VILNIUS TECH
Vilnius Gediminas Technical University

Mokslas – Lietuvos ateitis / Science – Future of Lithuania
ISSN 2029-2341 / eISSN 2029-2252
2021 Volume 13, Article ID: mla.2021.15180, 1–5
https://doi.org/10.3846/mla.2021.15180

Electronics and electrical engineering
Elektronika ir elektros inžinerija

ANALYSIS OF LINUX OS SECURITY TOOLS FOR PACKET FILTERING AND PROCESSING

Dmitrij MELKOV*, Šarūnas PAULIKAS

Vilnius Gediminas Technical University, Vilnius, Lithuania

Received 22 June 2021; accepted 30 June 2021

Abstract. Open-source software and its components are widely used in various products, solutions, and applications, even in closed-source. Majority of them are made on Linux or Unix based systems. Netfilter framework is one of the examples. It is used for packet filtering, load-balancing, and many other manipulations with network traffic. Netfilter based packet filter iptables has been most common firewall tool for Linux systems for more than two decades. Successor of iptables – nftables was introduced in 2014. It was designed to overcome various iptables limitations. However, it hasn't received wide popularity and transition is still ongoing. In recent years researchers and developers around the world are searching for solution to increase performance of packet processing tools. For that purpose, many of them trying to utilize eBPF (Extended Berkeley Packet Filter) with XDP (Express Data Path) data path. This paper focused on analyzing Linux OS packet filters and comparing their performances in different scenarios.

Keywords: Linux, Netfilter, iptables, nftables, eBPF, XDP, firewalls, packet filters.

Introduction

Nowadays, open-source software is extensively used in many different areas and devices. From supercomputers and enterprise level network devices to smartphones and various Internet of Things (IoT) devices. Open-source software is decentralized in many cases, so multiple stakeholders can do adjustments or fixes. Development of open-source software is rapid, and it basically open to everyone without any cost. So, it is quite clear why it has achieved such popularity.

In networking, open-source code is also playing a major role. TCP/IP stack in Linux and Unix systems is solid, mature and offers switching, routing, firewalling, and other functionality with possibility to tune it for various purposes. That is why it widely used by many companies in their proprietary software and hardware products using proprietary software. Cisco Open NX-OS is built on Linux kernel (Cisco DevNet, 2021). Others Cisco's operating systems such as IOS-XE, NX-OS are also built on Linux. Juniper's Junos OS Evolved runs natively on Linux whereas classical Junos OS runs over an instance of the FreeBSD (Juniper Networks, 2021). Citrix Systems Netscaler software is also based on FreeBSD (Citrix, 2017).

Additionally, in a lot of Linux based systems Netfilter framework is utilized for packet filtering, load-balancing and other manipulations with IP packets. The most famous Netfilter's utility iptables was introduced back in 1998. It became a standard for firewalling tools. However, various architectural limitations of iptables have pushed developers to introduce his successor – nftables. It overcome main iptables limitations (Westphal, 2016). For example, addition and removal of rules is now atomic, it is especially useful for applications such as Kubernetes or Red Hat's Openshift where ruleset updates are constant and very frequent. Most rule handling was moved to userspace. Support of new protocol will not require to implement kernel changes. Instead, only nft tool need to be updated. Since version 3.13 nftables was merged into the Linux kernel. Despite all advantages, full migration from iptables to nftables have not happened yet. In 2018 iptables was considered legacy tools and iptables-nft tool was released to translate iptables rules into nftables and to enforce migration.

Increasing network speeds and transferred data rate has led Linux community to think about alternative options of iptables replacer. In recent years, a lot of attention is focused on using eBPF functionality to make iptables alternative based on it. First results are showing that performance gain

*Corresponding author. E-mail: *dmitrij.melkov@vilniustech.lt*

could be quite significant. What was one of the controversies in case of nftables.

This paper is structured as follows. In "Related work" section results of similar works are presented. In section "Packet flow in Netfilter and eBPF" packet flow in iptables, nftables and eBPF is discussed. Our measurement results of UDP traffic are presented in section "Experimental results". Conclusions are provided in last section.

1. Related works

In our previous work "Performance testing of Linux firewalls" (Melkov et al., 2020) we measured how TCP throughput depends on number of installed rules in iptables or nftables. Experiment was done using different Netfilter chains: PREROUTING, INPUT, FORWARD, OUTPUT. Different amount of virtual CPU (vCPU) was used on virtual machine with installed iptables or nftables. Results showed performance advantage of iptables over nftalbes in all scenarios. Best result was achieved using ipset extension. Scholz et al. (2018) examined how number of processed packets per second depends on number of rules in iptables and nftables. Then they compared these results with results when XDP was used. XDP utilize eBPF virtual machine to process packets before they reach kernel. Witch such set-up they were able to reach four times better results than using iptabes and nftables. Bertrone et al. (2018a) in paper "Toward an eBPF-based clone of iptables" proposed architecture of a possible replacement of iptables with an equivalent software based on eBPF technology. They proposed how to implement matching algorithm and connection tracking using eBPF preserving iptables semantic and syntax. In their another paper "Accelerating Linux Security with eBPF iptables" (Bertrone et al., 2018b) they made performance tests of iptables and their designed iptables alternative – bpf-iptables. Two different tests were done. In first test they measured UDP throughput in FORWARD chain and in second test TCP throughput was measured in INPUT chain. In both cases custom bpf-iptbales tool outperformed standard iptables. Greater advantage seen with increased number of installed rules. In further work from same authors "Securing Linux with a faster and scalable iptables" (2019) they did nftables performance test in same scenarios. Results were worse than using custom created bpf-iptables and standard iptables. Tumolo form Politecnico di Torino in his master thesis "Toward a faster iptables in eBPF" (Tumolo, 2018) implemented his own version of iptables using eBPF and measured UDP throughput and ICMP latency. Results were compared with standard iptables results. Higher throughput and lower latency were achieved using his custom created bpf-iptables tool. Article "Benchmarking nftalbes" published by Sutter (2017) confirms other performance testing of iptables and nftables. In this article nftalbes was able to outperform iptables only in scenario when native nftables set functionality was used. Using this functionality, it is possible to add multiple targets into single match rule. But same result could be achieved using ipset extension for iptables. It was confirmed in the same article and in our previous paper.

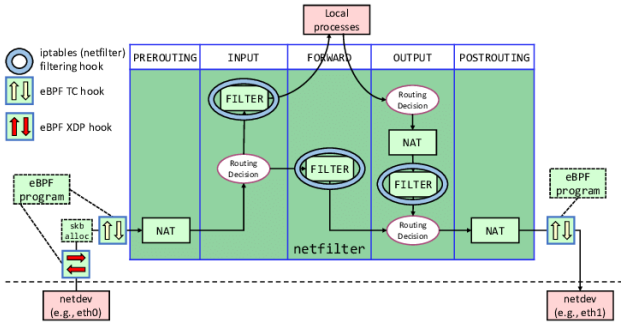

Figure 1. Location of Netfilter chains and eBPF hooks (Miano et al., 2019a)

2. Packet flow in Netfilter and eBPF

In case of Netfiler/iptables each packet travels through several chains and tables. First, every packet that enters system will go thorough Raw, Mangle and NAT tables of PREROUTING chain. If packet destined to local applications, it would enter INPUT chain and will go through Mangle and Filter tables. Otherwise, when packet should be routed, it will enter FORWAD chain and will go through same tables as in INPUT chain. If packet was locally generated, it enters OUTPUT chain and will also go through Raw, Mangle, NAT, and Filter tables. At the very end of the path, every packet enters POSTROUTING chain which contains mangle and NAT tables. In case of ipbales packet will go through all chains. Unlike iptbales, nftables does not have all chains and tables by default. So, it is up to user to determine which chains and tables should be used (Suehring, 2015).

eBPF programs can be attached even before packet enter PREROUTING chain of Netfilter. eBPF based program XDP provides possibility to process packets before TCP/IP stack achieving higher performance of packet processing (Miano et al., 2019b). Location of Netfilter's chains, most popular tables and eBPF hooks are shown in Figure 1.

3. Experimental results

Experiment was done in network laboratory at the Faculty of Electronics of the Vilnius Tech University. We used same testbed as in our previous work (Melkov et al., 2020). It was designed according to recommendations provided int RFC 3511.

Testbed was made from 2 IBM System x3550 M3 servers with installed ESXi hypervisor and physical switch Cisco Catalyst 3650 series. VM with installed iptables version 1.8.3 was hosted on ESXi server with 12 CPU × 2.40 GHz and 96 GB of RAM. To translate rules of iptables into nftables, iptables-nft tool was used. Sender and receiver VM's were hosted on another ESXi server with 8 CPU × 2.4 GHz and 96 GB of RAM. For each

VM we dedicated 8 GB of RAM and 2 vCPU. As operating system we used Ubuntu 18.10 version. To ensure L3 connectivity, sender and receiver had IP addresses from different subnets and were connected to separate virtual Switch (vSwitch). VM with installed packet filter had two interfaces, one for subnet of sender and another for subnet of receiver. Each interface was connected to separate vSwitch. Two 1 Gb/s uplinks from each ESXi server were connected to physical Cisco switch. Logical diagram of testbed is shown in Figure 2. For traffic generation and analysis iPerf tool was used. Measurements were done in FORWARD and INPUT chains of iptables and nftables using 1, 2 and 4 dedicated vCPU for VM. We measured three different UDP flows. First flow consisted of 128 B packets with 15 Mbps bandwidth, second flow consisted of 512 B packets with 30 Mbps bandwidth and third flow consisted of 1280 B packets with 45 Mbps flow. Measurements for each flow were done separately. For the beginning, VM with installed packet filter had 2 dedicated vCPU.

Experimental results of iptables are shown in Figure 3. As we can see from the graph, in case of 15 Mbps flow of 128 B packets degradation starts at around 5 thousand installed rules. Degradation means that packet filter is not able to process all UDP packets and starts to drop some of them. When 30 Mpbs flow consists of 512 B packets degradation starts at around 11 thousand installed rules and in case of 45 Mbps flow that consists of 1280 B packets degradation starts at around 20 thousand rules. After the point when packet filter has more than 20 thousand installed rules number of processed packets per second decreases in same manner for each flow. So, at this point there is no difference for firewall what size packets are, as number of processed packets will remain the same. Then we repeated measurements, but filtering was done in INPUT chain. In that scenario receiver was VM with installed packet filter itself. In contrast to our previous work (Melkov et al., 2020), when TCP throughput was better in INPUT chain, in case of UDP traffic results were the same as in FORWARD chain. Also, there were no difference in results with 1 or 4 dedicated vCPU.

Same measurements were done filtering packets with nftables instead of iptables. Experimental results are shown in Figure 4. Performance of nftables worse than iptables

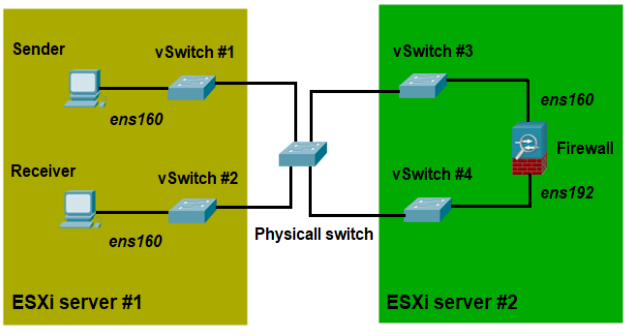

Figure 2. Logical network diagram of testbed (Melkov et al., 2020)

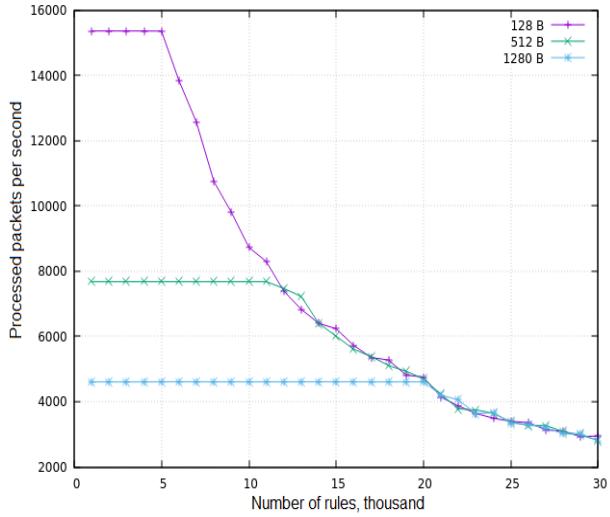

Figure 3. Processed UDP packets per second using iptables

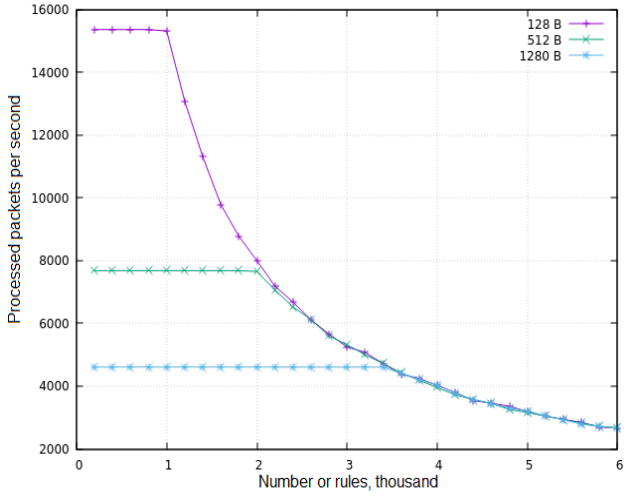

Figure 4. Processed UDP packets per second using nftables

as decrease in number of processed packets per seconds starts earlier. It starts at around 1, 2 and 3.5 thousand of installed rules for 128 B, 512 B and 1280 B packets accordingly. As in previous case, performance of filtering packet in FORWARD and INPUT chain is the same.

In Figure 5 advantage of iptables over nftables are shown. As we can see form this picture, advantage is greater with smaller size packets. It is increases with number of installed rules into packet filter. With 6 thousand rules installed, iptables processing around 5.2 times more 128 B packets. In case of 512 B and 1280 B packets advantage is around 2.9 and 1.7 times accordingly.

In order to find reason of decrease in number of processed packets we tried to measure CPU utilization during the tests on VM where packet filter was installed. We were able to find relation between CPU performance decrease and CPU utilization. On Figure 6 CPU utilization during iptables test with 128 B packets are shown. As we can see from this graph, when number of installed rules

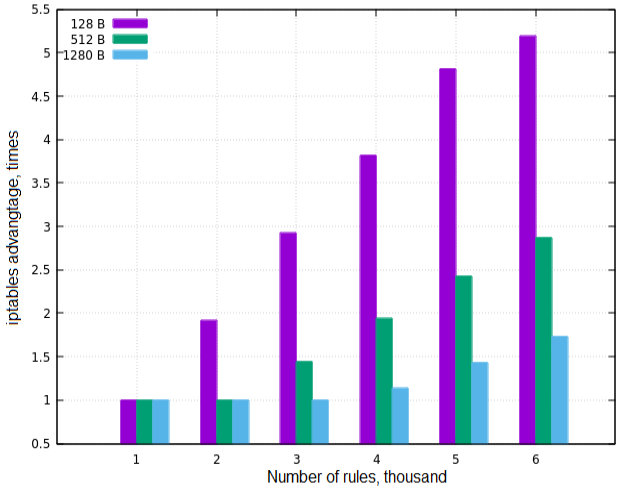

Figure 5. Advantage of iptables over nftables

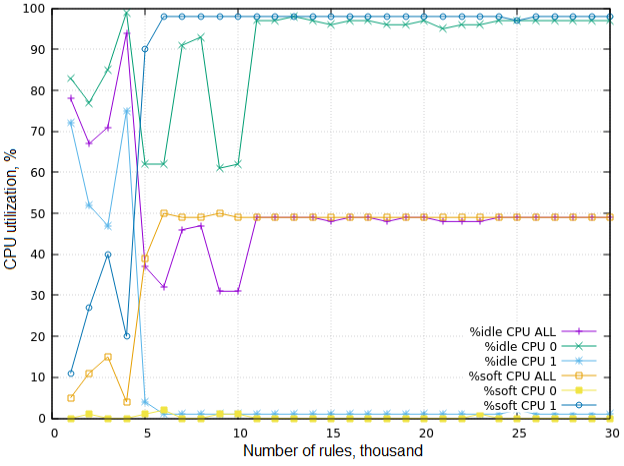

Figure 6. CPU utilization filtering 128 B packets in iptables

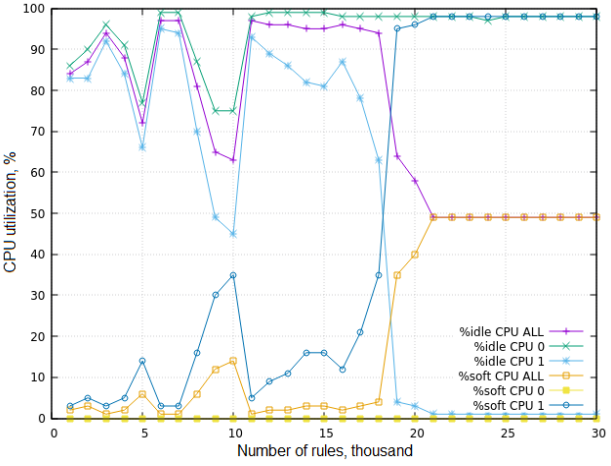

Figure 7. CPU utilization filtering 1280 B packets in iptables

is lower than 5 thousand there are some fluctuations of CPU utilization. But when there are more than 5 thousand installed rules usage of CPU1 for handing software interrupts reaching almost 100%, while CPU0 is idle for almost 100%. Breaking point of 5 thousand rules are the same as the point of performance degradation in Figure 3 for 128 B packets. In Figure 7 CPU utilization during test of iptables with 1280 B packets are shown. Again, CPU1 was utilized for almost 100% starting at around 20 thousand rules. It matches degradation point of flow with 1280 B packets as in Figure 3. Same CPU utilization pattern was also visible in case of using nftables.

Conclusions

Development of open-source packet filtering tool attracts a lot of attention from developers and researchers. The main goals are to overcome limitations of iptables and to achieve higher filtering performance. Nftables solves a lot of limitations. However, our and other benchmarks shows that if entire rule set is not designed for nftables, performance is worse than using standard iptables (Tumolo, 2018). It is problem for applications such as Kubernetes that uses ipables as transition to nftables without performance degradation will require change of ruleset's logic. Use of eBPF virtual machine with XDP program can bring significant increase of performance, but still a lot of work should be done to keep various Netfilter functionality such as connection tracking and other.

In our paper we presented how number processed UDP packets per second depend on number of installed rules into iptables or nftables packet filter. It was concluded, that iptables outperforms nftables. Also, performance is not depending on chain where filtering is performed and not depending on amount of dedicated vCPU for VM with installed packet filter. We noticed, that packet filters use only single vCPU during processing of UDP packets.

So far researcher's attention mainly focused on measuring TCP throughput when filtering is performed in eBPF virtual machine. Our future works should be focused on measuring UDP flow characteristics (packet loss, latency, jitter) while eBPF and XDP technologies are used for filtering.

References

Bertrone, M., Miano, S., Pi, J., Risso, F., & Tumolo, M. (2018a). *Toward an eBPF-based clone of iptables* [Conference presentation]. The Technical Conference on Linux Networking, Montreal, Canada.

Bertrone, M., Miano, S., Risso, F., & Tumolo, M. (2018b). *Accelerating Linux security with eBPF iptables* [Conference presentation]. The ACM SIGCOMM 2018 Conference, Budapest, Hungary. SIGCOMM. https://doi.org/10.1145/3234200.3234228

Cisco DevNet. (2021). *Open NX-OS Linux.* https://developer.cisco.com/docs/nx-os/#!open-nx-os-linux/open-nx-os-linux

Citrix. (2017). *How to check the version of FreeBSD on NetScaler.* https://support.citrix.com/article/CTX221291

Juniper Networks. (2021). *Junos OS Evolve overview*. https://www.juniper.net/documentation/us/en/software/junos/evo-overview/topics/concept/evo-overview.html

Melkov, D., Šaltis, A., & Paulikas, Š. (2020). *Performance testing of Linux firewalls* [Conference presentation]. 2020 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream), Vilnius, Lithuania. IEEE. https://doi.org/10.1109/eStream50540.2020.9108868

Miano, S., Bertrone, M., Risso, F., Vásquez Bernal, M., Lu, Y., & Pi, J. (2019a). Securing Linux with a faster and scalable iptables. *ACM SIGCOMM Computer Communication Review*, *49*(3), 2–17. https://doi.org/10.1145/3371927.3371929

Miano, S., Doriguzzi-Corin, R., Risso, F., Siracusa, D., & Sommese, R. (2019b). Introducing SmartNICs in server-based data plane processing: the DDoS mitigation use case. *IEEE Access*, *7*, 107161–107170. https://doi.org/10.1109/ACCESS.2019.2933491

Scholz, D., Raumer, D., Emmerich, P., Kurtz, A., Lesiak, K., & Carle, G. (2018). *Performance implications of packet filtering with Linux eBPF* [Conference presentation]. 30th International Teletraffic Congress, Vienna, Austria. IEEE. https://doi.org/10.1109/ITC30.2018.00039

Suehring, S. (2015). *Linux firewalls: Enhancing security with nftables and beyond* (4th ed.). Addison-Wesley.

Sutter, P. (2017). Benchmarking nftables. *Red Hat Developer blog*. https://developers.redhat.com/blog/2017/04/11/benchmarking-nftables

Tumolo, M. (2018). *Towards a faster iptables in eBPF* [Master thesis]. Politecnico di Torino.

Westphal, F. (2016). What comes after "iptables"? Its successor, of course "nftables". *Red Hat Developer blog.* https://developers.redhat.com/blog/2016/10/28/what-comes-after-iptables-its-successor-of-course-nftables

LINUX OS PAKETŲ FILTRAVIMO IR APDOROJIMO SAUGUMO PRIEMONŲ ANALIZĖ

D. Melkov, Š. Paulikas

Santrauka

Atvirojo kodo priemonės plačiau naudojamos skirtinguose produktuose ir programose. Dauguma iš jų yra padaryta panaudojant *Linux* arba *Unix* sistemas. *Netfiler* tvarkyklė yra vienas iš pavyzdžių. Ji naudojama paketams filtruoti, apkrovai paskirstyti ir kitoms manipuliacijoms su paketais atlikti. *Netfilter* paketų filtras *ipatables* jau du dešimtmečius yra populiariausia *Linux* ugniasienė. Nauja ugniasienė *nftables* buvo pristatyta 2014 metais ir turėjo įveikti *iptables* trūkumus. Tačiau *nftables* taip ir negavo visuotinio pripažinimo, daug sistemų taip ir nebuvo perkeltos į iptables. Todėl pastaruosius metus mokslininkai ir programinės įrangos kūrėjai ieško naujo sprendimo padidinti paketų apdorojimo našumą. Tam jie bando išnaudoti tokias technologijas kaip eBPF ir XDP. Šio straipsnio tikslas padaryti *Linux* OS paketų filtro analizę ir palyginti jų našumą skirtinguose scenarijuose.

Reikšminiai žodžiai: *Linux*, *Netfilter*, *iptables*, *nftables*, eBPF, XDP, ugniasienė, paketų filtrai.