Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 07, July-2021

Accretion of Suricta with DPDK for Traffic Monitoring using Optimized Detection System IDS/IPS

V. Pavithra
Dept. of CSE
Ramaiah Institute of Technology, affiliated to VTU
Bangalore, India

***Abstract*—In the era of Internet with rapid growth of advancing technologies, the network security concerns are increasing prominently as the number of network attacks has attracted more professional's attention. To ensure computer system protection, the intrusion detection system (IDS) and Intrusion Prevention System (IPS) Engine is been introduced as high performance network for Suricata and a network monitoring tool with open Source which is owned by Open Information Security Foundation (OISF). DPDK (Data Plane Development Kit) adopts polling method for data packet processing, which saves CPU interruption time, memory copy time, and provides the application layer with a quick and effective data packet processing system, making network applications more convenient to create. DPDK Suricata is introduced in this paper for multiple packet ACL rules support with the use of DPDK poll-mode driver for traffic analysis to get better performance. Rx-Tx threads with DPDK ports for packets interface, pre-parsing filter and rule filters to allow the relevant packets. So, it helps to build ACL key for IPv4 and IPv6, As DPDK Suricata is also added with worker mode like IPS, IDS, BYPASS modes to configure and run the rules accordingly.**

Keywords-DPDK, Suricata, Intrusion Detection System, Intrusion Prevention System, ACL, Poll Mode Driver, rules

I. INTRODUCTION

In Today's, business networks are emerging to a large extent managing high traffic is so tough and transmit 10 gigabytes per second on a backbone. So Suricata's multi-threaded architecture allows its users to scale horizontally over a single appliance by adding threads for packet processing as the amount of traffic makes it obligatory. Suricata is an IDS / IPS engine based on rules that uses externally built rule sets to track network traffic and send system administrator warnings when unusual events occur. Designed to be companionable with existing components of network protection, Suricata provides simplified functionality of output and pluggable library options to accept calls from other applications. Any suspicious activity or intrusion is usually recorded or centrally collected using a network for security information and event management.

DPDK packet processing is using polling to achieve packet processing, thereby saving time for CPU interruption and time for memory retrieval. DPDK provides a simple and efficient application layer packet processing method which makes network application more feasible. DPDK packet capture library calls PMD's network card feature to verify if the packet has been captured to analyze the data packet which is stored in cache.

In our paper we discuss about DPDK and Suricata performance by integrating Suricata into DPDK platform for more secure to monitor the networking traffic with 3 Tuple ACL rules can overcome the network security issues in a large business network to detect and prevent them by introducing Suricata pipeline for better performance. So, the ACL rule we follow are alert, drop and forward. As Access control list helps to manage and mitigate traffic accordingly by avoiding cyber-attacks if any encountered. By selecting the opmode to be IDS or IPS does the job accordingly.

Configure the DPDK interface on a Suricata work pipeline as this helps in knowing how many interfaces are bound to our user application. DPDK ACL in which shows the packet processing analysis by using DPDK poll-mode drivers which helps to send the relevant IPv4 and IPv6 on Suricata worker pipeline.

II. EASE OF USE

Suricata is one of the free software intrusion detection systems, which utilizes remotely developed run sets to screen sniffed activity and alarm when suspicious occasions arise. One of Suricata's distinctive features, particularly as opposed to Snort, is that it has a dynamic port agnostic protocol security capability. This means that when these interact over non-standard ports, it can recognize some of the more popular application layer protocols, such as HTTP, DNS and TLS.

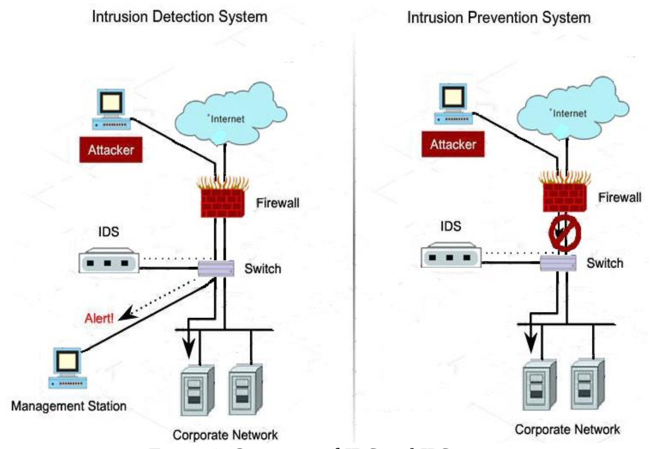

Figure 1: Overview of IDS and IPS

In the Figure 1 we know that an intrusion detection system (IDS) is an application for devices or software that control a network for malicious activity or policy violations and network monitoring tool for securely transmitting packet into an end host. if positioned at a strategic point or points within a network to track traffic to and from all devices on the network, an IDS can conduct a passing traffic analysis and compare the traffic transmitted to the database of known attacks on the subnets. The warning can be sent to the administrator until an attack is detected or an irregular activity is sensed.

An intrusion prevention device operates by actively monitoring transmitted network traffic for malicious activities and established patterns of attack. The IPS engine analyzes network traffic and continuously compares the bitstream for identified attack patterns to its internal signature database.

Intrusion detection mechanisms may also make monitoring and analysis more difficult, such as observing and responding to unusual traffic patterns or packets. Can include detection mechanisms

Data Plane Development Kit (DPDK) is a set of data plane libraries which is managed by Linux foundation as it is open source software project. The network interface controller with poll-mode drivers to offload TCP packet processing from the operating system kernel to user space. Using the interrupt-driven processing provided in the kernel, this offloading achieves higher efficiency and packet throughput than is possible.

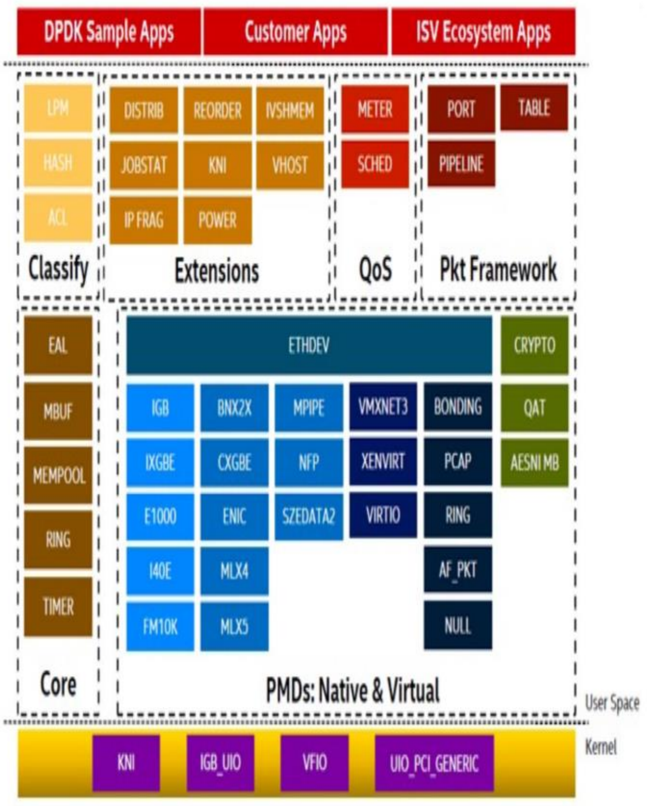

Figure 2: DPDK Structure

In this Figure 2 describes a core component of DPDK as it consists of Environment abstraction layer, Ring Buffer management, Memory Pool management, Network message buffer management, Timer management and specifying detailed structure. It picks-up and sets a single pThread on each core, and then becomes a thread run-to-completion. DPDK uses SR-IOV to bypass the Linux kernel for direct ports control. The PMD uses SR-IOV Virtual Feature interfaces one per PMD to manage the connection to port hardware. The applications, along with many utilities, use DPDK generic APIs to access and manage the ports. As each application runs on a single core for better performance, Applications need to share data, then the application needs to provide the means to communicate, and DPDK contains several types of ways to communicate between lockless rings, semaphor. The application needs to survey the ports that it wants to accept data packets called mbufs, RTE_MBUF.

III. LITERATURE SURVEY

In this section the research scholars work has been discussed with various way of bestowing knowledge out by interested key domain to highlight network related issues. With the Internet usage exceeds to peaks in day-to-day life by accessing various applications in a huge traffic, the flow of packet processing is analyzed in their setup to get better performance. DPDK and Suricata is been a main work carried out by the authors for their research as security concern and performance of an application using various environments. So, they are detailed below with the work carried by the researchers.

[1] Naga Surya Lakshmi et al. use open-source IDS tools for performance analysis of Suricata and Snort with respect to speed and size of a packet. The packet size ranging from 512, 1024, 2048 with speed of internet from 200Mbps, 400Mbps, 600Mbps, 800Mbps, 1Gbps. With two protocols i.e TCP and UDP are compared with Snort and Suricata so the packet loss is noticed at both the end. The increase in packet size with decrease in packet loss thus follows reciprocal relationship between them. The rate of packet size increases with internet speed simultaneously.

[2] Dongyan et al. comparison of Snort and DPDK to enhance performance using 100Mbps to 1000Mbps. using DPDK based intrusion detection system solves the problem of traditional snort ids which detects all the packets sent. whereas snort ids detect less compare to DPDK ids so it optimizes to better performance of an various packet size and higher speed for network flow more than 1000Mbps gives better result.

[3] Zhang et al. FloWatcher-DPDK, a lightweight high-speed software traffic controller capable of generating fine-grained per-packet and per-flow statistics with 64B packets at a line rate of 10 Gbps, using only 2 CPU cores. The method leverages the RSS hash, previously determined by the NIC and available as packet metadata, as a flow identifier to prevent unnecessary computation and memory access and employs a cautious design where flow tables are matched with line boundaries of caches. FloWatcher-DPDK outperforms state-of-the-art alternative solution to its environment and comprehensive parameter tuning was conducted.

[4] Martino et al. Proposed DPDKStat, a high-speed Statistical Traffic Analysis (STA) that combines the Intel DPDK system with the passive traffic analyzer Tstat to achieve a line rate processing of 40Gbps. In addition to publishing a concise assessment using real-time traffic traces, the architecture challenges to achieve scalability. A periodic package acquisition strategy (leveraging the latest SCHED DEADLINE Linux scheduling discipline, never considered in

previous works), conduct an in-depth analysis to reduce timestamp errors and prevent packet reconfiguration and losses.

[5] Kevin et al. Supported level for Supervisory Control and Data Acquisition (SCADA) protocols in well-known open source intrusion detection (IDS) systems. identify and improve a different IDS, Suricata, to provide support for the monitoring threats against SCADA devices that run the industrial control protocol EtherNet / IP (ENIP). At an 18Mbps throughput reflecting several SCADA networks, Suricata can run viably on hardware platforms with smaller memory and a less powerful CPU. Adding a module to process the Suricata ENIP rules did not affect output in terms of CPU utilization or Packet Drops. observed that there are varying levels of the 3 most popular open source IDS systems. Support to 3 SCADA protocols. Description of the new protocol by rule. Analyze the rule and store it in the data structure that fits the rule, Add an ENIP packet parser. ENIP module-one focused on the analysis of individual packets and of packet streams. In the end, the other was adopted and accepted for integration as part of Suricata's main distribution.

IV. METHODOLOGY

With rapid growth of high-speed networks and increasing complexity in cyber-attacks, high-performance IDS tools are needed to process packets quickly, reconstruct streams and apply pattern matching for signature-based threat detection. The efficiency of these systems depends on many factors including the efficiency of its pattern matching engine in which incoming packets are tested.

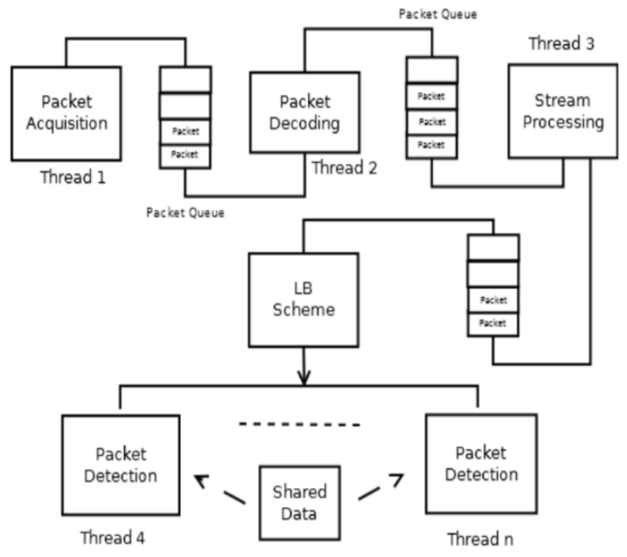

Figure 3: Suricata Architecture [5]

In the Figure 3 As Suricata plays very important role in various aspects especially with security concerns in the network by processing a data packet has a capability to detect and analyze the traffic pattern. It is compatible with other environments for more efficient and higher performance with their respective environment like DPDK Suricata is been introduced. Suricata has been developed as a "next phase IDS platform" with IPS (Intrusion Prevention System) capabilities configured to be backwards compatible with Snort rulesets. Suricata has been designed as a multi-threaded system, allowing multiple cores to be exploited.

In above architecture there are n number of threads which performs different actions as packet acquisition which collect huge amount of data which is received from the network is been sent from thread 1 to packet queue to forward the packets in thread 2. Thread 2 receives packet and start decoding and analyzing the network traffic and with the help of packet queue is been forwarded next thread i.e Thread 3 as packet stream are further portioned into sub streams and distributed across the processing task for parallelism with load balancing. In order to allow maximum use of the computing power available in a multi-core system and to avoid processor race conditions, there is a need to ensure even Load Balancing (LB) between cores. The aim of Suricata's LB is to evenly dispatch the workload across all cores and minimize idle time.

Suricata therefore shows superior performance on multi-core machines with rulesets designed for Suricata. As a result, it can easily analyze huge traffic volumes without having to cut down on the number of laws. Suricata also stands out for its ability to provide visibility into the application layer and faster HTTP stream parsing.

It may analyze HTTP traffic regardless of the port number used and does not rely on port numbers for traffic identification. Suricata also enables inspection of streams within the protocol and can therefore extract files for further analysis from HTTP sessions.

V. IMPLEMENTATION

A. Proposed Workflow

As discussed in introduction, Suricata is an open source software project which gives better performance when run on a DPDK platform for better efficiency and packet processing. The DPDK environment for packet processing applications allows for two configurations, run-to-completion and pipeline. In the run-to-completion model, the RX descriptor ring for packets is polled through an API in a specific port. Packets are then processed at the same core and put through an API for transmission on a port's TX descriptor ring.

One core poll in the pipe-line model is rings one or more RX descriptors of a port via an API. Packets are obtained and passed through a ring to another hub. The other core continues processing the packet which can then be positioned on a port's TX descriptor ring for transmission through an API.

A Poll Mode Driver (PMD) comprises of APIs which are provided to configure the devices and their respective queues through the BSD driver running in user space. In addition, a PMD accesses the RX and TX descriptors directly without interruptions (except for interrupts with Connection Status Change) to instantly receive, process and transmit packets in the user's application.

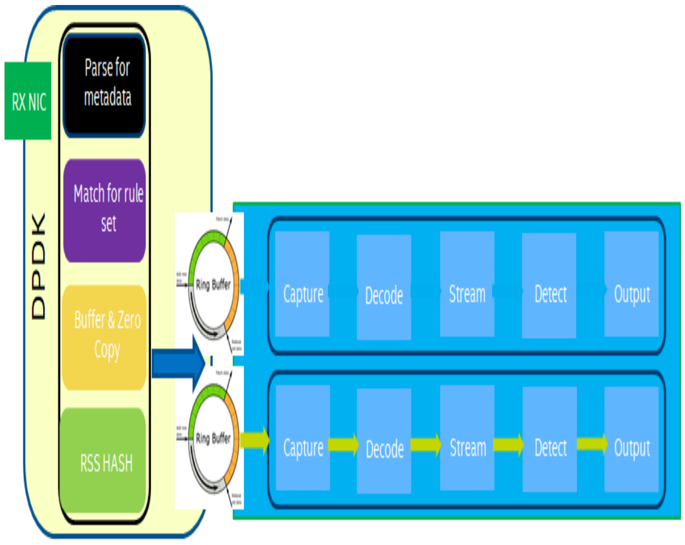

Figure 4: DPDK-Suricata Architecture for Proposed Work.

The overall software optimization strategies must be considered and balanced against available low-level hardware-based optimization features (CPU cache properties, bus speed, NIC PCI bandwidth, and so on) to achieve optimum performance. The expense of invoking the rte_eth_tx one function between multiple packets. Enable rte_eth_tx burst to take advantage of burst-oriented hardware features (prefetch data in cache, use of NIC head / tail registers) to reduce the number of Processor cycles per packet.

For example, avoiding excessive read memory accesses to ring transmitter descriptors, or consistently using pointer arrays that match cache line boundaries and sizes exactly. To suppress operations that would otherwise be inevitable, such as ring index wrap back control, apply burst-oriented device optimization techniques.

B. Experimental Setup

The test analysis is performed based on the rule set on Suricata-dpdk in turn uses 2 protocol, they are TCP and UDP. In the process of setup we mainly need dpdk pktgen, dpdk-19.11,suricta_3.0 and rules.

Our test is mainly focusing on 2 protocols with the scenario of 64,128,512,1024 byte line rate and assigning 1 DPDK thread for 10G interface for IPS/IDS. In ruleset our analysis is focused only on alert and allow to stress single worker thread. Mainly focusing on DPDK rx_burst and tx_burst to capture the line rate of packets and the threshold limit is measured only for single worker thread with minimal zero_copy. As per Figure 4.

VI. RESULTS

Figure 4 shows the traffic received from the interface is run on PacketAcquireLoop function of Suricta as there is no dedicated worker thread to handle the traffic in the interface provided so the packets scheduled for worker thread fetches from PMD interfaces which is in userspace.

Figure 5,6,7,8,9 and 10 depicts the traffic is received in the interface run on specified DPDKlcore, then this packets are forwarded to PacketAcquireLoop function with the help of ring buffer. It is receiving thread excepts in pipeline model which will not allow on same core with yaml config so the worker threads scheduled fetches packets by dequeuing ring buffer which is connected to it in the userspace.

The CPUAffinity is checked for getting enough core available to receive traffic on interface which are present in dpdkintel and run on dedicated dpdklcore. As we add rule or signature we must extend the filtering to rule the matched packet by allowing packets forwarded to copy_interface and further the worker thread process for matched rules.

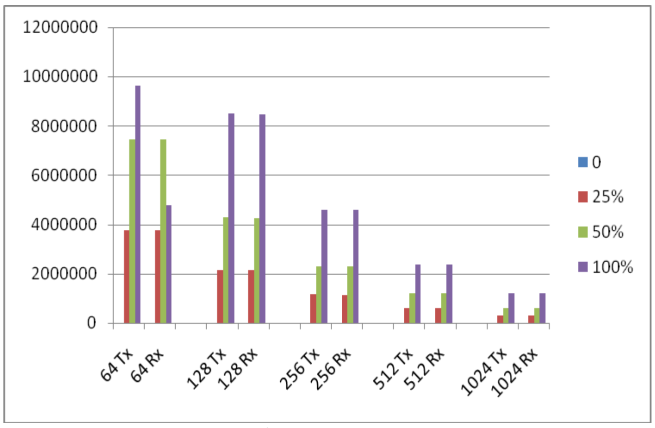

Figure 5: The number of packets sent and received on port 0

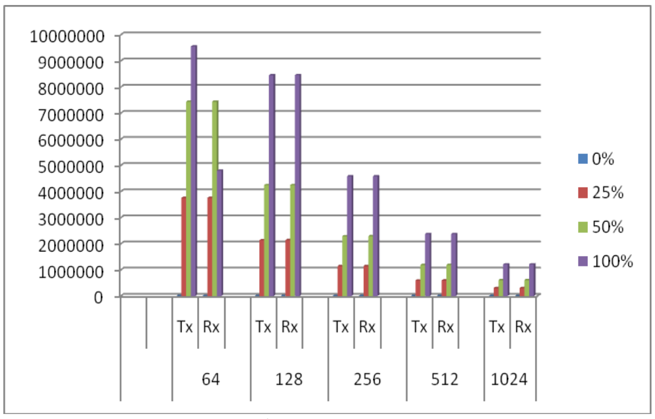

Figure 6: The number of packets sent and received on port 1

The graph above in fig 5 and fig 6 is tested for port0 and 1 with no rule hit in 64,128,256,512,1024 byte line rate as in the graph blue,red,green and purple indicates packetsize per rate.

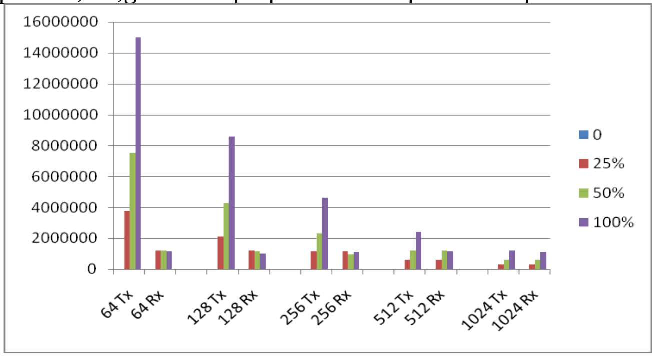

Figure 7: The number of packets sent and received on Port 0 with rule match.

The graph shown above in figure 7 and figure 8 is validated for alert rule in port 0 using yaml config and the rule set as alert udp any any -> any any, alert tcp any any -> any any. This are tested in 64,128,256,512,1024 byte line rate with 0,25,50,100 pactketsize per rate by rx and tx .

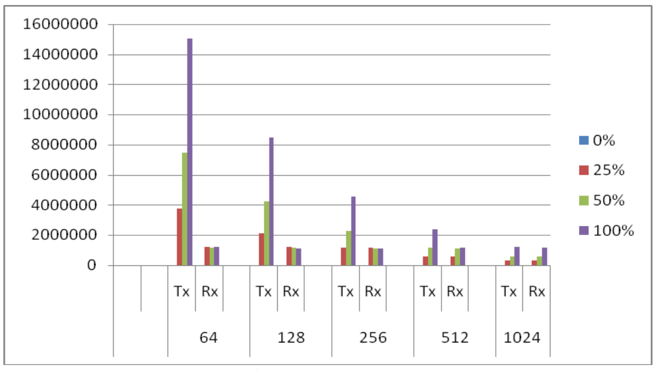

Figure 8: Number of packet sent and received in port 1

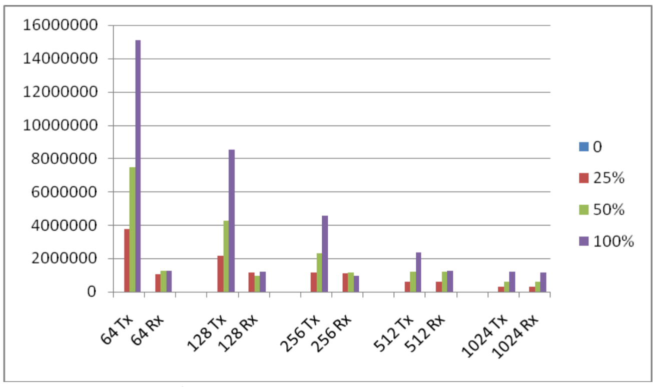

Figure 9: Number of packets sent and received in port 0 using allow rule match.

The graph in figure 9 and 10 depicts for allow rule in port 0 and 1 with yaml configured as allow udp any any -> any any, allow tcp any any -> any any. This is tested with specifed line rate and packetsize shown in the above graph.

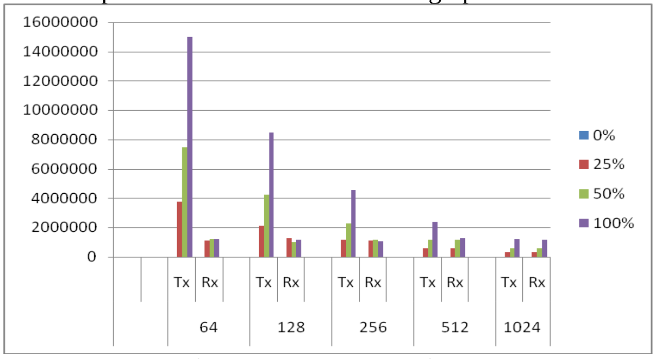

Figure 10: Number of packets sent and received for allow rule in port 1

As the Tests are done using 2 ports dedicated to suricata dpdk, the graph shown above for allow rule in port 1. So this are done for 1 DPDK thread and 1 worker thread for 2*10G.

VII. CONCLUSION AND FUTURE SCOPE

Suricata is a high performance monitoring tool for IPS and IDS with the utilization of single DPDK worker thread with the network traffic received on the interface by allocating dedicated lcore using CPU affinity(yaml config) which accelerates the IDS and IPS processing of suricata worker thread to perform better.

The future scope of work is to add multi worker thread to per core, To allocate and deallocate 'structpkt' using the mbuff sector, Pre parsing the frame to accelerate the decode logic.

ACKNOWLEDGMENT

I would like to acknowledge the Ramaiah Institute of Technology college Management, Staff and my Mentor for their very indeed and extreme support and encouragement to do this research.

REFERENCES

[1] M. Naga Surya Lakshmi, Y Radhika, "*A Comparative paper on measuring the performance of Snort and Suricata with variable packet size and speed*", (IJET) International Journal of Engineering & Technology, 8(1) (2019) 53-58
[2] Dongyan Zhang and Shuo Wang "*Optimization of Traditional Snort Intrusion detection system*", IOP Conference Series: Materials Science and Engineering, 2019
[3] Roman Fekolkin, "*Intrusion Detection and Prevention Sysytems: Overview of Snort and suricata*", 2015
[4] Tianzhu Zhang, Leonardo Linguaglossa, Massimo Gallo, Paolo Giaccone, Dario Rossi,"*FloWatcher-DPDK: light-weight line-rate flow-level monitoring software*", 2019.
[5] Kevin Wong, Craih Dilabough, Nabil Seddigh, Biswajit Nandy,"*Enhancing Suricata Intrusion Detection System for Cyber Security in SCADA Networks*", IEEE 30th Canadian Conference on Electrical and Computer Engineering(CCECE), 2017.
[6] Haozhe Ren, & Mei Nian. (2018). Dpdk-based high-speed packet acquisition method. Computer system applications, 27(6), 242-245.
[7] Park, W., & Ahn, S. (2017). Performance comparison and detection analysis in snort and suricata environment. Wireless Personal Communications, 94(2), 241-252.
[8] Ning Zhao, & Shucui Xie. (2016). Analysis and application of dpdk-based efficient packet acquisition technology. Computer engineering and science, 38(11), 2209-2215.