</tbody>
</table>
7.1.5. 1024 Bytes Frames

Figure 10. Latency results from 1280 bytes frame size experiments.

Figure 10 shows the results from the experiments using all four treatments. Note that ipset and nftset are overlapping in figure 10. Table 13 shows the results using linear look-ups at different rule-set sizes. Nftables performed better than iptables at rule-set sizes up to, and including, 100 rules, but performed worse at larger rule-sets. The p-values for rule-sets up to, and including, 100 rules are, with the exception of a 20-rule rule-set, low enough to conclude that nftables performs better than iptables and reject the null hypothesis. In all other cases the null hypothesis is not rejected.

Table 14 shows the results using indexed data structures. Nftset performed better than IPset in most cases but the p-values are too high to draw any conclusions based on the results. Only using a 40-rule rule-set is the null hypothesis rejected. In all other cases the null hypothesis is not rejected.

Table 13. Latency results in ms from 1024 bytes frame size experiments using linear look-ups.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>iptables</td>
<td>0.572</td>
<td>0.573</td>
<td>0.569</td>
<td>0.570</td>
<td>0.567</td>
<td>0.573</td>
<td>5.194</td>
<td>9.276</td>
<td>13.374</td>
<td>17.692</td>
</tr>
<tr>
<td>nftables</td>
<td>0.569</td>
<td>0.565</td>
<td>0.565</td>
<td>0.563</td>
<td>0.562</td>
<td>4.773</td>
<td>11.653</td>
<td>22.429</td>
<td>33.178</td>
<td>43.808</td>
</tr>
<tr>
<td>p-value</td>
<td>0.373</td>
<td>0.004</td>
<td>0.050</td>
<td>5.8E-5</td>
<td>0.004</td>
<td>9.1E-10</td>
<td>2.3E-28</td>
<td>3.6E-46</td>
<td>5.1E-53</td>
<td>5.9E-44</td>
</tr>
</tbody>
</table>

Table 14. Latency results in ms from 1024 bytes frame size experiments using indexed data structures.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPset</td>
<td>0.572</td>
<td>0.576</td>
<td>0.571</td>
<td>0.573</td>
<td>0.570</td>
<td>0.570</td>
<td>0.570</td>
<td>0.571</td>
<td>0.573</td>
<td>0.567</td>
</tr>
<tr>
<td>nftset</td>
<td>0.572</td>
<td>0.570</td>
<td>0.569</td>
<td>0.572</td>
<td>0.569</td>
<td>0.570</td>
<td>0.569</td>
<td>0.569</td>
<td>0.572</td>
<td>0.573</td>
</tr>
<tr>
<td>p-value</td>
<td>0.766</td>
<td>0.004</td>
<td>0.447</td>
<td>0.586</td>
<td>0.856</td>
<td>0.874</td>
<td>0.559</td>
<td>0.307</td>
<td>0.651</td>
<td>0.144</td>
</tr>
</tbody>
</table>
7.1.6. 1280 Bytes Frames

Figure 11. Latency results from 1280 bytes frame size experiments.

Figure 11 shows the results from the experiments using all four treatments. Note that ipset and nftset are overlapping in figure 11. Table 15 shows the results using linear look-ups at different rule-set sizes. Nftables performed better than iptables using rule-set sizes up to, and including, 100 rules but performed worse at rule-set sizes larger than 100. The p-values are extremely low at all rule-sizes. The null hypothesis is rejected at rule-set sizes up to, and including, 100 rules but not rejected at rule-set sizes larger than 100 rules.

Table 16 shows the results using indexed data structures. Nftset performed better than IPset at all rule-set sizes, but the p-values are not always low enough to draw any conclusions based on the result. The null hypothesis is rejected at 20-, 100-, 1000-, 2500-, 7500- and 10000-rule rule-sets, but not rejected at 40-, 60-, 80- and 5000-rule rule-sets.

Table 15. Latency results in ms from 1280 bytes frame size experiments using linear look-ups.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>iptables</th>
<th>nftables</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.764</td>
<td>0.752</td>
<td>1.1E-6</td>
</tr>
<tr>
<td>40</td>
<td>0.760</td>
<td>0.727</td>
<td>4.9E-9</td>
</tr>
<tr>
<td>60</td>
<td>0.758</td>
<td>0.719</td>
<td>3.5E-14</td>
</tr>
<tr>
<td>80</td>
<td>0.753</td>
<td>1.7E-14</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0.752</td>
<td>1.7E-16</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>0.792</td>
<td>1.3E-34</td>
<td></td>
</tr>
<tr>
<td>2500</td>
<td>5.057</td>
<td>5.0E-45</td>
<td></td>
</tr>
<tr>
<td>5000</td>
<td>9.167</td>
<td>2.4E-47</td>
<td></td>
</tr>
<tr>
<td>7500</td>
<td>13.324</td>
<td>2.5E-47</td>
<td></td>
</tr>
<tr>
<td>10000</td>
<td>17.595</td>
<td>2.1E-39</td>
<td></td>
</tr>
</tbody>
</table>

Table 16. Latency results in ms from 1280 bytes frame size experiments using indexed data structures.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>IPset</th>
<th>nftset</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.765</td>
<td>0.761</td>
<td>0.044</td>
</tr>
<tr>
<td>40</td>
<td>0.764</td>
<td>0.761</td>
<td>0.175</td>
</tr>
<tr>
<td>60</td>
<td>0.763</td>
<td>0.760</td>
<td>0.189</td>
</tr>
<tr>
<td>80</td>
<td>0.763</td>
<td>0.759</td>
<td>0.169</td>
</tr>
<tr>
<td>100</td>
<td>0.764</td>
<td>0.761</td>
<td>0.004</td>
</tr>
<tr>
<td>1000</td>
<td>0.766</td>
<td>0.759</td>
<td>0.010</td>
</tr>
<tr>
<td>2500</td>
<td>0.764</td>
<td>0.759</td>
<td>0.017</td>
</tr>
<tr>
<td>5000</td>
<td>0.761</td>
<td>0.759</td>
<td>0.274</td>
</tr>
<tr>
<td>7500</td>
<td>0.765</td>
<td>0.759</td>
<td>0.005</td>
</tr>
<tr>
<td>10000</td>
<td>0.764</td>
<td>0.757</td>
<td>0.001</td>
</tr>
</tbody>
</table>
7.1.7. 1518 Bytes Frames

Figure 12. Latency results from 1518 bytes frame size experiments.

Figure 12 shows the results from the experiments using all four treatments. Note that ipset and nftset are overlapping in figure 12. Table 17 shows the results using linear look-ups at different rule-set sizes. Nftables performed better than iptables using rule-set sizes up to, and including, 100 rules but performed worse at rule-set sizes larger than 100. The p-values are extremely low at all rule-sizes. The null hypothesis is rejected at rule-set sizes up to, and including, 100 rules but not rejected at rule-set sizes larger than 100 rules.

Table 18 shows the results using indexed data structures. Nftset performed better than IPset at all rule-set sizes except 10000 rules where the mean performance is equal. For rule-set sizes up to, and including, 1000 rules the p-values are low enough to reject the null hypothesis. At rule-set sizes larger than 1000, the null hypothesis is not rejected.

Table 17. Latency results in ms from 1518 bytes frame size experiments using linear look-ups.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>iptables</td>
<td>0.789</td>
<td>0.785</td>
<td>0.783</td>
<td>0.780</td>
<td>0.779</td>
<td>0.803</td>
<td>4.954</td>
<td>8.837</td>
<td>12.745</td>
<td>16.731</td>
</tr>
<tr>
<td>nftables</td>
<td>0.779</td>
<td>0.772</td>
<td>0.764</td>
<td>0.755</td>
<td>0.748</td>
<td>4.538</td>
<td>11.168</td>
<td>21.387</td>
<td>31.838</td>
<td>42.340</td>
</tr>
<tr>
<td>p-value</td>
<td>1.5E-7</td>
<td>7.2E-7</td>
<td>6.3E-17</td>
<td>7.2E-19</td>
<td>3.3E-17</td>
<td>1.3E-55</td>
<td>3.3E-45</td>
<td>2.6E-45</td>
<td>1.7E-46</td>
<td>6.7E-49</td>
</tr>
</tbody>
</table>

Table 18. Latency results in ms from 1518 bytes frame size experiments using indexed data structures.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPset</td>
<td>0.790</td>
<td>0.790</td>
<td>0.792</td>
<td>0.791</td>
<td>0.789</td>
<td>0.790</td>
<td>0.787</td>
<td>0.789</td>
<td>0.790</td>
<td>0.788</td>
</tr>
<tr>
<td>nftset</td>
<td>0.783</td>
<td>0.782</td>
<td>0.782</td>
<td>0.783</td>
<td>0.784</td>
<td>0.783</td>
<td>0.784</td>
<td>0.788</td>
<td>0.787</td>
<td>0.788</td>
</tr>
<tr>
<td>p-value</td>
<td>0.003</td>
<td>1.0E-5</td>
<td>2.6E-5</td>
<td>4.7E-4</td>
<td>0.015</td>
<td>5.8E-5</td>
<td>0.131</td>
<td>0.562</td>
<td>0.103</td>
<td>0.558</td>
</tr>
</tbody>
</table>
7.2. Throughput
As discussed previously, throughput was tested alongside latency in order to save time as tests on physical hardware is very time consuming since all machines need to be rebooted between each test for validity reasons. This means throughput results were collected at rule-set sizes of <specified rule-set size – 1>.

The throughput results were collected using iPerf3 which was configured to show throughput in kilobits/s for increased precision. The results in this section are presented using Megabits/s for readability.

To differentiate nftables using linear look-ups and nftables using indexed data structures they are referred to as “nftables” and “nftset” respectively.
7.2.1. 64 Bytes Frames

Figure 13. Throughput results from 64 bytes frame size experiments.

Figure 13 shows the results from the experiments using all four treatments. Note that ipset and nftset are overlapping in figure 13. Table 19 shows the results using linear look-ups at different rule-set sizes. Nftables performed worse than iptables at all rule-set sizes. The p-values are very low or extremely low in all cases. The null hypothesis is not rejected.

Table 20 shows the results using indexed data structures. Nftset performed better than IPset at all rule-set sizes. In all but one case (20-rule rule-set) the p-values are low enough to reject the null hypothesis.

Table 19. Throughput results in Mb/s from 64 bytes frame size experiments using linear look-ups.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>iptables</td>
<td>108.34</td>
<td>107.70</td>
<td>107.24</td>
<td>107.02</td>
<td>105.71</td>
<td>22.83</td>
<td>8.60</td>
<td>4.57</td>
<td>3.15</td>
<td>2.39</td>
</tr>
<tr>
<td>nftables</td>
<td>108.23</td>
<td>95.53</td>
<td>82.27</td>
<td>71.73</td>
<td>63.31</td>
<td>9.56</td>
<td>3.62</td>
<td>1.86</td>
<td>1.26</td>
<td>0.95</td>
</tr>
<tr>
<td>p-value</td>
<td>0.219</td>
<td>4.4E-42</td>
<td>9.2E-58</td>
<td>5.7E-55</td>
<td>1.5E-82</td>
<td>1.7E-91</td>
<td>2.0E-110</td>
<td>5.5E-98</td>
<td>3.6E-114</td>
<td>2.0E-113</td>
</tr>
</tbody>
</table>

Table 20. Throughput results in Mb/s from 64 bytes frame size experiments using indexed data structures.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPset</td>
<td>108.02</td>
<td>108.18</td>
<td>106.99</td>
<td>107.90</td>
<td>107.90</td>
<td>106.80</td>
<td>107.59</td>
<td>108.15</td>
<td>108.14</td>
<td>108.21</td>
</tr>
<tr>
<td>nftset</td>
<td>108.14</td>
<td>108.61</td>
<td>108.49</td>
<td>108.16</td>
<td>108.45</td>
<td>108.47</td>
<td>108.35</td>
<td>108.26</td>
<td>108.18</td>
<td>108.26</td>
</tr>
<tr>
<td>p-value</td>
<td>0.110</td>
<td>1.2E-6</td>
<td>9.1E-6</td>
<td>8.9E-11</td>
<td>1.2E-8</td>
<td>5.9E-3</td>
<td>2.9E-8</td>
<td>0.101</td>
<td>0.679</td>
<td>0.471</td>
</tr>
</tbody>
</table>
7.2.2. 128 Bytes Frames

Throughput results from 128 bytes frame size experiments.

Figure 14. Throughput results from 128 bytes frame size experiments.

Figure 14 shows the results from the experiments using all four treatments. Note that ipset and nftset are overlapping in figure 14. Table 21 shows the results using linear look-ups at different rule-set sizes. Nftables only performed better than iptables using a 20-rule rule-set. The p-value at 20 rules is low enough to reject the null hypothesis. In all other cases the null hypothesis is not rejected.

Table 22 shows the results using indexed data structures. Nftset performed better than IPset in some cases and performed worse in others. The null hypothesis is rejected at 40-, 80-, 1000- and 7500-rule rule-sets but not rejected at 20-, 60-, 100-, 2500-, 5000- and 10000-rule rule-sets.

### Table 21. Throughput results in Mb/s from 128 bytes frame size experiments using linear look-ups.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>iptables</td>
<td>481.59</td>
<td>488.01</td>
<td>488.52</td>
<td>487.09</td>
<td>470.25</td>
<td>103.60</td>
<td>39.16</td>
<td>20.82</td>
<td>14.35</td>
<td>10.89</td>
</tr>
<tr>
<td>nftables</td>
<td>485.72</td>
<td>419.22</td>
<td>365.70</td>
<td>317.26</td>
<td>281.56</td>
<td>42.27</td>
<td>16.48</td>
<td>8.47</td>
<td>5.74</td>
<td>4.31</td>
</tr>
<tr>
<td>p-value</td>
<td>0.002</td>
<td>2.2E-61</td>
<td>1.5E-74</td>
<td>2.2E-65</td>
<td>6.2E-97</td>
<td>2.4E-106</td>
<td>4.1E-116</td>
<td>8.9E-104</td>
<td>3.8E-114</td>
<td>7.9E-118</td>
</tr>
</tbody>
</table>

### Table 22. Throughput results in Mb/s from 128 bytes frame size experiments using indexed data structures.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPset</td>
<td>491.56</td>
<td>486.01</td>
<td>490.08</td>
<td>488.67</td>
<td>490.32</td>
<td>490.19</td>
<td>491.63</td>
<td>490.74</td>
<td>489.88</td>
<td>489.89</td>
</tr>
<tr>
<td>nftset</td>
<td>491.15</td>
<td>490.13</td>
<td>489.99</td>
<td>491.29</td>
<td>485.30</td>
<td>491.38</td>
<td>491.01</td>
<td>491.06</td>
<td>490.74</td>
<td>490.83</td>
</tr>
<tr>
<td>p-value</td>
<td>0.231</td>
<td>2.4E-6</td>
<td>0.631</td>
<td>7.1E-10</td>
<td>7.2E-5</td>
<td>0.003</td>
<td>0.146</td>
<td>0.304</td>
<td>0.037</td>
<td>0.054</td>
</tr>
</tbody>
</table>
7.2.3. 256 Bytes Frames

Throughput results from 256 bytes frame size experiments.

Figure 15. Throughput results from 256 bytes frame size experiments.

Figure 15 shows the results from the experiments using all four treatments. Note that ipset and nftset are overlapping in figure 15. Table 23 shows the results using linear look-ups at different rule-set sizes. Nftables performed worse than iptables at all rule-set sizes. As the p-values are low or extremely low in all cases, the null hypothesis is not rejected at any rule-set size.

Table 24 shows the results using indexed data structures. Performance was virtually identical across the board. The p-values are very low at all rule-set sizes except 60. The null hypothesis is therefore rejected at all but one rule-set size. Do note however that the difference is, at most, 0.02 Mbit/s.

Table 23. Throughput results in Mb/s from 256 bytes frame size experiments using linear look-ups.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>iptables</td>
<td>725.55</td>
<td>725.55</td>
<td>725.55</td>
<td>725.54</td>
<td>725.54</td>
<td>264.11</td>
<td>100.10</td>
<td>53.30</td>
<td>36.70</td>
<td>27.86</td>
</tr>
<tr>
<td>nftables</td>
<td>725.54</td>
<td>725.52</td>
<td>725.49</td>
<td>725.51</td>
<td>718.68</td>
<td>108.03</td>
<td>42.44</td>
<td>21.80</td>
<td>14.67</td>
<td>11.08</td>
</tr>
<tr>
<td>p-value</td>
<td>0.001</td>
<td>1.4E-12</td>
<td>3.3E-9</td>
<td>1.4E-13</td>
<td>3.5E-40</td>
<td>3.8E-99</td>
<td>7.4E-109</td>
<td>3.6E-108</td>
<td>3.9E-101</td>
<td>6.3E-97</td>
</tr>
</tbody>
</table>

Table 24. Throughput results in Mb/s from 256 bytes frame size experiments using indexed data structures.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPset</td>
<td>725.55</td>
<td>725.55</td>
<td>725.54</td>
<td>725.55</td>
<td>725.55</td>
<td>725.55</td>
<td>725.55</td>
<td>725.55</td>
<td>725.55</td>
<td>725.55</td>
</tr>
<tr>
<td>nftset</td>
<td>725.54</td>
<td>725.55</td>
<td>725.55</td>
<td>725.54</td>
<td>725.54</td>
<td>725.54</td>
<td>725.55</td>
<td>725.53</td>
<td>725.55</td>
<td>725.54</td>
</tr>
<tr>
<td>p-value</td>
<td>2.1E-5</td>
<td>0.001</td>
<td>5.2E-5</td>
<td>0.340</td>
<td>7.4E-5</td>
<td>0.030</td>
<td>2.0E-5</td>
<td>6.9E-5</td>
<td>1.5E-5</td>
<td>3.2E-5</td>
</tr>
</tbody>
</table>

Note: ipset and nftset are overlapping
7.2.4. 512 Bytes Frames

Figure 16. Throughput results from 512 bytes frame size experiments.

Figure 16 shows the results from the experiments using all four treatments. Note that ipset and nftset are overlapping in figure 16. Table 25 shows the results using linear look-ups at different rule-set sizes. Performance between nftables and iptables was identical up to, and including, 100-rule rule-sets but nftables performed worse at rule-sets larger than 100 rules. The null hypothesis is not rejected at any rule-set size.

Table 26 shows the results using indexed data structures. Performance between nftset and IPset was identical at all rule-set sizes. The null hypothesis is not rejected at any rule-set size.

Table 25. Throughput results in Mb/s from 512 bytes frame size experiments using linear look-ups.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>iptables</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
<td>593.66</td>
<td>224.31</td>
<td>118.91</td>
<td>81.78</td>
<td>62.02</td>
</tr>
<tr>
<td>nftables</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
<td>248.45</td>
<td>94.36</td>
<td>48.47</td>
<td>32.62</td>
<td>24.51</td>
</tr>
<tr>
<td>p-value</td>
<td>6.6E-4</td>
<td>0.145</td>
<td>0.074</td>
<td>0.366</td>
<td>0.003</td>
<td>6.9E-109</td>
<td>4.6E-116</td>
<td>1.9E-114</td>
<td>2.7E-126</td>
<td>3.7E-128</td>
</tr>
</tbody>
</table>

Table 26. Throughput results in Mb/s from 512 bytes frame size experiments using indexed data structures.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPset</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
</tr>
<tr>
<td>nftset</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
<td>843.92</td>
</tr>
<tr>
<td>p-value</td>
<td>0.292</td>
<td>0.114</td>
<td>0.601</td>
<td>0.147</td>
<td>0.757</td>
<td>3.0E-6</td>
<td>0.015</td>
<td>0.537</td>
<td>4.0E-5</td>
<td>0.006</td>
</tr>
</tbody>
</table>
7.2.5. 1024 Bytes Frames

Figure 17. Throughput results from 1024 bytes frame size experiments.

Figure 17 shows the results from the experiments using all four treatments. Note that ipset and nftset are overlapping in figure 17. Table 27 shows the results using linear look-ups at different rule-set sizes. Performance between nftables and iptables was identical up to, and including, 100-rule rule-sets but nftables performed worse at rule-sets larger than 100 rules. The null hypothesis is not rejected at any rule-set size.

Table 28 shows the results using indexed data structures. Performance between nftset and IPset was identical at all rule-set sizes. The null hypothesis is not rejected at any rule-set size.

Table 27. Throughput results in Mb/s from 1024 bytes frame size experiments using linear look-ups.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>Throughput (Mbit/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>iptables</td>
</tr>
<tr>
<td>20</td>
<td>907.50</td>
</tr>
<tr>
<td>40</td>
<td>907.50</td>
</tr>
<tr>
<td>60</td>
<td>907.50</td>
</tr>
<tr>
<td>80</td>
<td>907.50</td>
</tr>
<tr>
<td>100</td>
<td>907.47</td>
</tr>
<tr>
<td>1000</td>
<td>469.86</td>
</tr>
<tr>
<td>2500</td>
<td>249.38</td>
</tr>
<tr>
<td>5000</td>
<td>171.42</td>
</tr>
<tr>
<td>7500</td>
<td>68.57</td>
</tr>
<tr>
<td>10000</td>
<td>51.40</td>
</tr>
</tbody>
</table>

Table 28. Throughput results in Mb/s from 1024 bytes frame size experiments using indexed data structures.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>Throughput (Mbit/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IPset</td>
</tr>
<tr>
<td>20</td>
<td>907.50</td>
</tr>
<tr>
<td>40</td>
<td>907.50</td>
</tr>
<tr>
<td>60</td>
<td>907.50</td>
</tr>
<tr>
<td>80</td>
<td>907.50</td>
</tr>
<tr>
<td>100</td>
<td>907.50</td>
</tr>
<tr>
<td>1000</td>
<td>907.50</td>
</tr>
<tr>
<td>2500</td>
<td>907.50</td>
</tr>
<tr>
<td>5000</td>
<td>907.50</td>
</tr>
<tr>
<td>7500</td>
<td>907.50</td>
</tr>
<tr>
<td>10000</td>
<td>907.50</td>
</tr>
</tbody>
</table>

p-value column: 0.034, 0.072, 0.009, 0.109, 0.006, 3.0E-99, 3.2E-100, 1.2E-103, 6.3E-107, 2.9E-101
7.2.6. 1280 Bytes Frames

![Throughput 1280 Bytes Frames](image)

Note: ipset and nftset are overlapping

Figure 18. Throughput results from 1280 bytes frame size experiments.

Figure 18 shows the results from the experiments using all four treatments. Note that ipset and nftset are overlapping in figure 18. Table 29 shows the results using linear look-ups at different rule-set sizes. Performance between nftables and iptables was identical up to, and including, 100-rule rule-sets but nftables performed worse at rule-sets larger than 100 rules. The null hypothesis is not rejected at any rule-set size.

Table 30 shows the results using indexed data structures. Performance between nftset and IPset was identical at all rule-set sizes. The null hypothesis is not rejected at any rule-set size.

Table 29. Throughput results in Mb/s from 1280 bytes frame size experiments using linear look-ups.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>iptables</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
<td>593.07</td>
<td>314.87</td>
<td>216.49</td>
<td>164.22</td>
<td></td>
</tr>
<tr>
<td>nftables</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
<td>654.33</td>
<td>249.67</td>
<td>128.93</td>
<td>86.71</td>
<td>65.44</td>
<td></td>
</tr>
<tr>
<td>p-value</td>
<td>0.852</td>
<td>0.528</td>
<td>0.759</td>
<td>0.249</td>
<td>0.698</td>
<td>9.7E-91</td>
<td>4.7E-116</td>
<td>5.8E-110</td>
<td>3.4E-125</td>
<td>2.8E-121</td>
</tr>
</tbody>
</table>

Table 30. Throughput results in Mb/s from 1280 bytes frame size experiments using indexed data structures.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPset</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
</tr>
<tr>
<td>nftset</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
<td>920.60</td>
</tr>
<tr>
<td>p-value</td>
<td>1.000</td>
<td>0.123</td>
<td>0.249</td>
<td>0.839</td>
<td>0.221</td>
<td>1.7E-4</td>
<td>0.115</td>
<td>0.759</td>
<td>0.531</td>
<td>0.009</td>
</tr>
</tbody>
</table>
7.2.7. 1518 Bytes Frames

Figure 19. Throughput results from 1518 bytes frame size experiments.

Figure 19 shows the results from the experiments using all four treatments. Note that ipset and nftset are overlapping in figure 19. Table 31 shows the results using linear look-ups at different rule-set sizes. Performance between nftables and iptables was identical up to, and including, 100-rule rule-sets but nftables performed worse at rule-sets larger than 100 rules. The null hypothesis is not rejected at any rule-set size.

Table 30 shows the results using indexed data structures. Performance between nftset and IPset was identical at all rule-set sizes. The null hypothesis is not rejected at any rule-set size.

Table 31. Throughput results in Mb/s from 1518 bytes frame size experiments using linear look-ups.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>iptables</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
<td>707.47</td>
<td>375.28</td>
<td>258.12</td>
<td>195.75</td>
<td></td>
</tr>
<tr>
<td>nftables</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
<td>774.24</td>
<td>295.15</td>
<td>151.33</td>
<td>77.00</td>
<td></td>
</tr>
<tr>
<td>p-value</td>
<td>5.4E^{-5}</td>
<td>4.3E^{-4}</td>
<td>9.9E^{-8}</td>
<td>1.5E^{-6}</td>
<td>0.050</td>
<td>1.1E^{-93}</td>
<td>1.5E^{-101}</td>
<td>3.2E^{-102}</td>
<td>2.0E^{-102}</td>
<td>5.5E^{-115}</td>
</tr>
</tbody>
</table>

Table 32. Throughput results in Mb/s from 1518 bytes frame size experiments using indexed data structures.

<table>
<thead>
<tr>
<th>Rule-set Size</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>2500</th>
<th>5000</th>
<th>7500</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPset</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
</tr>
<tr>
<td>nftset</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
<td>927.67</td>
</tr>
<tr>
<td>p-value</td>
<td>0.007</td>
<td>0.208</td>
<td>5.1E^{-6}</td>
<td>3.7E^{-6}</td>
<td>3.4E^{-4}</td>
<td>1.4E^{-4}</td>
<td>0.002</td>
<td>0.206</td>
<td>0.814</td>
<td>0.005</td>
</tr>
</tbody>
</table>
8. Conclusions

The research aim (section 3) was to compare the latency and throughput between nftables and iptables at different rule-set sizes and frame sizes.

The research questions (section 3.2) specifically asked if there are statistically significant differences in latency and throughput performance between nftables and iptables at differently sized rule-sets and frames.

Significant differences can be found under certain circumstances while under other circumstances performance was found to be identical. In some cases, minor, but statistically significant, differences were found both in latency and throughput.

Performance using indexed data structures in the form of IPset and the set-structure in nftables was overall very close in all cases. At times there were minor, but statistically significant, differences both in favour of and against nftables, depending on the parameters used in the test in question. The differences were, however, very small in the grand scheme of things. For most practical applications, performance using indexed data structures can be considered identical in all cases.

Using linear look-ups the differences were more pronounced. For the most part, nftables performed worse than iptables and in many cases significantly so. Some general trends noticed during the experiment were:

- In terms of latency, nftables does not perform well using small frame sizes and falls behind iptables very quickly, sometimes as early as 20-rule rule-sets.
- In terms of latency, nftables does not perform well using large rule-sets. At what point nftables starts falling behind iptables depends on the frame size used, but once rule-set sizes reach 1000 rules nftables performs significantly worse regardless of frame size.
- In terms of throughput, nftables does not perform well using small frame sizes and falls behind iptables very quickly, sometimes as early as 40-rule rule-sets.
- In terms of throughput, nftables does not perform well using large rule-sets. At what point nftables starts falling behind iptables depends on the frame size used, but once rule-set sizes reach 1000 rules nftables performs significantly worse regardless of frame size.
- At larger frame sizes (1024 bytes and above) and smaller rule-sets (up to, and including, 100 rules) nftables performs well. In terms of latency nftables performs better than iptables by small, but sometimes statistically significant, margins. In terms of throughput the performance of nftables and iptables is virtually identical.
9. Discussion

The results were very surprising and interesting. The reasons for nftables quite odd behaviour under some circumstances are unknown. It is possible that CPU load was a factor at smaller frame sizes, but such data was not recorded as the act of recording the data could impact the results as more context switches would have been made. If this is the case, the issue is likely to be exacerbated by using faster NICs which are often used in enterprise firewalls.

If CPU-load was a significant factor in performance, it is possible that nftables performs well under non-throughput speeds. Non-throughput speeds are, perhaps, a more realistic real-world scenario.

A non-LTS (Long Term Support) release of Ubuntu was used in the experiment to ensure that a recent version of nftables was available in the repository. It is unlikely that this had any effect on performance, but bugs in certain releases of operating systems are possible. It is also possible that later versions of nftables, which is still under active development, will resolve some of the issues revealed in this experiment.

It was noted during the experiment, but not scientifically measured, that adding rules to nftables, both linear and set-elements, was considerably slower than adding rules to iptables or IPset. This was very surprising as atomic rule insertion is supposed to be one of the advantages of nftables. It is possible that a single rule insertion is faster, but adding a large number of rules through, for example, a script was slower. It is possible that the security of iptables (i.e. that it is doing what it is supposed to be doing) is affected during rule updates in ways that nftables is not. This is an experiment for another study. If a large number of rules are to be added to nftables and time is a concern, it is recommended to write a configuration file containing the rules and then loading the configuration file with the “nft” user space tool.

The minor, and sometimes odd, differences found under some circumstances could be attributed to the fact that round-trip time is not the most ideal way of measuring latency. There are several points in the system where latency could be affected, on both sides of the firewall. These differences should even themselves out if sufficiently long tests (both in duration and number of iterations) are performed. It is possible that tests longer than two minutes, or more than 20 iterations, could result in more accurate results.
10. Future Work

Hickman et al. (2003) has proposed benchmarking methodology for a number of different aspects of a firewall not tested in this experiment. TCP-related performance in various forms, for example concurrent TCP connection capacity, maximum TCP connection establishment rate, or maximum TCP connection tear down rate.

Distributed Denial of Service (DDoS) attacks have increased over the past couple of years (CDNetworks 2017), providing a very good reason to test how nftables and iptables handle DDoS attacks.

Applications layer protocols, such as maximum HTTP transaction rate or HTTP transfer rate are other areas that have very realistic real-world applications.

Traffic that has been fragmented prior to reaching the firewall may need to be reassembled before making a forwarding decision which may have a performance impact for the firewalls.

Enterprise hardware are usually equipped with NICs capable of higher speeds than the 1 Gbit/s used in this experiment. Testing performance at higher speeds would be very interesting as they put greater load on the CPU.

Both firewall and clients can be tested in virtualized forms rather than the physical ones used in this experiment.

In addition to latency, testing delay and throughput at various levels of media utilization would provide more accurate real-world data as no firewall operates at throughput levels at all times.

Testing performance using a larger number of concurrent connections would provide more realistic data, perhaps in conjunction with studying frame loss under those circumstances.

NAT is commonly used on many networks and the translation may affect performance of firewalls differently. This could be tested with only minor changes to the configuration in this experiment.

As nftables performs well using larger frame sizes, testing performance using jumbo frames could be of interest as jumbo frames are sometimes used on LANs.

Memory footprints, especially when using indexed data structures, could be of great interest for those using very large rule-sets.

The performance impact, if any, of using rule counters, which are not mandatory in nftables, could be studied in detail.
References


http://services.google.com/fh/files/blogs/google_delayexp.pdf [2018-03-08]


Cedexis (2015). *Online Gaming: Go Fast or Go Home* [whitepaper].


D-Link. (n.d.). *What is the difference between Store-and-Forward switching and Cut-Through switching?*.


### Appendix A. Validity Threats

<table>
<thead>
<tr>
<th>Threat</th>
<th>Description</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low statistical power</td>
<td>Insufficient number of tests, or duration of the tests, may not provide enough data to draw valid conclusions.</td>
<td>The duration of each test was four times longer than specified by RFC 3511 (Hickman et al. 2003) and performed 20 times in accordance with RFC 2544 (Bradner &amp; McQuaid 1999).</td>
</tr>
<tr>
<td>Fishing for results</td>
<td>“Looking” for a specific outcome because it corresponds with preconceived notions of what the outcome should be can affect the conclusions drawn.</td>
<td>No results were excluded regardless of test or how strange the result may have looked.</td>
</tr>
<tr>
<td>Reliability of measures</td>
<td>Badly designed instruments used in the execution of the experiment may influence the results.</td>
<td>The tools used in the experiment are widely used and their implementations were verified with Wireshark in pilot tests.</td>
</tr>
<tr>
<td>Reliability of treatment</td>
<td>Inconsistent implementation of the rule-sets may affect the conclusions drawn.</td>
<td>Both firewalls were configured in ways that should be equal in as many ways as possible, for example including counters in nftables. See Appendix B-E for configuration details.</td>
</tr>
<tr>
<td>implementation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random irrelevancies in</td>
<td>Elements outside the experimental setting may sometimes influence the results and consequently the conclusions.</td>
<td>The networks were isolated from any other networks in order to prevent unwanted traffic that may have influenced the results. All computers had manually set IP addresses to eliminate DHCP requests and static ARP mappings to eliminate ARP requests.</td>
</tr>
<tr>
<td>experimental setting</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random heterogeneity of</td>
<td>If the computers or setting used in the experiment were sufficiently different the results may be due to individual differences rather than due to the treatment.</td>
<td>All computers use identical hardware and all computers use the same operating system. All computers were restarted after each test to ensure identical circumstances for all tests.</td>
</tr>
<tr>
<td>subjects</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random heterogeneity of</td>
<td>Different NICs on the firewall computer may influence the results.</td>
<td>This threat is accepted. Performance was tested in pilot tests and both NICs complies with Ethernet specifications for gigabit speeds.</td>
</tr>
<tr>
<td>subjects</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interaction of setting and treatment</td>
<td>The parameter chosen for the experiment, such as rule-set sizes and concurrent connections, may not be representative of real-world applications which can impact the usefulness of the results.</td>
<td>Rule-set sizes were, in part, decided upon after conversations with administrators working in the field. Unrealistic number of concurrent connections is accepted as a threat as the experiment was deemed difficult to perform given the requirements in RFC 3511 (Hickman et al. 2003).</td>
</tr>
</tbody>
</table>
Appendix B. iptables configuration

*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [0:0]
:forward-from-wan - [0:0]
:forward-from-lan - [0:0]
-A FORWARD -i enp2s0 -j forward-from-wan
-A FORWARD -i lo -j forward-from-lan
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp --dport 22 -j ACCEPT
-A forward-from-wan -p tcp --dport 22 -j ACCEPT
-A forward-from-lan -j ACCEPT
-A forward-from-wan -p tcp --dport 40000 -d 172.16.2.3 -j ACCEPT
-A forward-from-wan -p tcp --dport 40001 -d 172.16.2.3 -j ACCEPT
- A forward-from-wan -p tcp --dport <40000 + rule-set size - 4> -d 172.16.2.3 -j ACCEPT
- A forward-from-wan -p tcp --dport 5201 -d 172.16.2.3 -j ACCEPT
- A forward-from-wan -p icmp --icmp-type 8 -d 172.16.2.2 -j ACCEPT

-A forward-from-wan -p tcp --dport <40000 + rule-set size - 4> -d 172.16.2.3 -j ACCEPT
- A forward-from-wan -p tcp --dport 5201 -d 172.16.2.3 -j ACCEPT
- A forward-from-wan -p icmp --icmp-type 8 -d 172.16.2.2 -j ACCEPT
Appendix C. IPset configuration

**Iptables**

*filter

:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [0:0]
:forward-from-wan - [0:0]
:forward-from-lan - [0:0]
-A FORWARD -i eno1 -j forward-from-wan
-A FORWARD -i enp2s0 -j forward-from-lan
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp --dport 22 -j ACCEPT
-A forward-from-wan -p tcp --dport 22 -j ACCEPT
-A forward-from-lan -j ACCEPT
-A forward-from-wan -m set --match-set forward-from-wan-set dst,dst -j ACCEPT
-A forward-from-wan -p udp --dport 5201 -d 172.16.2.3 -j ACCEPT
-A forward-from-wan -p icmp --icmp-type 8 -d 172.16.2.2 -j ACCEPT

**IPset (hash:ip,port)**

add forward-from-wan-set 172.16.2.3,tcp:40000
add forward-from-wan-set 172.16.2.3,tcp:40001
.
.
.
add forward-from-wan-set 172.16.2.3,tcp: <40000 + rule-set size - 5>
add forward-from-wan-set 172.16.2.3,tcp:5201
Appendix D. nftables configuration

table ip filter {
  chain input {
    type filter hook input priority 0; policy drop;
    counter
    iif "lo" counter accept
    tcp dport ssh counter accept
    counter packets 0 bytes 0
  }
  chain forward {
    type filter hook forward priority 0; policy drop;
    counter
    iifname "enol" jump forward-from-wan
    iifname "enp2s0" jump forward-from-lan
  }
  chain output {
    type filter hook output priority 0; policy accept;
    counter
  }
  chain forward-from-wan {
    counter
    tcp dport ssh counter accept
    ip daddr 172.16.2.3 tcp dport 40000 counter accept
    ip daddr 172.16.2.3 tcp dport 40001 counter accept
    ... 
    ip daddr 172.16.2.3 tcp dport <40000 + rule-set size - 5> counter accept
    ip daddr 172.16.2.3 tcp dport 5201 counter accept
    ip daddr 172.16.2.3 udp dport 5201 counter accept
    ip daddr 172.16.2.2 icmp type echo-request counter accept
  }
  chain forward-from-lan {
    counter
    accept
  }
}
Appendix E  nftables set configuration

table ip filter {
  set forward-from-wan-set {
    type ipv4_addr . inet_service
    elements = { 172.16.2.3 . 40000, 172.16.2.3 . 40001, (…), 172.16.2.3 . <40000 + rule-
    set size - 5>, 172.16.2.3 . 5201 }
  }

  chain input {
    type filter hook input priority 0; policy drop; counter
    iif "lo" counter accept
tcp dport ssh counter accept
    counter
  }

  chain forward {
    type filter hook forward priority 0; policy drop; counter
    iifname "en01" jump forward-from-wan
    iifname "enp2s0" jump forward-from-lan
  }

  chain output {
    type filter hook output priority 0; policy accept; counter
  }

  chain forward-from-wan {
    counter
tcp dport ssh counter accept
    ip daddr . tcp dport @forward-from-wan-set counter accept
    ip daddr 172.16.2.2 udp dport 5201 counter accept
    ip daddr 172.16.2.2 icmp type echo-request counter accept
  }

  chain forward-from-lan {
    counter
    accept
  }
}
Appendix F. Test script

#!/usr/bin/env python3

import os, subprocess
from time import sleep

subprocess.call(['ssh', 'root@172.16.2.3', 'arp -s 172.16.2.1 68:05:ca:39:3b:bb'])
subprocess.call(['ssh', 'root@172.16.1.3', 'arp -s 172.16.1.1 6c:0b:84:08:fb:ac'])
subprocess.call(['ssh', 'root@172.16.2.3', 'iperf3 -s'])
subprocess.call(['ssh', 'root@172.16.2.2', 'arp -s 172.16.2.1 68:05:ca:39:3b:bb'])
subprocess.call(['ssh', 'root@172.16.1.3', 'iperf3 -b 0 -u -t 125 -f k -i 0 -c 172.16.2.3 <payload size> >> /root/result'])
subprocess.call(['ssh', 'root@172.16.1.1', 'arp -s 172.16.1.2 6c:0b:84:09:00:c4'])
subprocess.call(['ssh', 'root@172.16.1.1', 'arp -s 172.16.2.2 6c:0b:84:08:fa:78'])
o.s.system('arp -s 172.16.1.1 6c:0b:84:08:fb:ac')
sleep(2)
resultfile = open('/root/result', "a")
temp = open("/root/temp", "w")
subprocess.call(['ping', "-s <payload size>", "-w 120", "-q", "172.16.2.2"], stdout=temp)
temp.close()
temp = open("/root/temp", "r")
pingresults = temp.readlines()
resultfile.write(pingresults[-1])
temp.close()
resultfile.close()