International Journal of Computer Science and Informatics

Volume 2 | Issue 2 Article 7

October 2012

Approaches for improving the performance of snort Intrusion Detection Systems

V.P. Kshirsagar
Dept.of Computer Science & Engg., Govt. College of Engg., An Autonomous Institute Aurangabad (M.S.) IN, vkshirsagar@gmail.com

S.S. Vishnu
Dept.of Computer Science & Engg., Govt. College of Engg., An Autonomous Institute Aurangabad (M.S.) IN, swativishnu@gmail.com

S.M. Tidke
Dept.of Computer Science & Engg., Govt. College of Engg., An Autonomous Institute Aurangabad (M.S.) IN, sonalikothri@gmail.com

Recommended Citation

Kshirsagar, V.P.; Vishnu, S.S.; and Tidke, S.M. (2012) "Approaches for improving the performance of snort Intrusion Detection Systems," *International Journal of Computer Science and Informatics*: Vol. 2 : Iss. 2 , Article 7.
DOI: 10.47893/IJCSI.2012.1073
Available at: https://www.interscience.in/ijcsi/vol2/iss2/7

Approaches for improving the performance of snort Intrusion Detection Systems

V.P.Kshirsagar, S.S.Vishnu & S.M.Tidke
Dept.of Computer Science & Engg., Govt. College of Engg., An Autonomous Institute Aurangabad (M.S.) IN
E-mail : vkshirsagar@gmail.com, swativishnu@gmail.com, sonalikothri@gmail.com

Abstract **-** The process of monitoring the events occurring in a computer system or network and analyzing them for sign of intrusions is known as intrusion detection systems (IDS).In this paper the architecture of the snort which is an open source Intrusion detection system is explained. It is a rule based system hence the structure of the rule is also explained. But to match with the high speed of network traffic the performance of the SNORT need to be improved hence the various methods has been developed three of them are reviewed here which are Rules Matching Algorithm Based on Dynamic Adjustment, NAPI and LASSP.

Keywords – *IDS ; SNORT ;Rule Header; Rule Option*

I. INTRODUCTION

In today's enterprise environment the security is becoming and important issue. Hackers and intruders have made many successful attempts to bring down high-profile company networks and web services. For secure communication over the network various techniques like cryptography are developed. Also the techniques for securing the network itself from attacks are being developed. Using intrusion detection methods, you can collect and use information from known types of attacks and find out if someone is trying to attack your network or particular hosts.

Snort is an open source *Network Intrusion Detection System* (NIDS) which is available free of cost.[1].To cope up with today's high network traffic which is in gigabit snort needs to be highly efficient..An intrusion detection system that fails to perform packet inspection at high rate will allow malicious packets to enter the network undetected capture and inspect packet payloads, in order to identify signatures of malicious activities. Typically, a NIDS residing on the edge of a network performs deep packet inspection on every packet that enters the protected network against several thousands of attack signatures. The signatures are represented as a set of rules which are frequently updated by the security community. When a packet is matched against a signature, an alert is raised, indicating an attempted intrusion or other misuse. For a PC-based solution, Snort can run on the top of many platforms such Linux, FreeBSD, and Windows. In practice Snort is installed more often on Linux OS platforms. Linux measures better than Windows in terms of supportability, stability, and security [3]. More importantly, Linux networking subsystem gives better performance than Windows or FreeBSD. Snort needs to be highly efficient to keep up with today's increasing traffic of link speeds of tens of Gbps. [2]

II. SNORT ARCHITECHTURE

Snort is a free open-source IDS available, which can be also integrated with third party tools. A single threaded user level application snort, utilizes the transmission control protocol TCP/IP to identify signature of malicious activities [2].

Typically, an IDS residing on the edge of a network performs deep packet inspection on every packet that enters the protected network against several thousands of attack signatures [2].

Snort consists of the following four components:

1. Packet capture/decode engine.

Snort's packet-capturing engine uses the libpcap packet-capturing library written in Lawrence Berkeley National Laboratories. Captured packets are processed by decoding engine and decoded packets become compliant with the network-level protocols.

Resultant packets are re-decoded for upper-level protocols that are TCP and UDP.

2. Preprocessor plug-ins:

Packets are passed through a number of preprocessors. This step aims investigating and processing packets before they are passed to the detection engine. Every other preprocessor examine the packets for a different attribute and make a decision to pass the packet to the detection engine without making any modification, modifying and then passing it to detection engine or not passing and generating an alert for the packet.

3. Detection engine:

Detection engine tests packets for a number of attributes stated in Snort rules definition file. Detection plug-ins provide extra detection functions.

4. Output plug-ins:

These plug-ins accept alarms generated from detection engine, preprocessors or decoding engine.[3]

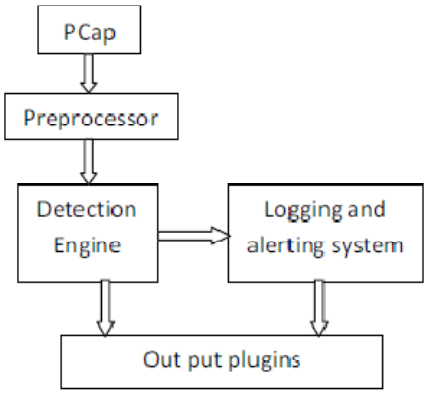

Figure 1.Snort Components

III.RULE STRUCTURE

Snort is a rule-based network intrusion detection system (N-IDS). Every rule consists of two logical parts: the rule header and rule options. It also contains criteria for matching a rule against data packets. Rule header has five sections; rule actions (action to be taken when an intrusion is detected), end-to end source and destination information (source and destination IP addresses and port numbers depending on the protocol), and direction of traffic and protocol type (TCP, UDP, or ICMP).It also has direction field that determines the source address and destination address in the rule.

Rule options consist of various conditions that help deciding whether the mentioned misuse operation has occurred or not. The action in the rule header is invoked only when all criteria in the options are true. In general, an option may have two parts: a keyword and an argument. Arguments are separated from the option keyword by a colon. A sample Snort rule is shown in Fig. 2.

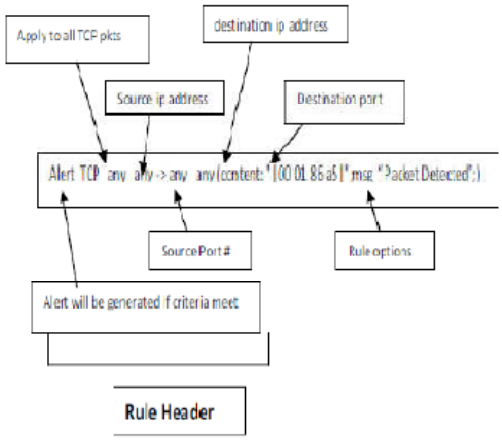

Figure 2. Structure of snort rule

IV.SNORT PERFORMANCE IMPROVEMENT

A. Rules Matching Algorithm Based on Dynamic Adjustment

Snort is based on rules search techniques for intrusion detection. As discussed earlier the rule has two parts rule header and rule option. In this method rule header refers to RTN (Rule Tree Node), which includes action, protocol, source IP address, port and some general information of data stream. Thus RTN structure is created by linking all the rules having same condition.

The second part of the rule rule option refers to OTN (Optional Tree Node), which includes specific detection flags, alerts information, content matching etc. Every matching sub function (plug-in) is inserted into FUNC linked list. Corresponding operations are to be triggered when certain condition is true. Rule linked list is classified into corresponding list according to the second keyword, such as IP, TCP, UDP and ICMP.Thus all the given rules are divided in to four trees i.e. IP, TCP, UDP and ICMP.

During the running procedure of network, the service and application system running on networks are relative stable. Network attacks present

certain general characteristics, so network traffic monitored by Snort also appears the same characteristics. If one or more packets with attack signature are detected, corresponding matching rules are to be adjusted so as to make the subsequent packets matched with related rules as soon as possible, which may effectively shorten the detection delay of Snort.

A new reference pointer is inserted in, which points to rule option and list is setup between reference pointer and rule header. Instead of searching sequentially as done normally in snort, by dynamic option reference linked list, the system is to begin search from rule option, then rule match operations are conducted upon option nodes pointed by option reference. If certain operations are matched then value of corresponding reference pointer is set to 1.This comparison continues with next rule options, this adjustment remains same until search founds another reference with high priority.

The process is summarized as below:

1. Create reference linked list of rule library
2. Add reference pointer to linked list pointing to rule option and assign reference pointer.
3. Dynamically set reference number in reference linked list.
4. Detect the attack.[4]

B. NAPI

Current versions of Linux kernel integrates a packet reception mechanism known as New API (NAPI). Upon the arrival of a packet, NAPI disables the received interrupt (RxInt) of the network device to stop the generation of further interrupts and then raises a softirq of NET_RX_SOFTIRQ type to schedule polling. Softirq is a non-urgent (or deferrable) interruptible kernel event that gets handled by *__do_softirq* kernel function. The actual polling and processing of packets take place during the execution of *net_rx_action* which is the handling function of NET_RX_SOFTIRQ softirq. Up to budget *B* packets are processed in *net_rx_action* for all network devices or interfaces, and up to quota *Q* packets are processed per interface. NAPI's polling period in Linux is not exhaustive and is designed to leave CPU processing power for other tasks in order to prevent starvation. This is accomplished using budget and handling time bounds. Budget and time basically limit the number of packets that get processed per NET_RX_SOFTIRQ handling. Pending softirqs will be processed by *__do_softirq* repeatedly up to a maximum of MAX_SOFTIRQ_RESTART (set to 10 by default). After reaching this maximum limit and if there are still pending softirqs, *do_softirq* will no longer handle softirqs. Softirqs will be handed by *ksoftirqd* which is a lower priority kernel thread with a nice value of 19. the percentage share of CPU bandwidth between user processes and NAPI (or networking subsystem) is primarily controlled by a number of key configurable parameters. These parameters include budget *B*, timer *T*, MAX_SOFTIRQ_RESTART and *ksoftirqd* priority. Out of these configurable parameters we experimentally found out that the CPU bandwidth allocated to the user processes is largely sensitive to NAPI's configurable parameter of budget *B*. the default budget of 300 gives the worst throughput, while budget 2 gives the best throughput.[2]

C. Logic Analyzer for Tweaking Snort Security and Performance

The various parameters (options) that can be used for better performance of snort are

–v: with this snort will just show the IP and TCP/UDP/ICMP headers.

-vd: It will make snort to display packet data as well as headers.

-vde: It will show additional data link layer header

-b: log the packet in tcpdump format.

These options defines mode Option Security Level (msl) and Mode Option Performance Level (mpl)

-A fast: this defines Mode performance level with alert option (MPLA).

Mode Option Security Level (msl): Level of security defined on the basis of option with which snort is running.

Mode Option Performance Level (mpl): Level of performance defined on the basis of option with which snort is running

Mode performance level with alert option (MPLA): Level of performance defined on the basis of alert option (A- fast etc) NIDS mode

LASSP architecture finds the security and performance level. It take three inputs, *first* is snort configuration file, *second* is ports/services information through *nmap* utility and *third* is running mode configuration i.e. inspection is done for every single packet.

Here based on services and other factors priorities are given to rules. Rules for currently running service are having higher priority than others. Depending on type of active number of running rules the *Priority Security Level (psl) and Priority Performance Level (psl) is defined*. If all priority rule are running security is high and performance level is low. If the configuration files, modes of snort are given as input, the priorities are

given to rules and then the system will prepare facts /predicates. These predicate will be given to prolog engine, which determines the security and performance level of the system.[5]

V. CONCLUSION

Various techniques can be used to implement IDS. This paper mainly focuses on SNORT as IDS. In the paper, we have mainly concentrated on Architecture of the SNORT and Structure of the rule used in snort. For the growing demands of the snort performance various methods are developed out of which three methods are reviewed here.

REFERENCES

[1] Intrusion Detection Systems with Snort Advanced IDS Techniques Using Snort, Apache, MySQL, PHP, and ACID by Rafeeq Ur Rehman Prentice Hall PTR

[2] K. Salah ,A. Qahtan *Department of Information and Computer Science, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia,* "Boosting Throughput of Snort NIDS Under Linux"

[3] M. Ali Aydın , A. Halim Zaim, K. Gökhan Ceylan Department of Computer Engineering, Faculty of Engineering, Istanbul University, 34320 Avcilar, Istanbul, Turkey "A hybrid intrusion detection system design for computer network security"

[4] Kuo Zhao, Jianfeng Chu, Xilong Che, Lin Lin, Liang Hu, Department of Computer Science and Technology "Improvement on Rules Matching Algorithm Based on Dynamic Adjustment"

[5] Khalid Hafeez, Muddassar Masood, Owais Malik, Zahid Anwar Department of Computing NUST School of Electrical Engineering and Computer Science Islamabad, Pakistan "LASSP: A Logic Analyzer for Tweaking Snort Security and Performance"