

Snort Performance Evaluation

¹Faeiz Alserhani, Monis Akhlaq, I. U. Awan, A. J. Cullen, J. Mellor
²Pravin Mirchandani

Abstract

Intrusion Detection Systems (IDS) are among the fastest growing technologies in computer security domain. These systems are designed to identify/ prevent any hostile intrusion into a network. Today, variety of open source and commercial IDS are available to match the users/ network requirements. It has been accepted that the open source intrusion detection systems enjoys more support over commercial systems on account of cost, flexible configuration, online support, cross platform implementation etc. Among open source IDS, Snort has secured a prominent place and has been considered as a *de facto* standard in the IDS market.

Snort is the real time packet analyser and packet logger that perform packet payload inspection by using contents matching algorithms. The system has been considered as a good alternative to expensive and heavy duty Network Intrusion Detection systems (NIDS).

This research work has focussed on analysing the performance of Snort under heavy traffic conditions. This has been done on a test bench specifically designed to replicate the current day network traffic. Snort has been evaluated on different operating systems (OS) platforms and hardware resources by subjecting it to different categories of input traffic. Attacks were also injected to determine the detection quality of the system under different conditions. Our results have identified a strong performance limitation of Snort; it was unable to withstand few hundred mega bits per second of network traffic. This has generated queries on the performance of Snort and opened a new debate on the efficacy of open source systems.

1. Introduction

Open source software has gained tremendous popularity and acceptance among academia and research community. Apart from being free of cost there are several other qualities which has made them popular. Few advantages of open source software are access to source code, detailed documentation, online forum support and rights to modify/ use. Our research has focused on a widely accepted open source software tool, Snort. Snort has received great acceptance in the IDS market and has been widely recognized as the reliable open source tool.

¹ Informatics Research Institute, University of Bradford, Bradford, BD7 1DP, United Kingdom, {f.m.f.alserhani, m.akhlaq2, i.u.awan, j.e.mellor, .j.cullen}@bradford.ac.uk

² Syphan Technologies, www.syphan.com, pmirchandani@syphan.com

Snort is capable of performing real-time traffic analysis and packet logging on the network. It performs protocol analysis and can detect variety of network threats by using content/ signature matching algorithms. Snort can be configured as a packet sniffer, packet logger and NIDS. As packet sniffer, it reads the packets off the network. In a packet logger mode, it logs packets to the storage device. NIDS mode enables the Snort to analyze the network traffic against set of defined rules in order to detect intrusion threats.

Our work has focused on the Snort capability as a NIDS. As a NIDS, Snort is composed of packet sniffer, preprocessor, detection engine and output device as shown in Figure 1. In a packet sniffer mode, it captures the network traffic and sends it to preprocessor. Snort relies on packet capturing libraries (libpcap and winpcap). The libraries act as a packet capturing interfaces to the underlying operating system. Preprocessor takes the raw packets, checks them against certain types of behavior (plug-in) for example HTTP plug-in etc. Once the packet has been observed to have a particular behavior, it is forwarded to detection engine otherwise considered as valid. The facility of plug-in allows a user to configure the application according to usage/ traffic flow. It also prevents the unnecessary processing of all incoming packets which may cause an additional burden on the system. Detection engine is the main component of a network intrusion detection system. It receives data from the preprocessors and checks it against certain set of rules. The rules are the set of requirements that generate an alert on fulfillment for eg; observing certain strings of characters in a file, observing known pattern of attacks etc. Snort allows the users to write their own rules on foreseeable threats/ network characteristics. It also supports the users to have an access to the community provided rules. In addition, registered users are also allowed to download Sourcefire VDB rules available on Snort website [1]. In the detection engine, if the data matches a rule, an alert is generated and sent to a log file via network connection [2].

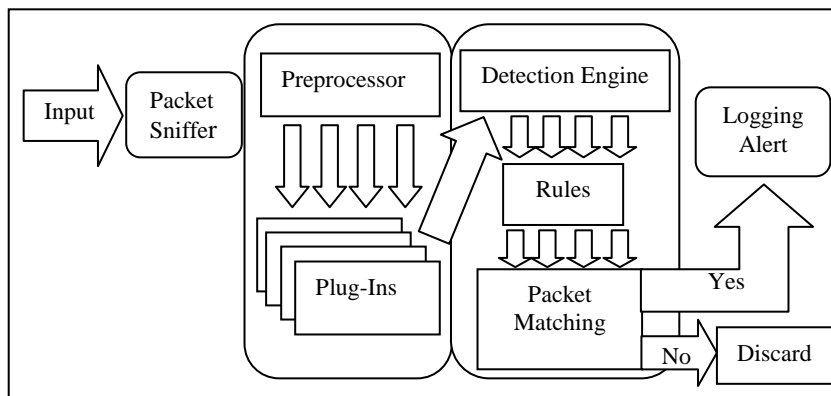


Figure 1: Snort Architecture

Our research has focused on evaluating the performance of Snort under different hardware resources by subjecting it to various traffic conditions.

Snort has been run on different OS platforms and similarly attacks were also generated from variable platforms in order to determine the response of the system. We have tried to accumulate the response of the system with a view to analyze its performance and efficacy.

Our effort has addressed the scalability issues associated with open source NIDS (Snort). Our results are based on the packet handling capacity of the system and its abilities to generate alerts on known attack signatures. We have found a strong performance limitation of Snort, the system starts dropping packet once subjected to heavy input traffic. This has also caused a major performance bottleneck in which known attacks are not being detected. The evaluation statistics is unique in terms of quality of test bench, methodology and in depth analysis. Our results can be considered as a bench mark for the improvement in performance of pattern matching NIDS.

2. Performance Test

2.1 Test-Bench

The network is built around ³ProCurve Series 2900 Switch [3]. All machines are Dell Precision T3400, Intel Core2Quad, Q6600, 2.40 GHz, 2 GB RAM, PCIe, 1.0 Gbps, Network Card (Broadcom NetXtremo Gigabit Ethernet) [4].

- **Snort Machines – Host Configuration.** Snort has been installed on Windows XP SP2 [5] and Linux 2.6 [6] platform as shown in Figure 2.

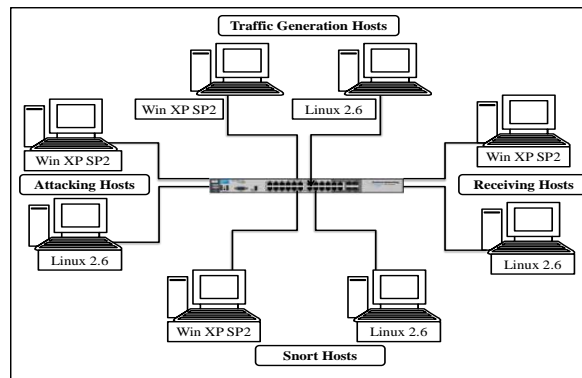


Figure 2: Test-Bench Snort Performance Evaluation - Host Platform

- **Snort Machines – Virtual Configuration.** Virtual platform has been built using VMWare Server [7] over Windows 2008 Server [8] and two virtual

³ ProCurve Series 2900, 10 Gbps with 24x1 Gbps ports and 2x10 Gbps.

machines-Windows XP SP2 and Linux 2.6 were created as shown in Figure 3.

- **Traffic Generation and Reception Hosts.** Two machines each were configured to generate and receive traffic on Windows XP SP2 and Linux 2.6 platforms respectively. We have used open source network traffic generators, D-ITG [9] on Linux 2.6 platform and LAN Traffic Version 2 [10] on Windows XP SP 2 platform.

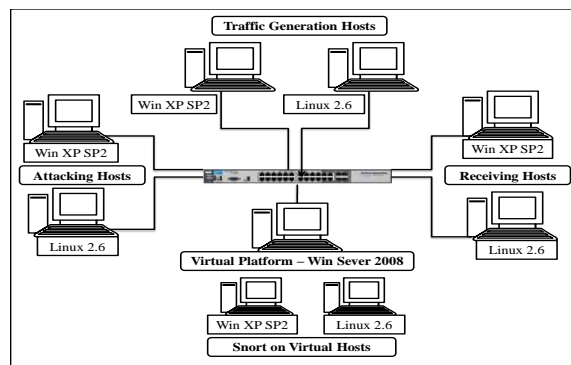


Figure 3: Test-Bench Snort Performance Evaluation - Virtual Platform

- **Attacking Hosts.** Metasploit framework [11] has been used on attacking host with Windows XP SP2 and Linux 2.6 platforms respectively.

2.2 Evaluation Methodology

In order to ascertain the capability of Snort, we proceeded as follows:

- Generating different categories of traffic load: 100 Mbps, 250 Mbps, 500 Mbps, 750 Mbps and 1.0 Gbps respectively.
- Injecting known attack signatures from similar OS platforms as of Snort under various traffic conditions.
- Injecting known attack signatures from different OS platforms as of Snort under various traffic conditions.
- Observing packet losses at Snort under various traffic conditions.

3 Results

3.1 Alert Generation

Performance of Snort was observed in relation to generating alerts on known attack signatures as follows:

3.1.1 Snort and Attacker on different OS Platforms.

Snort was evaluated on two different scenarios, host machine and virtual environment.

- **Snort on host machine.** Snort was configured on Windows XP SP2 host and attacks were generated from Linux 2.6 using Metasploit framework. The results obtained as shown in Figure 4 have identified decline in the performance of Snort once subjected to heavy traffic conditions for example at 1.0 Gbps of network traffic the system could detect only 20% of the attacks.
- **Snort on virtual environment.** Snort was configured on performance limiting environment of Windows XP SP2 and Linux 2.6 virtual machines respectively. The performance of Snort was found quite similar to that of host configuration; this is because packets analyzed in virtual configuration are quite less than the packets analyzed in host scenario. The results obtained (shown in Figure 4) have not accounted the packet losses due to virtual bottleneck.

3.1.2 Snort and Attacker on similar OS Platforms.

- **Snort on host machine.** Performance of Snort has been evaluated in relation to OS by generating attacks from similar OS platform and observing the packet loss. Snort has shown quite good performance up to 400 Mbps of network traffic by detecting 100% attacks; however its performance declined above 500 Mbps as shown in Figure 4. At 1.0 Gbps traffic, Snort was able to capture only 30 % of the generated attacks. Overall a significant improvement has been observed in comparison to previous scenario.
- **Snort on virtual environment.** Slight improvement in the performance of Snort has been observed in comparison to virtual scenario in which attacker and Snort were on different OS platforms as shown in Figure 4.

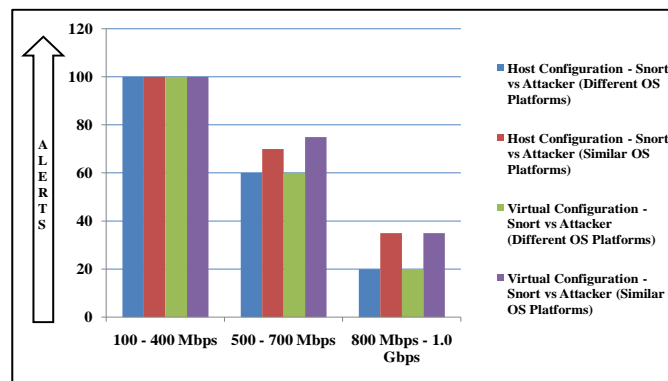


Figure 4: Alert Generation

3.2 Packet Loss

Snort performance deteriorates with the increase in network traffic; the system was unable to withstand few hundred mega bits of input traffic. While running on a different OS platform (i.e. not the same as of attacker) and for network traffic of 500 Mbps, the system dropped more than 50 % packets. This value has increased up to 75% for 1.0 Gbps of input traffic as shown in Figure 5. Performance of Snort became comparatively better in the scenario where both attacker and Snort are on similar OS platforms. The packets analyzed are comparatively higher than previous scenario as shown in Figure 6.

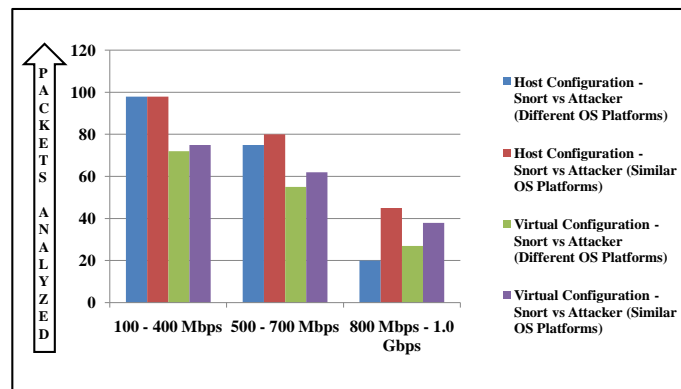


Figure 5: Packets Analyzed

In virtual scenario as shown in Figure 5, it was observed that the performance of Snort has improved. For example at 1.0 Gbps, Snort was able to analyze 38% of input traffic whereas in the similar scenario of host configuration (Figure 4) it captured only 27%. This difference is because; the numbers of packets received at virtual machines are comparatively lesser than the packets received at host configuration. The variation in packets received at virtual machine is actually an accounted packet loss incurred on the platform. Lesser number of packets received at virtual platforms shows an improved performance as shown in Figure 5; however actually it has deteriorated.

3.1.3 CPU Usage. Snort was found judicious in utilizing CPU resources in host configuration. The maximum value of CPU usage observed reached up to 30% for 1.0 Gbps of input traffic, shown in Figure 7. However; in virtual scenario CPU usage crossed over the acceptable threshold limit and Snort CPU Usage touched 80% for input traffic of 500 Mbps, shown in Figure 7. This high CPU usage in virtual scenario has caused the major performance bottleneck.

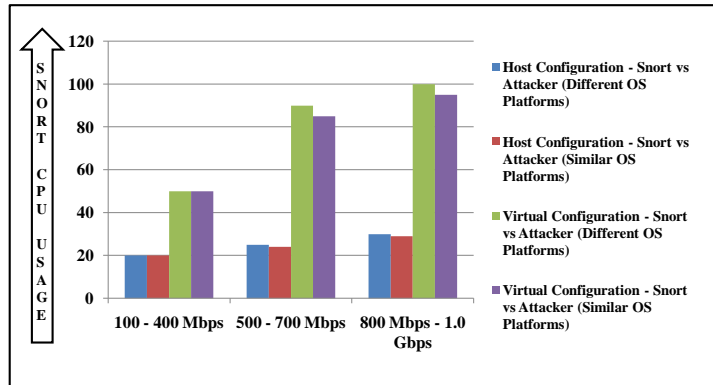


Figure 6: Snort CPU Usage

Conclusion

This research has focused on ways of determining the efficacy of the Snort in high-speed network environments. The test scenarios employed, involved the evaluation of application under different traffic conditions and observing the response of system to known attack signatures. The results obtained have shown a number of significant limitations in Snort, on both host and virtual configuration. We have confirmed that the underlying host hardware plays a prominent role in determining overall system performance. We have further shown that performance is further degraded as the number of virtual instances of NIDS is increased, irrespective of the virtual OS used. The performance of Snort was also linked to OS implementation; it performed well once the attacks were generated from the similar OS platforms and declines once the attacker is on a different OS implementation. These results shall be taken as a benchmark for improving the performance of the systems in future research works. Our future efforts include implementing techniques to prevent packet losses in NIDS under heavy traffic conditions, driving parallel logic to execute pattern matching algorithms to prevent processing overheads and implementation/evaluation of NIDS as hardware based systems.

References

- [1] Snort: www.Snort.org.
- [2] Andrew R Baker, Joel Esler, "Snort IDS and IPS Toolkit," Syngress, Canada, 2007.
- [3] ProCurve Series 2900 Switch: www.hp.com/rnd/products/switches/HP_ProCurve.
- [4] Faeiz Alserhani, Monis Akhlaq, Irfan Awan, Andrea Cullen, John Mellor and Pravin Mirchandani, "Evaluating Intrusion Detection Systems in High Speed Networks", In Press, Fifth International Conference of Information Assurance and Security (IAS 2009), IEEE Computer Society.
- [5] Windows XP SP2: www.softwarepatch.com/windows/xpsp2.html.
- [6] Linux 2.6: www.kernel.org.

- [7] VMware Server: www.vmware.com/products/server.
- [8] VMware Server: www.vmware.com/products/server.
- [9] D-ITG V 2.6: [www.grid.unina.it /Traffic/index.php](http://www.grid.unina.it/Traffic/index.php).
- [10] LAN Traffic V 2: www.topshareware.com/lan-traffic-v2/downloads/1.html.
- [11] Metasploit framework: www.metasploit.com.