International Journal of Engineering & Technology, 7(4.16) (2018) 188-191

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Comparative Study of Intrusion Detection Systems against Mainstream Network Sniffing Tools

Africa, Aaron Don[1]*, Torrizo, Lorwin Felimar[1]

[1] Department of Electronics and Communications Engineering,
De La Salle University Manila, 2401 Taft Avenue, Manila, Philippines
Corresponding author Email: aaron.africa@dlsu.edu.ph

Abstract

As the world embraces a technological revolution on how everyday devices are connected to the internet, users provide sensitive information using the internet which is broken down and distributed as packets throughout the network. Packet sniffers tap to these packets, capable of potentially compromising security and privacy of unsuspecting users. This study aims to put into the test some well-known Intrusion Detection Systems (IDS) and observe how they fare against popular packet-sniffing tools such as Wireshark and tcpdump. The varied sniffing methods and techniques from various sniffing tools will provide an evaluation of performance of the intrusion detection systems.

Keywords: *IDS, Intrusion Detection System, Packets, Sniffer*

1. Introduction

As the world embraces a technological revolution on how everyday devices are connected to the internet, users provide sensitive information using the internet which is distributed as packets throughout the network [1].
Depending on the configuration of the network, the packets travel with or without a control mechanism. As an example, when the devices are connected in hubs with no routing mechanism, packets transmitted by each device is broadcasted throughout the network [2]. Thus, as the packet is broadcast, it is up to the recipient device whether it receives or discards the packets sent to it, regardless if the packet itself is intended for it or not. In this scenario, a malicious user can set its client to receive packets not intended for them, which is the underlying concept in packet sniffing.
There are multiple methods in which packet sniffers can operate, some listed below [3,4,5]:

1. A device operating in Promiscuous mode: Promiscuous mode is defined as a mode of the network interface to accept all traffic it receives into its processing unit, even if the information is not intended for the device itself. In this manner, every information passing through the device's network interface can be read and analyzed, as long as it does pass through the device. As the sniffing method is truly passive, this type of sniffing method is hard to detect. This method is particularly useful for networks connected into hubs and for broadcasted packets, but of not much use for unicast packets and switched networks with proper packet forwarding.
2. ARP spoofing: In a switched network, the packet sent by a client flows towards the indicated gateway of the network. ARP spoofing operates in a way that the malicious user, through the packet sniffer or other tools, feeds a bogus gateway entry in the client's ARP cache which causes the client to either broadcast the packet or direct its traffic towards the packet-sniffing device. This can be prevented by setting a static ARP on all devices in the network.
3. MAC flooding: as the network switch has a limited size of its forwarding tables, there are certain scenarios where the switch cannot keep up with the amount of updating it receives from its clients. In this case, the switch enters a fail-open mode wherein it will operate as a hub, broadcasting all packets to all devices in the network. Some packet sniffers utilize this aspect of the switch by flooding the switch with MAC addresses until the network switch cannot keep up.

On the other hand, sniffing detection tools also exist to somewhat combat these packet sniffers [6]. Some of their operating principles are through pinging: as an example, as packet sniffers accept packets regardless of content, one way of detecting packet sniffers is sending a ping request with an included IP address but no MAC address [7]. The normal reaction for a client is to reject this packet since the MAC address does not match, but the sniffing devices will respond back [8,9].
Another method of detecting sniffing operation is by monitoring the transfer of packets and recognition of sequences that can be identified as sniffing. Such is the operation of Intrusion Detection Systems, IDS for short [10,11].
In this study, we aim to categorize commonly used packet sniffing tools, such as Wireshark and tcpdump and if commonly used packet monitoring tools and Intrusion Detection Systems such as Snort and Suricata detect the sniffing attempt.

2. Sniffers

A notable detail in sniffing tools is the use of a particular packet capture (pcap) library in order to operate properly. These sniffers can also be used in Hard Drives and USB [12]. These pcap libraries are considered the heart of the operation of sniffing tools, providing the capability to capture and filtering raw packetsLinux-based sniffers, libpcap is generally used, whereas most Windows-based sniffers utilize winpcap, a Windows port of libpcap.

Most of the sniffing tools utilize the network adapter's Promiscuous Mode by default, but even with this remains only capable of sniffing packets that are communicated in its respective network adapter. Thus, in order to be able to sniff packets between other devices in the switched network, it is necessary to operate a separate mechanism to aid in sniffing packets such as an ARP poisoning tool.

2.1. SmartSniff

SmartSniff [13], also known as smsniff, is a simple sniffing tool available for Windows with a workable GUI. SmartSniff provides three methods of capturing TCP/IP packets, but primarily uses winpcap for its operation. SmartSniff is capable of filtering out specific packets and can save packets data and summaries. Version 2.27 was used in this study.

2.2. Wireshark

Formerly known as Ethereal, Wireshark [14] is a well-known free and open source pocket analyzer, sporting multiple features such as filtering, statistics (with graphical display) for multiple protocols, and the ability to show the datagram in full. The software has cross-platform compatibility, able to run on Windows, Linux, and macOS. Wireshark runs in promiscuous by default and also utilizes winpcap/libpcap for its operation. This study uses the latest stable version (2.4.6) as of this paper's writing.

2.3. tcpdump/windump

tcpdump, one of the oldest packet sniffers available, is a command line network analyzer available for Linux systems, and has its counterpart in Windows called windump. tcpdump/windump [15] captures packets using libpcap/winpcap and its barebones operation uses minimal processing power. Regardless, tpcdump can provide an output file containing logs of details of packets transferred with timestamps which can be processes by the user, or even another software, if needed. This study uses Windump 3.9.5 for evaluation.

2.4. dsniff suite

dsniff [16] is a collection of various tools available for Linux for network analysis. Its various tools make it capable of intercepting/capturing packets and extracting vital information such as emails and passwords. dsniff is capable of utilizing ARP/DNS spoofing using its arpspoof and dnsspoof tools respectively and uses libpcap for its operation. dsniff v2.3 was used for this study.

2.5. Cain and Abel

Not strictly a general-purpose packet sniffer with filtering capabilities, Cain and Abel [17] is promoted primarily as a password recovery tool for Windows, sniffing through the packets travelling across the network. Most of its features involve cracking out encrypted passwords, which ranges from brute-forcing to cryptanalysis and hash calculators. It utilizes its built-in ARP poisoning tool to perform its man-in-the-middle attacks. Version 4.9.56 was used for this study.

2.6. Ettercap

Ettercap [18] is a free and open source comprehensive suite for sniffing which is primarily available for Unix systems. It promotes itself by being able to operate both actively (ARP poisoning) and passively (promiscuously), and contains several features including SSH sniffing. For this study, Ettercap version 0.8.2 was used.

2.7. ngrep

Similar in operation to tcpdump and windump, ngrep [19] is a command line network analyzer primarily available for Linux systems. One advantage it presents against tcpdump is its capability of searching regular expressions as a pattern in the packet. The latest version at this time of writing, v1.47, was used for this study.

2.8. Nmap

Network Mapper, shortened as Nmap [20], is another free and open-source security scanner primarily available for Linux but also available for Windows. This program can also scan databases [21]. Not necessarily classified as a packet sniffer but remains as a network probing tool, Nmap attempts to build a 'map' overview of the network, sending out packets and analyzes the responses which it uses to obtain information of the network such as the available hosts, their corresponding OSes, their packet filtering, etc. Nmap comes bundled with Zenmap, the GUI for general use, and version 7.70 was used for this study.

3. Intrusion Detection Systems

Sniffing detection tools are mostly classified as Intrusion Detection Systems (IDS) operating passively, as these tools are capable of monitoring network traffic and system activities for suspicious behavior but are generally not intended to prevent these activities. Due to their method operation, they are occasionally used as anti-virus and anti-Trojan detectors [22]. These detectors protect the information systems in the database [23,24].

These IDS operate by observing network traffic and detecting particular sequence of events based on a particular ruleset included in the IDS. It can be said that the rulesets themselves is the most critical aspect of Sniffer Detection, operating as the decision-maker for intrusion detection, and is the backbone of the alerting function of the IDS. If incomplete information is found Rough Set theory can be used to detect them [25,26]. Fuzzy Logic, Neural techniques and Spatial Algorithm can also be used [27, 28, 29, 30]. The rulesets are commonly obtained separately from the installation of the IDS, which can be free-of-charge or under a subscription. User-defined rules can also be provided to the IDS as part of the IDS's configurability.

Intrusion Prevention Systems (IPS) which operates in an active manner exists, which can stop or block these malicious behaviors, but is not within the scope of this study.

3.1. Snort

Snort [31] is a free and open source Intrusion Detection System, available for Windows and specific Unix systems, that can also operate as a packet sniffer and offers multiple output formats for logging. Snort uses command line interface and includes several tools and preprocessing features that aids in providing analysis of the packet transfer that it is capable of reading from the network card. This program can also be used in Open Source Web System [32].

For this study, the Windows version of Snort used is 2.9.11.1, while the utilized ruleset in conjunction is snortrules-snapshot-29111, available in the Snort website.

3.2. Suricata

As a relatively new open-source IDS initially released in 2009, Suricata [33] is available for both Windows and Unix computers.

Similar to Snort, it uses command line interface with multiple output formats for logging and can also provide inline intrusion prevention and network security monitoring. Suricata boasts of several features such as multi-thread support and hardware acceleration. The stable Windows version 4.0.4 was used for this study.
For this study, the utilized ruleset for Suricata is from Emerging Threats, a known ruleset provider for several IDS. Emerging Threats offer both free (OPEN) and subscription-based (PRO) rulesets for use, and this study uses version 8882 of the OPEN ruleset.

Table 1: Summary of Software used

Software	Version Used
smsniff	2.27
Wireshark	2.4.6
windump	3.9.5
dsniff	2.3
Cain and Abel	4.9.56
Ettercap	0.8.2
ngrep	1.47
Nmap	7.7
Snort	2.9.11.1
Suricata	4.0.4

4. Methodology

In conducting the study, the network will be managed by a TP-LINK Archer C9 v1.0 router, with four computers operating within the network on Ethernet.
As the study focuses on observing the detection of anti-packet sniffing tools against packet sniffers, the sniffed content within the datagram will not be necessarily read as a whole. Thus, as long as the packets are sniffed by the sniffing tools, the encrypted information will not be decoded and studied in detail.

4.1. Computers

As some of the packet sniffing software are exclusively available only to Linux or Windows-based computers, two different computers will be used as the sniffer.
The Linux-based computer will be a Raspberry Pi 3 Model B, running its version of Kali Linux. Kali Linux is a well-known penetration testing Linux distribution, with multiple sniffing tools already incorporated in it such as the dsniff suite and Wireshark.
The Windows-based computer will be running the latest version of Windows, which is Windows 10. Installation of the several sniffing tools will be necessary to be installed.
An additional computer will be tasked on running both Snort and Suricata, the intrusion detection systems. This computer also runs Windows 10 as both Snort and Suricata can be run on Windows-based systems.
Lastly, a separate computer was connected to the network, operating independently from the sniffing/IDS computers. The separate computer's packet traffic provides a different vantage point that can be monitored by the tools in this study.
All of the computers will operate as a peer under the client-server hierarchy system established by the router. No special permissions granted to the sniffing computers unless necessary for the operation of the sniffing tool [34].

5. Evaluation

Throughout the study, the performance of each IDS were pitted against the operation of the sniffers.
A variety of settings and operational parameters will be adjusted during testing, such as the sniffing software's mode of operation in order to provide variations in attempting to bypass the detection mechanism.
The sniffers' operation will also be characterized according to their method of sniffing in order to observe if a particular detection tool is effective against a particular sniffing method.
The sniffing software, as well as the IDS, were run one at a time in order to avoid possible conflicts in their operation. This means that, for example, windump ran alone and was unable to utilize the advantages any ARP spoofing software should be able to provide if one ran in the background. However, those with ARP spoofing integrated in their package will be able to.
As the IDS operate by providing alerts (including a few details) once they detect a specific pattern or traffic, the existence of alerts will be the metric in which the IDS can be said to detect the packet sniffer. It should be noted that the alerts may or may not explicitly state the existence of sniffing operation but should be consistently alarming the user for questionable packet traffic characteristics for a minimum of three trials.

6. Results

Table 2: IDS detection results

Packet Sniffing Tools	IDS detection	
	Snort (using open snortrules 29110)	Suricata (using EmergingThreats ruleset up to 2025455, v8882)
smsniff	NO	NO
Wireshark	NO	NO
Windump (tcpdump)	NO	NO
arpspoof (dniff suite)	YES[1]	NO
Cain and Abel	YES[1]	NO
Ettercap	YES	NO
ngrep	NO	NO
Nmap	YES[2]	YES[3]

[1]Alarm after closure of ARP spoofing
[2]Non-explicit detection
[3]using Intense Scan

One of the most notable, yet expected, results of this study is that since both Snort and Suricata rulesets provide alerts based on suspicious packet traffic entering and exiting the computer's network interface card, a passive packet sniffer will remain undetectable for both of them. Here we observe that smsniff, Wireshark, windump, and ngrep remain undetected by both tested IDSes, regardless if these run promiscuously or not. These sniffing tools, however, are not capable of sniffing packets that do not flow through the device's network interface by themselves, limiting their capability in analyzing network traffic unless used in conjunction with other sniffing-related processes.
For packet sniffers that use ARP spoofing/poisoning techniques, Snort detects the operation as long as the arpspoof preprocessor, which looks for Ethernet address inconsistencies, is enabled in its configuration and the computer running Snort is a target for the packet sniffer's ARP spoofing algorithm. The only drawback is that Snort cannot identify which device initiates the ARP spoofing algorithm.
Ettercap ARP spoofing mechanism, once run, is detected by Snort during its operation once packet traffic flows to and from the targeted computer.
In the case of Cain and Abel running, Snort initially detects suspicious behavior in the HTTP responses in the ARP-spoofed com-

puter. Once the ARP spoofing session of Cain and Abel is closed, an alert regarding ARP-spoofing is issued.

Similar to Cain and Abel, Snort detects dsniff's arpspoof (not to be confused with Snort's arpspoof preprocessor) once the ARP spoofing session has been terminated and the addresses have been re-ARPed.

The only tool detected by Suricata is Nmap, which is also indirectly detected by Snort. As Nmap's Intense scan sends multiple packets across the network, the characteristics of these packets are identified by Suricata and tagged appropriately as initiated by Nmap. On the other hand, Snort does not explicitly alert the user of Nmap activity; however, the consecutive small packet transfers from Nmap trigger the threshold size limit of Snort, and the user is alerted as such. Both Snort and Suricata can identify the IP of the device initiating Nmap.

7. Conclusions

It has always been difficult to detect packet sniffers which operate passively, and this study is no exception; both of the tested IDS are not capable of detecting the tested passive sniffers such as Wireshark and windump. Rather than saying that the technology is not there yet, it would be wiser to say that as long as IDS base their rulesets with observed packet traffic, it is not possible for these detection methods to detect packet sniffers.

However, sniffers that actively perform ARP spoofing can possibly be detected by IDS such as Snort with additional preprocessors. It should be noted while that Snort actually has a method of storing IP:MAC address parings for enhanced detection of ARP spoofing, it remained unused for this study, and yet Snort was still able to detect the anomalies by itself. In addition, as IDSes analyze packet transfer in and out of the network interface, it can detect tools potentially used for probing networks such as Nmap.

Ultimately, the operation of each IDS is dependent on its installed ruleset, which is fully modifiable by the user. Once the user has grasped how the rulesets operate, they can customize the detection algorithm to suit their use.

References

[1] Ansari, S., Rajeev, S. & Chandrashekar, H. Packet Sniffing: A brief introduction. *Potentials IEEE,* Vol. 21 (5), (2002), pp. 17-19.

[2] Chomsiri, T. Sniffing packets on LAN without ARP spoofing. *IEEE in Convergence and Hybrid Information Technology ICCIT'08. Third International Conference,* Vol. 2 (1), (2008), pp. 472-477.

[3] Anh, N. & Shorey, R. Network sniffing tools for WLANs: merits and limitations. *2005 IEEE International Conference on Personal Wireless Communications,* (2005).

[4] Hu, Q., Asghar, M. & Brownlee, N. Evaluating network intrusion detection systems for high-speed networks. *Telecommunication Networks and Applications Conference (ITNAC) 2017 27th International,* (2017), pp. 1-6.

[5] Guo, K., Lu, H. & Yu, R. Packet Capture and Protocol Analysis Based on Winpcap. *2016 International Conference on Robots & Intelligent System (ICRIS),* (2016).

[6] Goyal, P. & Goyal, A. Comparative study of two most popular packet sniffing tools-Tcpdump and Wireshark. *2017 9th International Conference on Computational Intelligence and Communication Networks (CICN),* (2017).

[7] Meghana, J., Subashri, T. & Vimal, K. A survey on ARP cache poisoning and techniques for detection and mitigation. *Signal Processing Communication and Networking (ICSCN). 2017 Fourth International Conference,* (2017), pp. 1-6.

[8] Arzhakov, A. & Silnov, D. Architecture of multithreaded network scanner. *IEEE Micro/Nanotechnologies and Electron Devices (EDM) 2017 18th International Conference of Young specialists,* (2017), pp. 43-45.

[9] Bhosale, D. & Mane, V. Comparative study and analysis of network intrusion detection tools. *2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT),* (2015).

[10] Tirumala, S., Sathu, H. & Sarrafzadeh, A. Free and open source intrusion detection systems. *Machine Learning and Cybernetics (ICMLC) 2015 International Conference,* (2015).

[11] Albin, E. & Rowe, N. A realistic experimental comparison of the suricata and snort intrusion -detection systems. *Advanced Information Networking and Applications Workshops (WAINA) 26th International Conference,* (2012).

[12] Africa, A., Mesina, A., Izon, J. & Quitevis, B. Development of a Novel Android Controlled USB File Transfer Hub. *Journal of Telecommunication, Electronic and Computer Engineering,* Vol. 9 (2-8), (2017), pp. 1-5.

[13] SmartSniff. (2018). https://www.nirsoft.net/utils/smsniff.html.

[14] Wireshark. (2018). https://www.wireshark.org/.

[15] Windump. (2013). https://www.winpcap.org/windump/.

[16] Dsniff, Dug Song. (2018). https://www.monkey.org/~dugsong/dsniff/.

[17] Cain and Abel. (2018). http://www.oxid.it/cain.html.

[18] Ettercap. (2018). http://www.ettercap-project.org/ettercap/index.html.

[19] Network grep. (2018). http://ngrep.sourceforge.net/usage.html.

[20] Nmap. (2018). https://nmap.org/.

[21] Africa, A., Aguilar, J., Lim Jr., C., Pacheco, P. & Rodrin, S. Automated Aquaculture System that Regulates Ph, Temperature and Ammonia. *9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM),* (2017).

[22] S.Dhar. (2018). http://www.just.edu.jo/~tawalbeh/nyit/incs745/presentations/Sniffers.pdf.

[23] A. Africa, A Rough Set-Based Expert System for diagnosing information system communication networks. *International Journal of Information and Communication Technology,* Vol. 11 (4), (2017), pp. 496-512.

[24] Africa, A., Bautista, S., Lardizabal, F., Patron, J. & Santos, A. Minimizing Passenger Congestion in Train Stations through Radio Frequency Identification (RFID) coupled with Database Monitoring System. *ARPN Journal of Engineering and Applied Sciences,* Vol. 12 (9), (2017), pp. 2863-2869.

[25] Africa, A. & Cabatuan, M. A Rough Set Based Data Model for Breast Cancer Mammographic Mass Diagnostics. *International Journal of Biomedical Engineering and Technology,* Vol. 18 (4), (2015), pp. 359-369.

[26] Africa, A. A Rough Set Based Solar Powered Flood Water Purification System with a Fuzzy Logic Model. *ARPN Journal of Engineering and Applied Sciences,* Vol. 12 (3), (2017), pp. 638-647.

[27] Africa, A. A Mathematical Fuzzy Logic Control Systems Model Using Rough Set Theory for Robot Applications. *Journal of Telecommunication, Electronic and Computer Engineering,* Vol. 9 (2-8), (2017), pp. 7-11.

[28] Brucal, S., Africa, A. & Dadios, E. Female Voice Recognition using Artificial Neural Networks and MATLAB Voicebox Toolbox. *Journal of Telecommunication, Electronic and Computer Engineering,* Vol. 10 (1-4), (2018), pp. 133-138.

[29] Africa, A. & Velasco, J. Development of a Urine Strip Analyzer using Artificial Neural Network using an Android Phone. *ARPN Journal of Engineering and Applied Sciences,* Vol. 12 (6), (2017), pp. 1706-1712.

[30] Loresco, P. & Africa, A. ECG Print-out Features Extraction Using Spatial-Oriented Image Processing Techniques. *Journal of Telecommunication, Electronic and Computer Engineering,* Vol. 10 (1-5), (2018), pp. 15-20.

[31] Snort. (2018). https://www.snort.org/.

[32] Africa, A. A Logic Scoring of Preference Algorithm using ISO/IEC 25010:2011 for Open Source Web Applications Moodle and Wordpress. *ARPN Journal of Engineering and Applied Sciences,* Vol. 13 (15), (2018).

[33] Suricata. (2018). https://suricata-ids.org/.

[34] Gadde, S., Ganta, R., Gupta, A., Rao, R. & Rao, K. Securing Internet of Things (IoT) Using Honey Pots. *International Journal of Engineering and Technology,* Vol. 7 (2.7), (2018), pp. 820-824.