DoS Attack in Cloud Based Kubernetes Using eBPF

Subjects: Computer Science, Cybernetics

Contributor: Amin Sadiq , Hassan Jamil Syed , Asad Ahmed Ansari , Ashraf Osman Ibrahim , Manar Alohaly , Muna Elsadig

Kubernetes is an orchestration tool that runs and manages container-based workloads. It works as a collection of different virtual or physical servers that support multiple storage capacities, provide network functionalities, and keep all containerized applications active in a desired state. It also provides an increasing fleet of different facilities, known as microservices. However, Kubernetes' scalability has led to a complex network structure with an increased attack vector. Attackers can launch a Denial of service (DoS) attack against servers/machines in Kubernetes by producing fake traffic load, for instance. DoS or Distributed Denial of service (DDoS) attacks are malicious attempts to disrupt a targeted service by flooding the target's service with network packets. Constant observation of the network traffic is extremely important for the early detection of such attacks. Extended Berkeley Packet Filter (eBPF) and eXpress Datapath (XDP) are advanced technologies in the Linux kernel that perform high-speed packet processing. In the case of Kubernetes, eBPF and XDP can be used to protect against DDoS attacks by enabling fast and efficient network security policies.

Denial of service (DoS) Distributed Denial of service (DDoS) attack kubernetes Extended Berkeley Packet Filter (eBPF) eXpress Datapath (XDP)

1. Introduction

Before the 1990s, the monitoring and analysis of the packets used to be performed using the classic packet filtering mechanism in which all the packets were copied from the kernel space to the userspace, leading to an increased packet processing latency. In 1992–1993, Steven McCanne and Van Jacobson introduced a mechanism named Berkeley Packet Filter (BPF), in which not all the packets are copied from the kernel space to the userspace [1]. The architecture of the BPF virtual machine is designed for a 32 bits machine with a fixed-length instruction set. BPF was first introduced with the Unix operating system. Then, this filtering mechanism was improved to the Extended Berkeley Packet Filter (eBPF), which was adopted by Linux and Windows [2]. eBPF is a Linux technology that allows for the safe and efficient execution of code within the Linux kernel. It provides a powerful mechanism for implementing complex network functions, such as firewalls, load balancers, and network monitoring, without the need for custom kernel modules. eBPF is a program executed on the Linux kernel when some event is triggered. eBPF provides a wide range of functionalities such as packet monitoring, tracing new processes, and the detection of any event generated by the computer, etc. eBPF has an improved instruction set architecture (ISA) architecture with more registers. Unlike BPF, eBPF filters all the packets at the kernel space to

better decrease the latency. Moreover, the high-speed processing power of eBPF facilitates the analysis of every packet in the network. XDP is a networking technology that provides a fast and efficient way to process network packets at the kernel level. It allows for the processing of packets before they reach the higher-level networking stack, enabling faster and more efficient handling of network traffic. XDP is well-suited for use in applications that require high-speed packet processing, such as network security, load balancing, and network monitoring.

BPF Programs are written in restricted C programming language due to the Linux kernel. However, a tool named BPF Compiler Collection, i.e., BCC [3], makes the BPF program much easier to write using Python and some restricted C languages. This BPF code is first compiled into BPF bytecode in userspace using a low-level virtual machine (LLVM) or Clang. After passing through a verifier (for authentication and verification), a just-in-time compiler helps to convert this code into a native machine code and execute the program using a separate virtual machine on Linux Kernel [4]. Infinite loops, global variables, etc., are not supported, so this limitation while implementing the programs must be considered.

Today, Kubernetes has become a leading open-source orchestration tool that provides dynamic scalability, manages different resources needed by the physical or virtual servers, and fulfills the different adaptations and dependencies of the applications. Kubernetes communicates between the different collections of containers on which different applications are running. It is a complex system that allows communication between different components within a cluster and between pods. The structure of a Kubernetes cluster network can be broken down into several key components: cluster network, pods network, etc. Like any complex software system, Kubernetes might have security flaws that attackers could use against users if it is not secured properly. If the cluster network is not properly secured, an attacker could potentially intercept sensitive data or inject malicious traffic into the network. Kubernetes uses containers to deploy applications, and these containers are built from images. If an attacker can inject malicious code into a container image, they could potentially compromise the application running in that container, Kubelet, which facilitates communication between various nodes, and containers that are executing on the nodes can all be big contributors to making the cluster vulnerable to attack while considering the cluster network. In addition, it provides the interface platform for the application that needs to provide high availability and scalability through its microservices [5]. Users of Kubernetes test the deployment features of an application by diverting traffic between different containers. On the other hand, malicious users may compromise Kubernetes clusters to control other users' containers. Over the past years, methods and gadgets for launching an attack have drastically improved. DoS (Denial of Service) or DDoS (Distributed Denial of Service) are potentially the most renowned attacks. The goal behind DoS or DDoS attacks is to bring the server out of reach for others and it will create a potential gap in the performance overhead of the system. eBPF is a revolutionary technology in a Linux kernel that handles everyday tasks and affects the processes running on the server. eBPF has a significant impact on system performance because of different complexities and the frequency of the events. It can be used to perform various tasks ranging from package isolation to opening a particular document in the file system. Unlike other ways to change the behavior of a part, eBPF does not require part recompilation or part module assembly and is protected from execution by security checks by stacking code verifiers. Each second is important in detecting malicious packets so that particular prevention methods can be applied. Express Data Path (XDP) [6] gives the quick execution of group dealing with and programmable association way in the cycle of Linux. XDP

generally processes the packages' level of the stack, which prompts the quick presentation without compromising the programmability.

In the past few years, detection and prevention approaches have been topics of interest in the fields of networks and cybersecurity. One of the major security risks to many networks in the last ten years is a DDoS attack. It can not only stop authorized users from using and accessing network resources, but it can also destroy the network [7]. Cloud services and microservices are at high risk of DDOS attacks [8]. In 2019, it is reported that a Kubernetes cluster that was being utilized to host a cryptocurrency mining operation came under attack from a distributed denial-of-service (DDoS) attack. The attack, which wrecked the Kubernetes cluster and halted the mining process, was launched by the attackers using a huge number of compromised machines. A DdoS attack against a Kubernetes cluster hosting a machine learning platform was reported by researchers in 2020. The attack was carried out by the attackers using a large number of compromised devices, which led to the collapse of the Kubernetes cluster. In [9][10], it was highlighted that these types of cyber attacks are not only limited to a specific area but operate in a different area as well, which helps the potential attackers use the open and less privileged loophole and then make the entire system compromised.

2. eBPF, Microservices, DDOS, and the Application of the eBPF to Various Containerized Services on Kubernetes

Several researchers [11][12][13], and [14][15][16] have proposed different strategies and solutions for the use of eBPF on different aspects of emerging technologies. A brief description of the related studies in **Table 1** is provided.

Table 1. Summary of related studies.

S.No.	Existing Work	Summary	Limitations
1	[12]	For the first time, eBPF was used for the detection tasks of the kernel like observing the page faults, over-viewing the memory distribution, etc. The author(s) has encountered the advantage and success factor of eBPF when developing complex networks.	While focusing on the numerous complicated services, the proposed design should also assess the system overhead.
2	[11]	The author(s) have proposed a solution for the Intrusion detection of the system by using eBPF at the kernel which filtered the huge amount of network packets using the matching rule set of the snorting method.	It seems that there should be state-of-the-art comparison with other features to evaluate the performance of the system
3	[15]	The aim is to implement an efficient and effective processing pipeline that will help to prevent Distributed Denial of Service (DDoS) attacks. The authors have increased the mitigation power at the server end which will drop the DDoS by using the specific ruleset. eBPF is a flexible tool that will help to sample the network traffic of the operating system using hardware-based filtering.	When the server has a lot of resources preserved on it, the performance of the work will be reduced by the suggested approach.

S.No.	Existing Work	Summary	Limitations
4	[16]	The author has proposed a solution based on eBPF which bypasses the kernel and filters all the network packets at the userspace which usually skips the overhead of the Linux network stack. The proposed solution has overcome the performance issue that arises when packets are filtered and analyzed at kernel space.	The recommended architectural designs perform with a small number of alterations. The lack of drivers and other resources will require significant adjustments to the solution.
5	[14]	The author(s) proposed a non-intrusive protocol-independent intelligent analytics system, more efficient and accurate and also provides the point-to-point observability solution with works on eBPF. It also enables the transparency of the packets for an application like Kubernetes in which having nodes, clusters, pods, containers, etc.	The proposed machine learning-based observability approach will become more complicated as the number of nodes and pods rises.
6	[17]	A framework was introduced by the author on Network Functions Virtualization to enhance its capability for in-kernel packet processing applications which provides flexibility dynamically in the run-time environment. The framework used for the development of the complex networks provides efficiency in the kernel data plane and flexible user-space control plane for the persistence environment.	There is a potential bottleneck mechanism that could cause a delay in the mechanism's latency when a large number of packets are transferred from kernel space to userspace.
7	[13]	The author(s) have highlighted the benefits of XDP and eBPF using the offloading mechanism based on eBPF to the kernel space from the userspace. The offloading-based mechanism is an optimized solution for the packet processing.	When a significant number of packets arrive at the same time, the proposed solution will begin to slow down and SmartNIC will become stuck.

[11][12][18].

f eBPF in
rther help
attack on

the server side and also bypassing the kernel and filtering out all the detailed information on the user side, which will overcome the overhead on the Linux network stack [15][16]. In [14][17], the accuracy and efficiency of the observability and monitoring based on the microservices were discussed, along with using a framework for Network Function Virtualization to achieve flexibility. In [13], the authors have discussed the offloading mechanism of the packet using eBPF.

In [11][12], the kernel used eBPF for detection duties like looking for page faults and monitoring the distribution of memory, among other things. The benefit and success element of eBPF was experienced while creating complex networks. Use of eBPF at the kernel, which filtered a sizable number of network packets using a matching rule set of the Snort approach, as the solution for the system's intrusion detection.

In [15][16], the objective is to develop a processing pipeline that is effective and efficient, which will aid in preventing DDoS attacks. By employing a specific ruleset, the authors have boosted the server-end mitigation strength, decreasing the DDoS. eBPF-based solutions essentially bypass the kernel and filter all network packets in userspace, avoiding the Linux network stack's overhead in the process. The performance problem that occurs when packets are filtered and examined in the kernel space has been resolved by the suggested fix.

In [14][17], a non-intrusive, protocol-independent intelligent analytics system that operates on eBPF is more effective, accurate, and offers a point-to-point observability solution. Additionally, it makes the packet transparency for the microservices possible. In addition, it enhances its capability for in-kernel packet processing applications, which provides flexibility dynamically in the run-time environment.

In [13], the advantages of XDP and eBPF utilizing the eBPF-based offloading method from the userspace to the kernel space have been noted. It is ideal to use the offloading-based approach for packet processing.

References

1. McCanne, S.; Jacobson, V. The BSD Packet Filter: A New Architecture for User-Level Packet Capture. In Proceedings of the USENIX Winter, San Diego, CA, USA, 25–29 January 1993; Volume 46.
2. Vieira, M.A.; Castanho, M.S.; Pacífico, R.D.; Santos, E.R.; Júnior, E.P.C.; Vieira, L.F. Fast packet processing with eBPF and XDP: Concepts, code, challenges, and applications. ACM Comput. Surv. CSUR 2020, 53, 1–36.
3. Scholz, D.; Raumer, D.; Emmerich, P.; Kurtz, A.; Lesiak, K.; Carle, G. Performance implications of packet filtering with Linux eBPF. In Proceedings of the 2018 30th International Teletraffic Congress (ITC 30), Vienna, Austria, 3–7 September 2018; Volume 1, pp. 209–217.
4. Nelson, L.; Van Geffen, J.; Torlak, E.; Wang, X. Specification and verification in the field: Applying formal methods to just-in-time compilers in the Linux kernel. In Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20), Virtual Conference, 4–6 November 2020; pp. 41–61.
5. Bernstein, D. Containers and cloud: From LXC to docker to Kubernetes. IEEE Cloud Comput. 2014, 1, 81–84.
6. Høiland-Jørgensen, T.; Brouer, J.D.; Borkmann, D.; Fastabend, J.; Herbert, T.; Ahern, D.; Miller, D. The express data path: Fast programmable packet processing in the operating system kernel. In Proceedings of the 14th International Conference on Emerging Networking Experiments and Technologies, Heraklion, Greece, 4–7 December 2018; pp. 54–66.
7. Fan, C.; Kaliyamurthy, N.M.; Chen, S.; Jiang, H.; Zhou, Y.; Campbell, C. Detection of DDoS attacks in software defined networking using entropy. Appl. Sci. 2021, 12, 370.
8. Alashhab, Z.R.; Anbar, M.; Singh, M.M.; Hasbullah, I.H.; Jain, P.; Al-Amiedy, T.A. Distributed Denial of Service Attacks against Cloud Computing Environment: Survey, Issues, Challenges and Coherent Taxonomy. Appl. Sci. 2022, 12, 12441.

9. Heidari, A.; Jabraeil Jamali, M.A. Internet of Things intrusion detection systems: A comprehensive review and future directions. Clust. Comput. 2022, 2022, 1–28.
10. Heidari, A.; Navimipour, N.J.; Unal, M. A Secure Intrusion Detection Platform Using Blockchain and Radial Basis Function Neural Networks for Internet of Drones. IEEE Internet Things J. 2023, 2023, 3237661.
11. Wang, S.-Y.; Chang, J.-C. Design and implementation of an intrusion detection system by using extended BPF in the Linux kernel. J. Netw. Comput. Appl. 2022, 198, 103283.
12. Miano, S.; Bertrone, M.; Risso, F.; Tumolo, M.; Bernal, M.V. Creating complex network services with eBPF: Experience and lessons learned. In Proceedings of the 2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR), Bucharest, Romania, 18–20 June 2018; pp. 1–8.
13. Hohlfeld, O.; Krude, J.; Reelfs, J.H.; Ruth, J.; Wehrle, K. Demystifying the Performance of XDP BPF. In Proceedings of the 2019 IEEE Conference on Network Softwarization (NetSoft), Paris, France, 24–28 June 2019.
14. Liu, C.; Cai, Z.; Wang, B.; Tang, Z.; Liu, J. A protocol-independent container network observability analysis system based on eBPF. In Proceedings of the 2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS), Hong Kong, China, 2–4 December 2020; pp. 697–702.
15. Bertin, G. XDP in practice: Integrating XDP into our DDoS mitigation pipeline. In Proceedings of the Technical Conference on Linux Networking, Netdev, Montréal, QC, Canada, 6–8 April 2017; Volume 2.
16. Miano, S.; Doriguzzi-Corin, R.; Risso, F.; Siracusa, D.; Sommese, R. Introducing smartnics in server-based data plane processing: The DDoS mitigation use case. IEEE Access 2019, 7, 107161–107170.
17. Miano, S.; Risso, F.; Bernal, M.V.; Bertrone, M.; Lu, Y. A framework for eBPF-based network functions in an era of microservices. IEEE Trans. Netw. Serv. Manag. 2021, 18, 133–151.
18. Abranches, M.; Michel, O.; Keller, E.; Schmid, S. Efficient Network Monitoring Applications in the Kernel with eBPF and XDP. In Proceedings of the 2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Heraklion, Greece, 9–11 November 2021; pp. 28–34.