Signature-Based IDS for Software-Defined Networking

Pavithra H, Abhishek A, Jayachandra M, Punithraj M N, Umakanth H P

***Abstract*--- Software-Defined Network is rising field in networks that guarantee to alter the approach of building, designing, and operating network architecture.SDN allows us to transform from ancient network design of proprietary based to free and programming network design.[1] However SDN is an new and improving technology comes with existing security threats as well new security threats. The vulnerabilities in the network will become additional receptive intruders. The main focus is presently shifted from multi purpose to one purpose of failure where the SDN controller could also be a main target. Once the SDN controller is been compromised the entire network will get compromised [1][2].**

The reconciliation of IDS into the Software defined network configuration is significant to delivering a system with assault step. Signature-based IDS (snort) is installed in the network for activity perception and assault recognition, by, reflecting the movement bound to the servers. However there area unit have few exception that the suggestion is created for attainable mitigation. So as to produce ascendable threat detection within the design, A flow-based detection model is been developed and enforced with machine learning technique to detect the anomaly and to overcome the limitation of signature-based IDS.[2]

The IDS developed is expected to detect and respond to the attacks on the networks. The signature database is updated to prevent further attacks. The IDS is also not adding any considerable amount of overhead to system performance. [4]

Index Terms--Software-defined Network; Intrusion Detection System; OpenFlow; Machine Learning; Neural Network;

Prof.Pavithra H, Computer Science and Engineering, RV College of Engineering, (email: pavithrah@rvce.edu.in), Bangalore, India
Abhishek A ,Computer Science and Engineering, RV College of Engineering, (email:abhishek1996achar@gmail. com), Bangalore, India
Punithraj MN, Computer Science and Engineering, RV College of Engineering, (email:punithraj14@gmail.com), Bangalore ,India
Umakanth HP, Computer Science and Engineering, RV College of Engineering ,(email: umakanth412@gmail.com), Bangalore, India
Jayachandra M, Computer Science and Engineering, RV College of Engineering , (email:m7jayachandra@gmail.com) ,Bangalore, India

I. INTRODUCTION

Software- Defined Network is rising field in networks that guarantee to alter the approach of building, designing, and operating network architecture.SDN allows us to transform from ancient network design of proprietary based to free and programming network design. which will provide more options to users.

Designing of the SDN architecture considers some of the security issues but, the SDN architecture still have some security issues that is needed to be fixed. Some of the security issues are directly from the ancient network environment, while some are more specific SDN architecture[1].

SDN architecture contains the security issues involved in the network environment, but the partition of the control plane from information plane brings an alternate type of security danger to the SDN design, which may found in these three layers: application layer, control layer, and infrastructure layer. The application layer is the layer where the applications are deployed. Control layer deals with the deciding the routing of the network packets. The infrastructure layer enables packet forwarding.

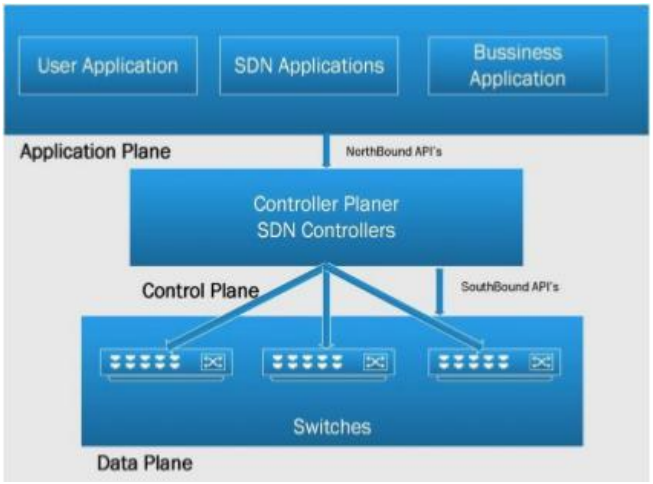


Fig 1.1 SDN Architecture

As a result of this security issue can lead to data modifications, unauthorized access to the network, data leakage, denial of service (DoS) [2]. The SDN architecture introduces the centralized control which will leads to the different attacks. The intruders who is trying to get access to the SDN controller will be identified by our Software. Once the SDN controller is compromised the intruder can

change the rules in the devices and deny a legitimate user access to the available resources (DoS attack. There are other attacks like man-in-the-middle(MITM), and side-channel, ARP poisoning, port scans, vulnerability scans etc. [3].

In order get a secure SDN environment, developing an intrusion detection system (IDS) for SDN architecture is one of the best approaches. Intrusion Detection System is a system developed only to identify and alert unauthorized access attempts, changes, or/and restricts computer system resources. The system is typically developed to identify malicious traffics and attacks against the single host computer or a network [4][5].

II. BACKGROUND/LITERATURE SURVEY

The architecture of the sdn allows us to implement a number of different new security concepts which were not possible in a non-sdn network. The work of [1] addresses the flow stat collection, feature extraction and aggregation of the network packets. But it fails to adapt to real-time scenarios. The work of [2] covers the line switch checking a packet sent by the attacker using the bernoulli trial. It also focuses on the opportunistic proxying and packetin rate monitoring. The limitation of this paper is sometimes it denies the access even to the legitimate users. The work of
[3] focuses on the weakness in the current techniques and their mitigation using more effective defense mechanisms. But in doing so it over empowers the switches. The work of
[4] concentrates on how sdn can bring benefits to security, however, the centralized control system brings its own new challenges. The work of [5] signifies the recent trends in sdn securities and the work of [6] gives the different threats and vulnerabilities in sdn. The work of [7] concentrates on the signature-based ids for sdn to develop a self-improving security mechanism. But it ignores the threats not detected by the signature based ids. The work of [8] gives the language abstraction for the sdn and virtualization using openflow model.

III. METHODOLOGY

In order to create host-system, servers with ODL, the Mininet network simulator is installed and configured for SDN.

Snort IDS is installed and configured to monitor the network traffic and to detect attacks intrusion by means of NIDS.

The flow statistics are collected using Wireshark which will be used to identify abnormality behaviour in the flow.

An alert or a signal is given to the user or the administrator.

Back-propagation algorithm is used to train the network [10].

IV. RESULTS

Curve Fitting and Time Series are the other neutral network types which are used to evaluate the performance of IDS model. As shown in the below result table, Performance accuracy of identifying abnormality with 97.3% detection rate is achieved by Pattern Recognition. At the beginning Fitting Curve has less performance.But performance in detection accuracy is improved with 89.5% with weight initialization and re-training [11][14]. Moreover, Time Series Neural Network method takes longer time for training, as it recorded the poorest result. Retraining is very difficult task in this case .It takes at least half an hour to complete training, hence retraining for several times is challenging and tedious task.

Figure. 4.1: Shell access Output

Hping is a command-line orientated TCP/IP packet assembler/analyzer. The interface is galvanized to the ping (8) UNIX system command, however, Hping isn't solely ready to send ICMP echo requests (Figure 4.1).Hping supports many other protocols such as UDP ,TCP, ICMP and RAW-IP protocols, incorporates a traceroute mode, the power to send files between a coated channel. The above shows that Hping attack is detected by snort in SDN [10].

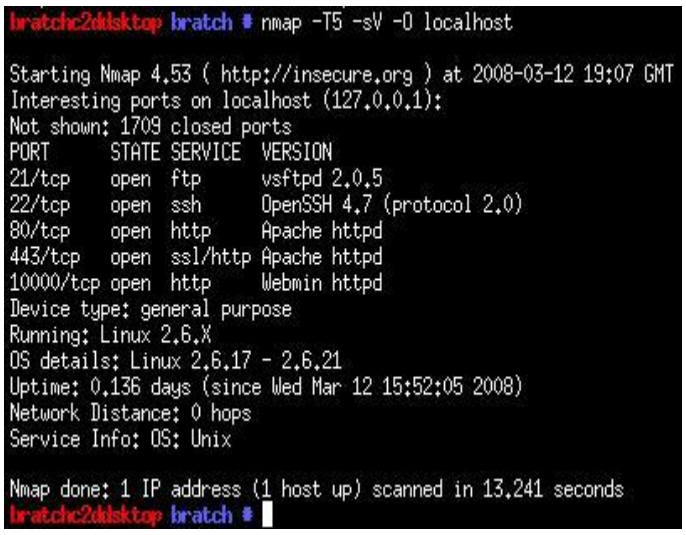
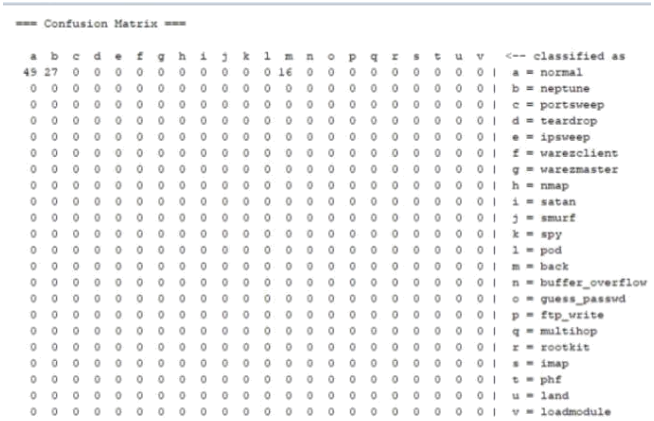


Figure 4.2: Port Scan output

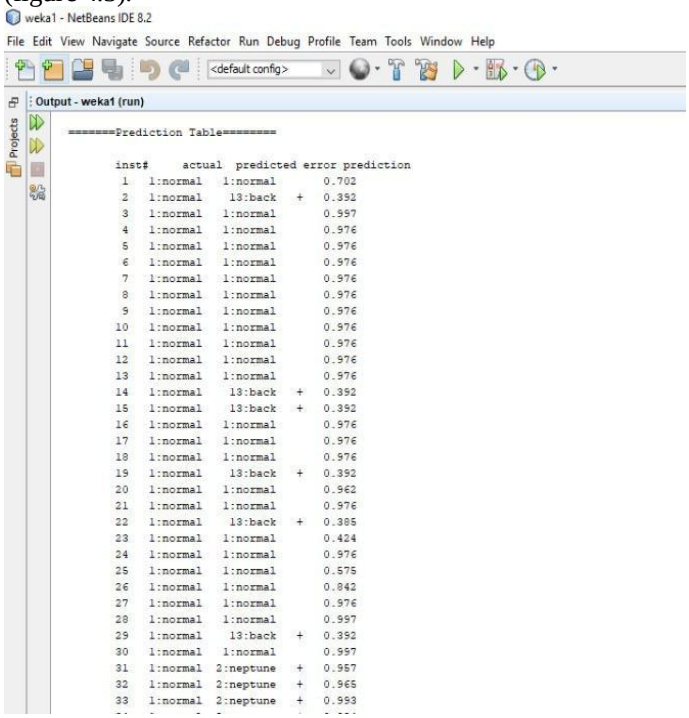


```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   k   l   m   n   o   p   q   r   s   t   u   v   <-- classified as
  49  27   0   0   0   0   0   0   0   0   0   0  16   0   0   0   0   0   0   0   0   0 |  a = normal
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  b = neptune
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  c = portsweep
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  d = teardrop
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  e = ipsweep
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  f = warezclient
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  g = warezmaster
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  h = nmap
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  i = satan
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  j = smurf
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  k = spy
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  l = pod
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  m = back
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  n = buffer_overflow
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  o = guess_passwd
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  p = ftp_write
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  q = multihop
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  r = rootkit
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  s = imap
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  t = phf
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  u = land
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 |  v = loadmodule
```

Figure. 4.3: Classification Output

An anomaly-based IDS is an intrusion detection system for detecting both network and computer intrusions and misuse by tracking system activity and classifying it as either normal behavior or anomaly behavior. The classification is based on rules, rather than signatures or patterns, and attempts to detect different type of network intrusion that failure the normal system operation. The above screenshot shows Attack class is classified by the anomaly model (figure 4.3).



```
weka1 - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Output - weka1 (run)
=======Prediction Table========

      inst#     actual  predicted error prediction
         1  1:normal   1:normal       0.702
         2  1:normal    13:back   +   0.392
         3  1:normal   1:normal       0.597
         4  1:normal   1:normal       0.976
         5  1:normal   1:normal       0.976
         6  1:normal   1:normal       0.976
         7  1:normal   1:normal       0.976
         8  1:normal   1:normal       0.976
         9  1:normal   1:normal       0.976
        10  1:normal   1:normal       0.976
        11  1:normal   1:normal       0.976
        12  1:normal   1:normal       0.976
        13  1:normal   1:normal       0.976
        14  1:normal    13:back   +   0.392
        15  1:normal    13:back   +   0.392
        16  1:normal   1:normal       0.976
        17  1:normal   1:normal       0.976
        18  1:normal   1:normal       0.976
        19  1:normal    13:back   +   0.392
        20  1:normal   1:normal       0.962
        21  1:normal   1:normal       0.976
        22  1:normal    13:back   +   0.385
        23  1:normal   1:normal       0.424
        24  1:normal   1:normal       0.976
        25  1:normal   1:normal       0.575
        26  1:normal   1:normal       0.842
        27  1:normal   1:normal       0.976
        28  1:normal   1:normal       0.597
        29  1:normal    13:back   +   0.392
        30  1:normal   1:normal       0.597
        31  1:normal  2:neptune   +   0.957
        32  1:normal  2:neptune   +   0.965
        33  1:normal  2:neptune   +   0.993
```

Figure. 4.4: Error Prediction Output

(Prediction table showing predicted attack and error rate)

The above figure 4.4 shows the error prediction done in the program to identify the possible false identifications.

V. KEY CHALLENGES

5.1 Integration of Components

A major challenge is the input format and flow of data to a signature and anomaly detection system. In the signature-based method only matched signature is alerted and new attacks are not identified so all new type of attacks is detected by anomaly system [5]. To cope with this challenge, we used some JavaScript and bro framework which will classify attribute in the packet flow and helps in conversion to KKD format.

Another challenge is listening and monitoring many interfaces and we used port mirroring in order to listen to many interfaces in snort [9].

5.2 Synchronization between Services

There are significant challenges with signature and anomaly-based detection system in real time environments and in this thesis, we propose effective and efficient techniques to address these challenges [12]. A major challenge is a real-time capturing of traffic and inputting to both system because in a signature- based system(snort) will log only matched signature but for anomaly detection flow of the packets in KDD format is required [13][14]. To cope with this challenge, we used real -time traffic capturing tool(Wireshark) and JavaScript for the conversion of flow of packet to KDD format.

VI. CONCLUSION

Software-Defined Network is an emerging field in networks that guarantee to alter the approach of building, designing, and operating network architecture. Due to the numerous advantage of this architecture, many companies are shifting from the traditional network architecture to new SDN architecture.[3] However SDN is an new and improving technology comes with existing security threats as well new security threats which will threatens the future of Software defined network. The vulnerabilities in the network will become additional receptive intruders. Our project present machine learning approach to detect intrusion detection in the Software defined network.[9] A flow-based detection model is been developed and enforced with machine learning technique to detect the anomaly in Software defined network.The Pattern Recognition method is been used in this project to compare accuracy rate with other neural network model [14].

VII. REFERENCES

[1] N. Feamster, J. Rexford, and E. Zegura, "The Road to SDN: An Intellectual History of Programmable Networks," Tech. Rep.,2013.

[2] S. Sezer, S. Scott-Hayward, P. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we Ready for SDN? Implementation Challenges for

Software-Defined Networks," Communications Magazine, IEEE, vol. 51, no. 7, pp. 36–43, 2013.
[3] Software-Defined Networking: The New Norm for Networks ONF White Paper April 13, 2012.

[4] ETSI 'Software-aware and Management-aware SDN" - 3rd ETSI Future Networks Workshop 9-11 April 2013 - http://docbox.etsi.org/Workshop/2013/201304_FNTWORKSHOP/eproceedings_FNT_2013.pdf.

[5] A. Manzalini and R. Saracco, "Software Networks at the Edge: a shift of paradigm", Proc. of IEEE SDN4FNS'13, Trento, Italy, Nov. 2013.

[6] http://www.sdncentral.com/comprehensive-list-of-open-source-sdn-projects.
[7] IETF WG, "Path Computation Element," http://datatracker.ietf.org/wg/pce/charter.
[8] S. Shin, P. Porras, V. Yegneswaran, and G. Gu, "A framework for integrating security services into software-defined networks," in Proc. ONS, 2013, pp. 1–2.
[9] S. Schechter, J. Jung, and A. Berger, "Fast detection of scanning worm infections," in Recent Advances in Intrusion Detection (RAID'04), ser. Lecture Notes in Computer Science, E. Jonsson, A. Valdes, and M. Almgren, Eds. Springer Berlin Heidelberg, 2004, vol. 3224, pp. 59– 81

[10] J. Twycross and M. M. Williamson, "Implementing and testing a virus throttle," in Proceedings of the 12th USENIX Security Symposium (SSYM'03), vol. 12. Berkeley, CA, USA: USENIX Association, 2003, pp.
[11] T. D. Nadeau and K. Grey, Centralized and distributed control and data planes, in SON: Software Defined Networks, 1st Ed, Sebastopol: OReilly Media, Inc, 20l3, pp. 9-44.
[12] Z. Jadidi, V. Muthukkumarasamy, E. Sithirasenan, and M. Sheikhan, "Flow-based anomaly detection using neural network optimized with GSA algorithm," in 2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops, 2013, pp. 76–81.
[13] P. Van Trung, T. T. Huong, D. Van Tuyen, D. M. Duc, N. H. Thanh, and A. Marshall, "A multi-criteria-based DDoS-attack prevention solution using software defined networking," in Advanced Technologies for Communications (ATC), 2015 International Conference on. IEEE, 2015, pp. 308–313.
[14] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A Survey of Security in Software Defined Networks," Communications Surveys & Tutorials, IEEE, vol. pp. 1-1, 2015.