iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** VI **Month of publication:** June 2022

DOI: https://doi.org/10.22214/ijraset.2022.44761

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VI June 2022- Available at www.ijraset.com

Network Intrusion Detection and Prevention

Saloni Anand[1], Kshitij Patne[2]
Vellore Institute of Technology, Vellore

Abstract: ***Intrusion Detection systems are now increasingly significant in network security. As the number of people using the internet grows, so does the chance of a cyberattack. People are adopting signature-based intrusion detection systems. Snort is a popular open-source signature-based intrusion detection system. It is widely utilised in the intrusion detection and prevention arena across the world. The aim of this research is to provide knowledge about intrusion detection systems, application vulnerabilities, and their prevention methods and to perform a comparison of the latest tools and mechanisms used to detect these threats and vulnerabilities.***

Keywords: Intrusion Detection System (IDS), Intrusion Prevention System (IPS), Snort Tool, Signature-based, Anomaly Based Tool

I. INTRODUCTION

The primary concerns of our everyday network usage are security and dependability. However, as network technology advances, attackers become more advanced than security. Although firewalls and router-based packet filtering are vital factors of a network security design, they are not sufficient on their own. As a result, the concept of SNORT Intrusion Detection System (IDS) and SNORT Intrusion Prevention System (IPS) is attracting security specialists. Snort, being an open-source intrusion detection system, is simple to configure and deploy in any setting. Some research articles are evaluated based on different performance criteria such as detection accuracy, scalability, and the capacity to identify unknown assaults. To address many difficulties highlighted during the literature study, such as low detection rate, inability to handle large amounts of traffic, unsupported automatic tweaking, and so on, our solution offers a level-based architecture. All levels are meant to be incremental, that is, able to supply the necessary functionality as well as its lower levels. To demonstrate the efficacy of the suggested architecture, we used Code Refactoring to incorporate it all into Snort Tool. In addition, an environment set up for future evaluation of the enhanced Snort Tool performance was provided.

II. INTRUSIVE DETECTION AND PREVENTION

An intrusion may be defined as unlawful entry to someone's property or territory, but in computer science, it is an act that jeopardises the fundamental computer network security goals of confidentiality, integrity, and privacy. Intrusion detection is the process of monitoring and analysing events in a computer system or network for signals of potential threats and violations of computer security practices, acceptable user regulations, or standard security policies. The Intrusion Detection System (IDS) detects intrusions in the network.

It is intended to monitor events that occur in a computer system or network and respond to occurrences that show signs of potential incidents of security policy breaches.

On the other hand, an intrusion prevention system (IPS) is a network security system or technology that is capable of not only detecting but also preventing invasion actions.

III. THREATS AND VULNERABILITIES

There are several forms of faults that can occur in a system, allowing an attacker to obtain access to the system.

A. Input or Access Validation Error

Buffer overflows and boundary condition errors are the two most prevalent types of input validation problems. Both of these arise when the software fails to properly check the input from the system. A buffer overflow attack takes advantage of the limits of a buffer, resulting in data overwriting neighbouring memory addresses. A boundary condition error happens when input to a programme leads it to exceed some limit. For example, the input might lead the machine to run out of memory. Access error occurs when the system's access control policy is faulty. As a result, the attacker can use this to obtain control of the system.

© IJRASET: All Rights are Reserved |

B. Exceptional Condition Handling Error

The system becomes weak because an exception has occurred. This exception may have gone unnoticed or been handled improperly, allowing the attacker to exploit the system. Configuration Error can also occur as a result of the end administrator's failure to configure the system. This is not the fault of the system's architecture, but of the user who mistakenly set it.

C. Race Condition Error

A race condition is a fault in a system in which the output is unexpectedly dependent on the timing of occurrences. Attackers can use race circumstances to launch denial-of-service attacks. In addition, attackers can exploit programmes that achieve race situations while in a privileged state. While in this privileged position, the attacker may induce the system to do illegal tasks.

IV. COUNTERMEASURES TO NETWORK INTRUSION

Snort is an open-source signature-based intrusion detection system that will either discard or allow packets to arrive on a certain interface based on the rules you have configured. is an open-source intrusion detection system. The power of this tool is in the creation of rules. Rules can be written to restrict traffic or just issue alarms using notifications which can be stored in a log file, forwarded to the console, or shown on the screen.

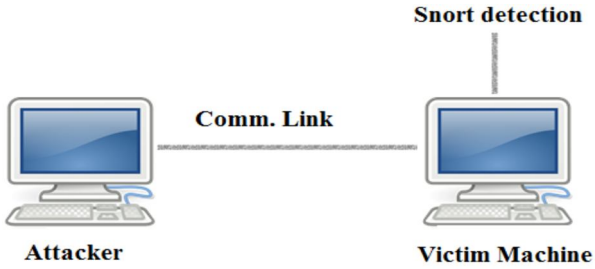


Fig. 1 Snort Tool Detection Tool

Here in this research, we have set up an architecture where the attacker tries to attack a victim machine. Snort is installed on the victim machine which helps in detecting the attack launched on it by the attacker based on the configuration and snort intrusion detection rules.

A. Snort has Two Modes of Operation

Packet Sniffing from the command line simply examines header information and prints the results to the screen. It may also be used as a packet logger mode, logging each packet into log files located in the root directory. The file can later be inspected with Snort or tcpdump. This state is for later use, such as whether someone wishes to inspect the collected packets. This mode is the real use of snort; in this mode, snort monitors the traffic and blocks any undesired traffic using the rules. The other mode is *The Network Intrusion Detection System mode (NIDS mode)* is the most significant mode when it comes to intrusion detection. Snort as a NIDS mode, employs its rules to detect intrusion activity on the network. Snort can perform real-time packet recording, content searching/matching, and protocol analysis, as well as to detect a series of attacks. It not only detects or identifies intrusions, but it may also prevent them by taking actions such as reject, drop, and block.

V. SIMULATION ENVIRONMENT AND REAL-TIME ANALYSIS

A. Configuration of Snort Tool

Snort installation from source code consists of many steps: obtaining the code, configuring it, compiling it, and finally installing it. The rules are created by the user in a text file that is connected to the *snort.conf* file, which contains all of the snort configurations. After downloading, we must set up the obtained code. The instructions for configuring Snort are shown below in a snapshot Fig. 2

```
cd ~/snort
wget https://www.snort.org/downloads/snort/snort-2.9.8.3.tar.gz
tar -xvzf snort-2.9.8.3.tar.gz cd snort-2.9.8.3
./configure
make
sudo make install
```

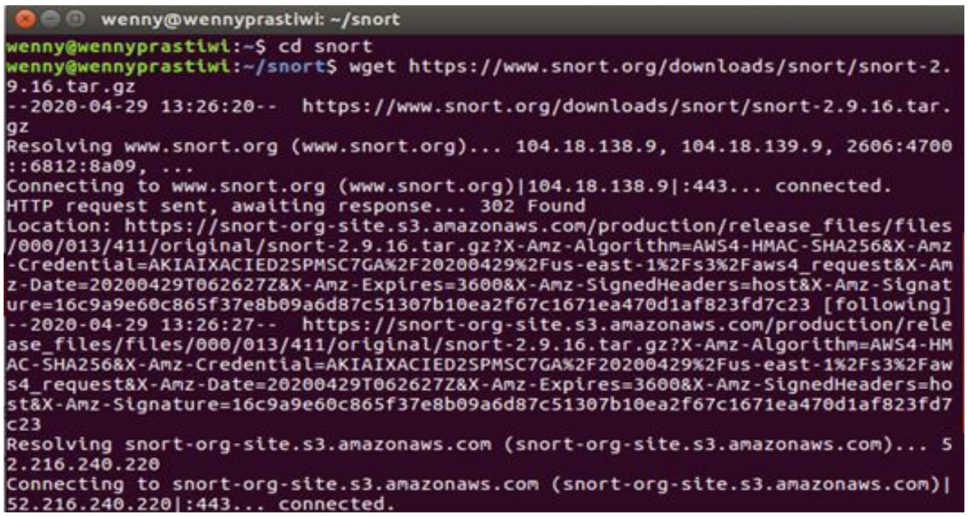


Fig. 2.1 Opening Snort config file

To configure the snort tool we write the following code and instructions in snort.conf file in the code editor and set all the variables in the config code such as HOME_NET IP address and router mac address according to Fig. 2.2 and Fig. 2.3

```
#--------------------------------------------------
#   VRT Rule Packages Snort.conf
#
#   For more information visit us at:
#     http://www.snort.org                   Snort Website
#     http://vrt-blog.snort.org/    Sourcefire VRT Blog
#
#     Mailing list Contact:      snort-users@lists.snort.org
#     False Positive reports:    fp@sourcefire.com
#     Snort bugs:                bugs@snort.org
#
#     Compatible with Snort Versions:
#     VERSIONS : 2.9.16.1
#
#     Snort build options:
#     OPTIONS : --enable-gre --enable-mpls --enable-targetbased --enable-ppm --enable-
perfprofiling --enable-zlib --enable-active-response --enable-normalizer --enable-reload --
enable-react --enable-flexresp3
#
#     Additional information:
#     This configuration file enables active response, to run snort in
#     test mode -T you are required to supply an interface -i <interface>
#     or test mode will fail to fully validate the configuration and
#     exit with a FATAL error
#--------------------------------------------------

###################################################
# Step #1: Set the network variables.  For more information, see README.variables
###################################################

# Setup the network addresses you are protecting
ipvar HOME_NET 192.168.1.0/24

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET !$HOME_NET

# List of DNS servers on your network
ipvar DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
ipvar SMTP_SERVERS $HOME_NET
```

Fig. 2.2 Snort config file

```
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules
var RULE_PATH c:\Snort\rules
# var SO_RULE_PATH ../so_rules
var PREPROC_RULE_PATH c:\Snort\preproc_rules

# If you are using reputation preprocessor set these
# Currently there is a bug with relative paths, they are relative to where snort is
# not relative to snort.conf like the above variables
# This is completely inconsistent with how other vars work, BUG 89986
# Set the absolute path appropriately
var WHITE_LIST_PATH c:\Snort\rules
var BLACK_LIST_PATH c:\Snort\rules

##################################################
# Step #2: Configure the decoder.  For more information, see README.decode
##################################################

# Stop generic decode events:
config disable_decode_alerts

# Stop Alerts on experimental TCP options
config disable_tcpopt_experimental_alerts

# Stop Alerts on obsolete TCP options
config disable_tcpopt_obsolete_alerts

# Stop Alerts on T/TCP alerts
config disable_tcpopt_ttcp_alerts

# Stop Alerts on all other TCPOption type events:
config disable_tcpopt_alerts

# Stop Alerts on invalid ip options
config disable_ipopt_alerts

# Alert if value in length field (IP, TCP, UDP) is greater th elength of the packet
# config enable_decode_oversized_alerts

# Same as above, but drop packet if in Inline mode (requires enable_decode_oversized_alerts)
# config enable_decode_oversized_drops

# Configure IP / TCP checksum mode
config checksum_mode: all
```

Fig. 2.3 Snort config file

In the configuration code, we included rules to detect and alert the victim computer against ddos, dns, dos, smtp, icmpt, tcp, virus, backdoor, bad traffic, server-mssql, tftp, scada, scan, imap, exploit, p2p, malware-backdoor, browser web-kit attacks etc.

B. Attack and Alert on Victim Monitor

After executing the rules without any error, start the snort tool to detect any intrusion. On the other side on Kali Linux, start a DOS Attack on the windows machine using hping3 TCP flooding. We see that TCP flooding has been identified by the snort and an alert message has been given with each packet request. To perform the experiment we hacked our own router which disconnected all the devices connected to that and at the same time the windows PC running snort on it sent us a "Bad Intrusion Detected!" message along with providing the IP that is pinging our router continuously. This made our computer safe from such attacks.

```
10/18-04:32:25.438523  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 216.58.203.163:443 -> 192.168.1.7:51023
10/18-04:32:25.438524  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 216.58.203.163:443 -> 192.168.1.7:51023
10/18-04:32:25.439180  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 216.58.203.163:443 -> 192.168.1.7:51023
10/18-04:32:25.440042  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 192.168.1.7:51023 -> 216.58.203.163:443
10/18-04:32:25.440670  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 192.168.1.7:51023 -> 216.58.203.163:443
10/18-04:32:25.440949  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 192.168.1.7:51023 -> 216.58.203.163:443
10/18-04:32:25.525167  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 216.58.203.163:443 -> 192.168.1.7:51023
10/18-04:32:25.525410  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 192.168.1.7:51023 -> 216.58.203.163:443
10/18-04:32:25.525879  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 216.58.203.163:443 -> 192.168.1.7:51023
10/18-04:32:25.526132  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 192.168.1.7:51023 -> 216.58.203.163:443
10/18-04:32:25.530303  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 216.58.203.163:443 -> 192.168.1.7:51023
10/18-04:32:25.531645  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 216.58.203.163:443 -> 192.168.1.7:51023
10/18-04:32:25.531872  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 192.168.1.7:51023 -> 216.58.203.163:443
10/18-04:32:25.590237  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 216.58.203.163:443 -> 192.168.1.7:51023
10/18-04:32:25.590237  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 216.58.203.163:443 -> 192.168.1.7:51023
10/18-04:32:25.590521  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 192.168.1.7:51023 -> 216.58.203.163:443
10/18-04:32:25.616213  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 192.168.1.7:51023 -> 216.58.203.163:443
10/18-04:32:25.650108  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 216.58.203.163:443 -> 192.168.1.7:51023
10/18-04:32:25.650108  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 216.58.203.163:443 -> 192.168.1.7:51023
10/18-04:32:25.650472  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 192.168.1.7:51023 -> 216.58.203.163:443
10/18-04:32:25.677181  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 192.168.1.7:51023 -> 216.58.203.163:443
10/18-04:32:25.689949  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 216.58.203.163:443 -> 192.168.1.7:51023
10/18-04:32:25.749615  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 216.58.203.163:443 -> 192.168.1.7:51023
10/18-04:32:28.736449  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 172.217.194.189:443 -> 192.168.1.7:64754
10/18-04:32:28.763595  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 192.168.1.7:64754 -> 172.217.194.189:443
10/18-04:32:30.292840  [**] [1:10001:0] Bad intrusion detecting! [**] [Priority: 0] {UDP} 192.168.1.7:56549 -> 104.18.4.27:443
```

Fig. 3 Victim Console Alerts

VI. CONCLUSIONS

After familiarisation with IDS and its categories, multiple Snort-based Intrusion Detection strategies are reviewed and realistically conducted in this project to maintain an institution's security against threats. With significant limitations, Snort-based IDPS utilising efficient rules, Honeypot, Hardware-assisted method, Neural Networks, and Multi-Sensors-like approaches may prevent from basic intrusions to serious DoS and DDoS type assaults in high speed and Cloud settings. This paper's main features include IDPS-related technologies and processes, as well as how SNORT may help in the whole process. The detection rate and false-positive rates of IDPs determine their performance. Nonetheless, IDPs appear to be constantly changing in order to continue to guard against the most recent and sophisticated security threats. There are several ways to improve the effectiveness of a Snort-based intrusion detection and prevention system.

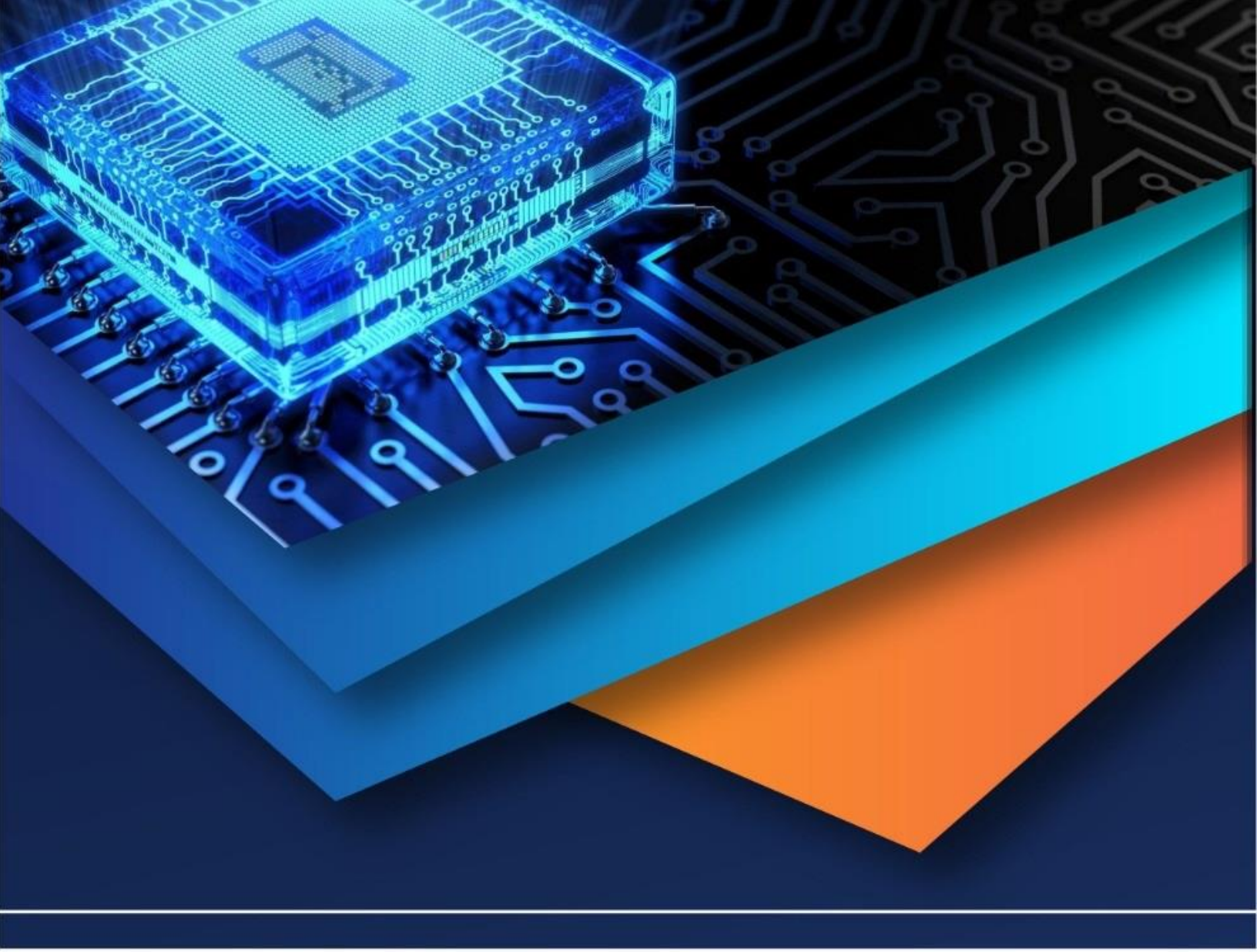
VII. ACKNOWLEDGMENT

We would like to thank Prof. Lavanya K for her valuable feedback. We would like to acknowledge all the efforts she has put in in our lab classes to teach us various information security concepts which we finally applied in our project for information security analysis and audit course in VIT, Vellore.

REFERENCES

[1] F. Gong. Next generation intrusion detection systems (IDS). McAfee Network Security Technologies Group, 2002.
[2] K. Sengaphay, S. Saiyod and N. Benjamas, "Creating Snort-IDS Rules for Detection Behavior Using Multi- sensors in Private Cloud", Lecture Notes in Electrical Engineering, pp. 589-601, 2016.
[3] Z. Chiba, N. Abghour, K. Moussaid, A. omri and M. Rida, "A Cooperative and Hybrid Network Intrusion Detection Framework in Cloud Computing Based on Snort and Optimized Back Propagation Neural Network", Procedia Computer Science, vol. 83, pp. 1200-1206, 2016.
[4] W. Park and S. Ahn, "Performance Comparison and Detection Analysis in Snort and Suricata Environment", Wireless Pers Commun, 2016.
[5] S. Patel and A. Sonker, "Rule-Based Network Intrusion Detection System for Port Scanning with Efficient Port Scan Detection Rules Using Snort", International Journal of Future Generation Communication and Networking, vol. 9, no. 6, pp. 339-350, 2016.
[6] A. Sagala, "Automatic SNORT IDS rule generation based on honeypot log", 2015 7th International Conference on Information Technology and Electrical Engineering (ICITEE), pp. 576-580, 2015.
[7] G. Ahmed, M. Khan and M. Bashir, "A Linux-based IDPS using Snort", Computer Fraud & Security, vol. 2015, no. 8, pp. 13-18, 2015.

© IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)